SAM-T04: what's new in protein-structure prediction for CASP6

Kevin Karplus*, Sol Katzman, George Shackleford, Martina Koeva, Jenny Draper, Bret Barnes, Marcia S

April 4, 2005
This is a preprint of an article accepted for publication in the CASP6 special issue of
Proteins: Structure, Function, and Bioinformatics.

Abstract

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].

We entered two automatic servers in CASP6, both of which are somewhat old: the SAM-T99 and SAM - T02 servers. These servers are essentially the same as the ones used in CASP5 [6], though the template library has grown over the past two years. Neither server had particularly impressive performance in CASP6. Results for an automatic method were submitted for evaluation as part of our CASP6 submissions, but the method has not yet been implemented as a web service, so could not participate in the evaluation of automatic servers.

For both the automatic and the human-assisted entries to CASP6, we relied heavily on our fragment-packing program, UNDERTAKER, which has undergone substantial development since CASP5. The same method was used for all targets, independent of the degree of similarity to any targets that we found, but we focussed more of our attention on new-fold and difficult fold-recognition targets, since these were the targets where we felt we could make the biggest gains by human intervention.

*email:karplus@soe.ucsc.edu Mailing address: Biomolecular 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.

One new method for our group in CASP6 was residue-residue contact prediction. We did not register enough predictions per target to be evaluated on most of them, and did only adequately on the few for which there enough predictions for the assessment method. We will not discuss contact prediction in this paper, but are preparing a separate paper explaining our method an analyzing the results.

According to the CASP6 assessors, our group had good results in the non-template category, so improvements in the fragment-packing program, UNDERTAKER, will be the main focus of this paper.

2 Methods

Although it has become popular to apply different techniques for targets with easily found templates and targets without templates, we applied the same protocol to all targets. This protocol consisted of fold recognition and fragment generation using HMMs followed by conformation generation and scoring with a stochastic search program. Human intervention consisted mainly of adding hand-picked constraints to the cost function of the stochastic search. There was little human intervention on targets with easily-found templates, as we spent most of our time working on the hardest targets.

For each target, we submitted one or more of the fold-recognition results (doing sidechain replacement on a template backbone with no refinement), a fully automatic prediction of the complete chain, and a result with some human intervention. In Section 3, we will examine how much was gained by the automatic prediction over simple sidechain replacement, and by human intervention over

the fully automatic procedure.

The SAM-T04 pipeline is very similar to the previous generation, SAM-T02, used in CASP5 [6].

- finding similar sequences with iterative search (using SAM-T2K and SAM-T04);
- predicting local structure properties with neural nets;
- finding possible fold-recognition templates using 2-track and 3-track HMMs;
- 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).

2.1 Iterative search

The main differences in fold recognition and alignments were that we used a new iterative search method (SAM-T04) in addition to the SAM-T2K method that we introduced in 2000, and that we used more multitrack HMMs.

The new iterative search of the nonredundant protein database NR [7] differs in several minor ways from the SAM-T2K search. The most notable differences are in the prefiltering and in the regularizers used for transition probabilities.

- One of the biggest constraints on the SAM-T2K search was that all sequences in the final multiple alignment had to be found in the initial prefiltering of the database, which was done by setting a large E-value on a BLAST search [8].

  In SAM-T04, prefiltering of the database is done using one iteration of psi-BLAST [9, 10] at each iteration of the search. This change allows the search to be much more sensitive, without requiring extremely loose threhsolds on the prefilter.

  The prefilter is set to limit the number of psi-BLAST hits to 3000—this cutoff is clearly visible in Figure 1. Occasionally one gets more than 3000 sequences in the mutiple alignment, because the target sequence aligns multiple times with repeated domains in proteins (for example, 1ugnA, one of the alleles of Lir1, has 8791 sequences in the multiple alignment because many of the sequences have multiple copies of the domain).

  Although SAM-T04 is generally more sensitive than SAM-T2K, sometimes SAM-T04 gets fewer sequences. One extreme case is 1wjpA which has 9144 sequences in the SAM-T2K alignment, but only 22 in the SAM-T04 alignment. Except where the reduction in size is due to the cap on the prefilter, the reduction is generally due to tighter thresholds on the psi-blast filter than on the older blast filter. It has not yet been determined whether the the reduction has more effect on false positives or true positives.

- The transition regularizers for SAM-T99 and SAM-T2K were set to avoid "choppy" alignments that had frequent insertions and deletions, sweeping the gaps together in highly variable regions. This multiple alignments are easier for humans to read, and are generally preferred by biologists, but information is lost about residues that really do correspond. In SAM-T04, a regularizer is used that keeps the costs of gaps fairly low even in the later iterations of the iterative search. The resulting multiple alignments look worse, but seem to work better for predicting local structure and contacts. (As always during CASP season, we had to press the method into service before we had time for extensive testing.)

