SAM-T04: What Is New in Protein–Structure Prediction for CASP6

Kevin Karplus,* Sol Katzman, George Shackleford, Martina Koeva, Jenny Draper, Bret Barnes, Marcia Soriano, and Richard Hughey
Biomolecular Engineering Department, University of California, Santa Cruz, Santa Cruz, California

ABSTRACT **The SAM-T04 method for predicting protein structures uses a single protocol across the entire range of targets, from comparative modeling to new folds. This protocol is similar to the SAM-T02 protocol used in CASP5, but has improvements in the iterative search for similar sequences in finding and aligning templates, in creating fragment libraries, in generating protein conformations, and in scoring the conformations. The automatic procedure made some improvements over simply selecting an alignment to the highest-scoring template, and human intervention made substantial improvements over the automatic procedure. The main improvements made by human intervention were from adding constraints to build (or retain) β-sheets and from splitting multidomain proteins into separate domains. The uniform protocol was moderately successful across the entire range of target difficulty, but was somewhat less successful than other approaches in CASP6 on the comparative modeling targets. Proteins 2005;Suppl 7:135–142. © 2005 Wiley-Liss, Inc.**

Key words: Fold recognition/new fold; SAM-T04; hidden Markov models; local structure prediction; UNDERTAKER protein conformation generation; FRAGFINDER protein fragment finder

INTRODUCTION

In previous CASP experiments, our team has concentrated on fold recognition using hidden Markov models (HMMS) with fairly good results.[1–3] 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 2 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 it 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 focused 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.

One new method for our group in CASP6 was residue-residue contact prediction. The CASP6 evaluation of residue-residue contacts was done at only one point: 0.2 contacts per residue. Since we registered only fairly confident predictions, for many targets we did not have enough contact predictions to be evaluated by the assessors. We will not discuss contact prediction here, but are preparing a separate article explaining our method and analyzing the results.

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

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 side-chain replacement on a template backbone with no refinement), a fully automatic prediction of the complete chain, and a result with some

Grant sponsor: National Institutes of Health; Grant number: ROI GM068570-01. Grant sponsor: Summer Undergraduate Research Fellowship in Information Technolgoy funded by National Science Foundation Research Experience for Undergraduates Program (to M. Soriano).

*Correspondence to: Kevin Karplus, Biomolecular Engineering Department, University of California, Santa Cruz, Santa Cruz, CA 95064. E-mail: karplus@soe.ucsc.edu

Received 13 April 2005; Accepted 30 June 2005

Published online 26 September 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/prot.20730

This article was originally published online as an accepted preprint. The “Published Online” date corresponds to the preprint version.

human intervention. In the Results and Discussion section, we examine how much was gained by the automatic prediction over simple side-chain 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 two- and three-track HMMs
- Making alignments to the templates
- Building a specific fragment library for the target (with FRAGFINDER)
- Packing fragments and fold-recognition alignments to make a three-dimensional (3D) structure (with UNDERTAKER).

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.

Prefiltering

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 thresholds on the prefilter. The greater sensitivity of SAM-T04 can be seen in Figure 1.

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 multiple alignment, because the target sequence aligns multiple times with repeated domains in proteins (e.g., 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 reduction has more effect on false positives or true positives.

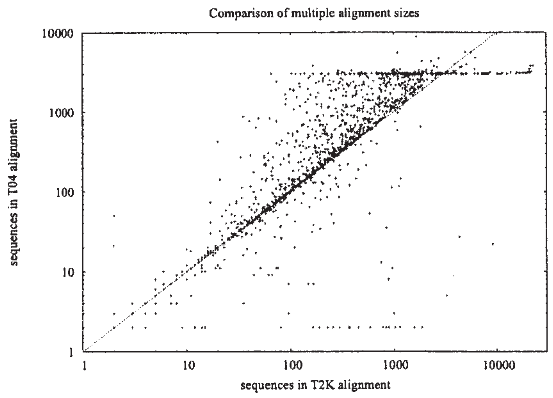

FIG. 1. Plots of the number of sequences in the SAM-T04 multiple alignments versus the number in the SAM-T2K multiple alignments, for alignments that were computed using the same version of the nonredundant protein database. Note that the SAM-T04 method generally finds more sequences to be similar, but 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 identical sequence found in the nonredundant protein database.

Regularizers

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. These 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 contacts, and we still have mixed results for predicting local structure. (As always, during CASP season, we had to press the method into service before we had time for extensive testing.)

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 pseudotorsion angle, Bystroff’s partition of the Ramachandran plot, and two burial properties, C_β coordination with a 14 Å 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.