2.2 Local structure prediction

We continue to use neural networks to predict various local structure properties [11, 12]. We are now predicting five backbone properties (DSSP, STRIDE, STR2, α pseudo-torsion angle, Bystroff's partition of the Ramachandran plot) and two burial properties C_β coordination with a 14 Ångstrom radius sphere and a new count we call near-backbone). We also combine the various predictions to get an averaged prediction for a traditional three-state (strand, helix, other) prediction. ***

Describe str2? Describe near-backbone?

Our neural nets now have 42 inputs for each position: a one-hot encoding of the amino acid in the target sequence

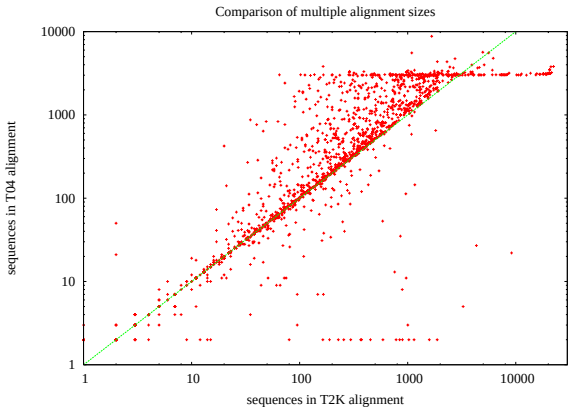

Figure 1: This figure plots the number of sequences in the SAM-T04 multiple alignments versus the number in the SAM-T2K multiple alignments, for alignments that were computed using the same version of the non-redundant protein database.
Note that the SAM-T04 method generally finds more sequences to be similar, but is usually capped at 3000 sequences by the settings of the psi-blast prefilter. The SAM-T04 alignments always contain at least two sequences, because the seed sequence is included, as is the indentical sequence found in the non-redundant protein database.

(20), a probability for each amino acid from a multiple alignment (20), and probabilities of insertion and deletion (2). The one-hot encoding of the target sequence is new and permits slightly more precise predictions when the target sequence differs from the dominant amino acid in the multiple alignment.

We have not yet done extensive testing of the new neural nets to quantify any improvement, but the combination of using the SAM-T04 multiple alignments, the extra inputs to the neural nets, and retrained networks appears to have given slight improvements in prediction of local structure.

2.3 Fragment generation

One of the most powerful operators in UNDERTAKER is fragment replacement, in which portions of the conformation are replaced by a contiguous piece of protein structure. This fragment replacement is similar to that used in Rosetta [13], but includes not just the backbone torsion angles, but full 3D information for all backbone and sidechain atoms (except hydrogens) in the fragment.

The conformation generator in undertaker uses three sources of backbone fragments for building the models:

- short, generic fragments. A library of about 1300 protein structures with good resolution is read in, and every fragment of length ≤ 4 is indexed. These *generic fragments* are used as possible replacements for exactly matching portions of the target chain.

- large fragments and alignments. For each alignment to a template found by the fold-recognition process, sidechain replacement is done and the resulting incomplete conformation stored. The sidechain replacement can be done either quickly by UNDERTAKER without optimization or by Dunbrack's SCWRL 3.0 [14, 15]. The conformation generator can use the contiguous pieces of this conformation as fragments or can do replacement of the entire conformation as a unit.

- medium-length fragments. Fragments of nine residues are found using the FRAGFINDER program of the SAM tool suite. For CASP6, we used three-track HMMs with amino-acid, str2, and C_β burial alphabets for finding medium-length fragments. The fragments are reported as short alignments to sequences in the template library, and used by undertaker in exactly the same way as longer fragments.

The main changes in fragment generation since SAM-T02 are that we now use three-track HMMs for finding the medium-length fragments, and UNDERTAKER filters the fragments as it reads in the alignments, unaligning residues that would be in improbable parts of the Ramachandran plot for that residue type. This filtering breaks some of the fragments into smaller ones, but reduces the number of residues predicted to be in the wrong conformation. We are hoping to be able to improve FRAGFINDER so that filtering its output is not necessary.

2.4 Conformation generation

The conformation generation in UNDERTAKER is an adaptive genetic algorithm that currently has 35 conformation-change operators. Three of them are the fragment replacement decribed above: InsertFragment for generic fragments, InsertSpecificFragments for fragments from fold-recognition and FRAGFINDER alignments, and InsertAlignment for replacing multiple fragments simultaneously. Also related is TwoFragment, which picks two fragments at random and replaces both. There is a standard crossover operation (CrossOver) for combining portions of different conformations, and a specialized one that does a fragment replacment at the crossover point (CrossAndInsert). Some operators do fragment replacement to try to improve specific parts of the cost function: ReduceClash, ReduceConstraint, ReduceBreak.

Another group of operators is associated with trying to close gaps in the backbone: ReduceBreak, MoveGap, CloseGap, HealGap, and HealPeptide (HealPeptide added after CASP6). Several operators move sidechains without affecting the backbone: Onerotamer, ClashingRotamer, and ClusteredRotamer. Some operators do small rigid-body movements of disconnected portions of the chain: JiggleSegment, JiggleSubtree, OptSegment, OptSubtree, OptAllSegments, and TweakMultimer (TweakMultimer added after CASP6). Some operators make small changes to torsion angles: TweakPhiSegment, TweakPhiSubtree, TweakPsiSegment, TweakPsiSubtree, TweakPsiPhiSegment, TweakPsiPhiSubtree, and TweakPeptide (TweakPeptide added after CASP6). There are also a few rather specialized operators: InsertSSBond, ImproveSSBond, ShiftSegment, and ShiftSubtree.

The genetic algorithm keeps track of which operators have made improvements in the conformations and how big these improvements were, favoring the use of operators that make large or frequent improvements.

To generate the starting conformations for the genetic algorithm, we build a random conformation, then repeatedly try doing all possible alignment replacements from our alignment library. For targets for which good templates and alignments are available, this generally gets the core of the conformation correct, and the genetic algorithm is mainly working on closing the loops and repacking sidechains, even though no part of the conformation is frozen.

2.5 Cost function

The generate-and-test method used by UNDERTAKER relies on a cost function to guide the genetic algorithm toward protein-like conformations. The cost function in UNDERTAKER is not an energy function, as it includes many non-physical terms. The cost function itself is a linear combination of any number of terms, selected at run time. There are currently 38 built-in cost function components, plus several parameterizable ones that can be read in from files. Not all the possible components were used in CASP6, and both the set used and the weighting coefficients were modified by hand on each target.

The fully automatic predictions used 14 terms:

- 6 burial terms (wet6.5, near_backbone, way_back, dry5, dry6.5, dry8, and dry12), each of which counted residues (for near_backbone and way_back) or atoms (for the others) within specific spheres near each residue. The cost function used negative log-probability of the observed burial, based on residue-specific histograms trained on a set of about 1300 good structures. The near_backbone and way_back burial functions are new—the others were used already in CASP5.

- 4 hydrogen bond terms. UNDERTAKER has a fairly sophisticated cost function for evaluating hydrogen bonds without explicit hydrogens. The cost function takes into account both distance and geometry, and uses different parameters for different types of hydrogen bonds. The different hydrogen bond terms use the same underlying cost function, but assign different weights to different classes of H-bonds. The four terms were hbond_geom (for all hydrogen bonds), hbond_backbone (giving extra weight for backbone-backbone H-bonds), hbond_geom_beta (giving still more weight for backbone H-bonds that are not part of a helix), and hbond_geom_beta_pair (giving even more weight for H-bonds that form part of a ladder between beta strands).

- 2 clash terms. Although UNDERTAKER does not have a Lennard-Jones-style energy function for Van der Waals interactions, it does have a soft_clashes

function that provides increasing penalties for worse conflicts between atoms. The definition of what consitutes a clash can be read from a file, and the particular clash table used for CASP6 grouped the atoms into 49 types and had tables for minimum acceptable distance between pairs of atom types for the same residue, residues adjacent on the backbone, and residues with separation of two or more.

The soft_clashes cost function does not distinguish between bonded and non-bonded atoms, so includes a check for bonds that are too short. UNDERTAKER does not have any other checks on bond lengths, in particular, it does not check for bonds that are too long. Since all bond lengths are copied from PDB files, the assumption is that they are all essentially good. This assumption is probabl wrong, and UNDERTAKER may need more extensive bond-length scoring.

In addition to the soft_clashes term, we used a backbone_clashes term which simply counted the number of pairs of backbone atoms that were closer than the minimum acceptable distance in the clash table.

- break cost One of the non-physical terms was a penalty for breaks in the backbone. The cost is proportional to the distance, not to distance squared, to avoid the potential problem of introducing many small gaps to break up a large one.

- constraints Another non-physical term is distance constraints between atoms. The user of undertaker can specify specific hydrogen bonds, disulfide bonds, or arbitrary atom-atom constraints. To simplify constructing the constraints, there are also commands for specifying that a particular region of the backbone is in a helix or a strand, and that a pair of regions are adjacent strands of a beta sheet, with the program producing appropriate hydrogen-bond, C_α, and C_β constraints.

  For the automatic method, helix and strand constraints were generated from the confident parts of the secondary structure predictions. Much of the human intervention consisted of adding sheet constraints to get appropriate pairing of beta strands.

- predicted α torsion angle

  In addition to the helix and strand constraints, we used the local structure predictions for the α torsion angle $(C_\alpha(-1), C_\alpha(0), C_\alpha(1), C_\alpha(2))$ as part of the cost function. The discrete probability vector from the neural net output was combined with histograms of α values to produce a nearly continuous probability distribution for each position in the chain. The negative log probability was used as a component of the cost function.

  Two components were used, based on α predictions from both SAM-T2k and SAM-T04 multiple alignments.

- hydrophobic radius of gyration

  To reward conformations that were appropriately compact, with the hydrophobic residues near the center, we included a term that was based on the radius of gyration, with atoms weighted by a hydrophobicity index for the residues. The particular hydrophobicity index used was one by Cid et al. [16]. We normalized the radius of gyration by the cube root of the length of the protein, then fit the distribution of normalized radii in our training set with a Gumbel distribution. The term of the cost function was a negative log probability of the normalized radius with this distribution.

- sidechain quality

  UNDERTAKER uses a different approach to scoring rotamers than other new-fold programs. We do not use Dunbrack's backbone-dependent rotamer library [14], nor do we compute the sidechain torsion angles. We look instead at the positions of three atoms for each residue: $C_\alpha(-1)$, $C_\alpha(+1)$, and the distal atom on the sidechain. These are put in a standard frame of reference based on the backbone atoms for the residue $N(0), C_\alpha(0), C(0)$. A mixture of Gaussian distributions for the 9-dimensional vector is used for each residue, and the negative log probability is used as a cost function. Note that this cost function gives the joint probability of the sidechain and backbone conformations for the residue, rather than a conditional probability, as is done in the backbone-dependent rotamer libraries.

  This rather crude cost function, which represents the sidechain as a single point, seems to work as well as

the more complex rotamer libraries used in SCWRL and Rosetta on the CASP6 targets. This test may not be meaningful, as the backbones were usually wrong enough that one would not expect optimal sidechains to match the experimental structures. We have not done any testing to see how the different rotamer representations work on backbones with only small errors.

- bond angle at C_α

  Nornally, the bond angles in UNDERTAKER conformations are copied from PDB files, and so are usually good. One conformation-change operator, HealGap, inserts peptide planes between adjacent C_α atoms, without paying attention to the backbone band angle at the C_α atom. We added a cost function based on the squared difference between the cosine of the bond angle and the cosine of the ideal bond angle, to penalize insertion of peptide planes that created bad bond angles. If the UNDERTAKER cost function is to be used for scoring conformations build by other programs, it may be necessary to add a general term that checks all bond angles, and not just the N-C_α-C angle.

- Ramachandran plot (bys) residue propensity ...

We sometimes added to the hand predictions terms for disulfide bonds.

2.6 Human intervention

3 Results and Discussion

3.1 Smooth GDT measure

3.2 Automatic vs. alignment

3.3 Human intervention vs. automatic

4 Conclusion

Acknowledgments

This work was supported primarily by NIH grant 1 R01 GM068570-01. Marcia Soriano was supported by a Summer Undergraduate Research Fellowship in Information Technology (SURF-IT), funded by the National Science Foundation Research Experience for Undergraduates program.

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 Rachel Karchin, Christian Barrett, Spencer Tu, Sugato Basu, Mark Diekhans, and Jonathan Casper, who made other contributions to the techniques and software.

We began work on the first few targets 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] Kevin Karplus, Rachel Karchin, Jenny Draper, Jonathan Casper, Yael Mandel-Gutfreund, Mark Diekhans, and Richard Hughey. Combining local-structure, fold-recognition, and new-fold methods for protein structure prediction. *Proteins: Structure, Function, and Genetics*, 53(S6, pages=491-496), 15October 2003.

[7] 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`.

[8] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[9] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3899–3402, 1997.

[10] Alejandro A. Schäffer, L. Aravind, Thomas L. Madden, Sergei Shavirin, John L. Spouge, Yuri I. Wolf, Eugene Koonin, and Stephen F. Altschul. Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. *Nucleic Acids Research*, 29(14):2994–3005, 2001.

[11] 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.

[12] Rachel Karchin, Melissa Cline, and Kevin Karplus. Evaluation of local structure alphabets based on residue burial. *Proteins: Structure, Function, and Genetics*, 55(3):508–518, 5 March 2004. Online: http://www3.interscience.wiley.com/cgi-bin/abstract/107632554/ABSTRACT.

[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] Michael Bower, Fred Cohen, and Roland Dunbrack. Prediction of protein side-chain rotamers from a backbone-dependent rotamer library: A new homology modeling tool. *Journal of Molecular Biology*, 267:1268–1282, 1997.

[15] Adrian A. Canutescu, Andrew A. Shelenkov, and Roland L. Dunbrack Jr. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Science*, 12:2001–2014, 2003.

[16] Hilda Cid, Marta Bunster, Mauricio Canales, and F. Gazitua. Hydrophobicity and structural classes in proteins. *Protein Engineering*, 5:373–375, 1992.