The STR2 alphabet is based on DSSP, but the β-sheet class is broken up into 7 classes: parallel middle strand, antiparallel middle strand, mixed middle strand, parallel edge strand, antiparallel edge strand with hydrogen bonds to residue, antiparallel edge strand without hydrogen bond, and other.

The near-backbone alphabet is a burial count that counts all residues within a sphere near the residue. The center of the 9.65 Å radius sphere is placed in a fixed location relative to the backbone [at position (−2.66,

−5.15, 3.48) where C_α is at the origin, N is on the positive x axis, and C is on the xy plane with positive y), so that the sphere counted is independent of the residue. A different spot, near the backbone, defines the location of the residues to count: (1.24, 0.64, 0.23). The spot locations and sphere radius were optimized to maximize the mutual information between the residue identity and the measured burial.

Our neural nets now have 42 inputs for each position: a one-hot encoding of the amino acid in the target sequence (20 inputs), a probability for each amino acid from a multiple alignment (20 inputs), and probabilities of insertion and deletion (2 inputs). 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.

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 only the backbone torsion angles but also full 3D information for all backbone and side-chain atoms (except hydrogens) in the fragment.

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

1. 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.
2. Large fragments and alignments. For each alignment to a template found by the fold recognition process, side-chain replacement is done and the resulting incomplete conformation stored. The side-chain 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 replace the entire conformation as a unit.
3. 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.

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 described 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 replacement at the crossover point (CrossAndInsert). Some operators do fragment replacement to try to improve specific parts of the cost function: ReduceClash, ReduceConstraint, and 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 side-chains 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 was added after CASP6). Some operators make small changes to torsion angles: TweakPhiSegment, TweakPhiSubtree, TweakPsiSegment, TweakPsiSubtree, TweakPsiPhiSegment, TweakPsiPhiSubtree, and TweakPeptide (TweakPeptide was 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 are, 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 side-chains, even though no part of the conformation is frozen.

Cost Function

The generate-and-test method used by UNDERTAKER relies on a cost function to guide the genetic algorithm toward proteinlike conformations. The cost function in

UNDERTAKER is not an energy function, as it includes many nonphysical 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:

- Six burial terms (wet6.5, near_backbone, way_back, dry5, dry6.5, dry8, and dry12). Each burial term counts residues (for near_backbone and way_back) or atoms (for the others) within specific spheres near each residue. The cost function uses 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.
- Four 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 (H-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 β-strands).
- Two 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 constitutes 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 nonbonded atoms, so it 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 Protein Data Bank (PDB) files, the assumption is that they are all essentially good. This assumption is probably wrong, and UNDERTAKER may need more extensive bond-length scoring.

  In addition to the soft_clashes term, we used a backbone_clashes term that 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 nonphysical 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 nonphysical 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 β-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 β-strands. We also experimented with adding constraints based on residue-residue contact predictions.

- 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 we used was 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.
- Side-chain 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 side-chain 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 side-chain. 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 nine-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 side-chain 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 side-chain 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 incorrect enough that one would not expect optimal side-chains 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. Normally, the bond angles in UNDERTAKER conformations are copied from PDB files, so they are usually good. One conformation-change operator, HealGap, inserts peptide planes between adjacent C_α atoms, without paying attention to the backbone bond 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 built 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 have a simple residue-propensity score for a partition of the Ramachandran plot. The particular partition we use is one proposed by Bystroff et al.[17] We have a neural net that does position-specific prediction, but we are not yet feeding this prediction into UNDERTAKER's cost function.

Human Intervention

Human intervention was mainly in the form of changes to the cost function—increasing or decreasing weights for the terms and adding disulfide bonds, hydrogen bonds, or other constraints. Human intervention was most valuable in new-fold prediction, especially when we tried various β-sheet topologies.

RESULTS AND DISCUSSION

Smooth GDT Measure

The Global Distance Test (GDT) score used in CASP assessment has a moderately serious problem with quantization. That statistic averages the number of points that can be superimposed to ≤1, ≤2, ≤4, ≤8 Å, and tiny differences can move a pair of points over one of the thresholds, resulting in relatively large changes in the GDT score.

We can recast the GDT score function as a normalized sum of a goodness function:

$$\sum_{1\le i\le n} \mathrm{good}[\mathrm{dist}(i)]/[n^*\mathrm{good}(0)].$$

where n is the number of residues, dist(i) is the minimum over all superpositions tried of the ith largest distance between corresponding C_α atoms, and good(d) = $d \le 1?4 : \{d \le 2?3 : [d \le 4?2 : (d \le 8?1 : 0)]\}$.

We can define a smooth curve to replace good(d):

$$\mathrm{sgood(d)} = \left\{\begin{array}{ll} \mathrm{w_{max}} & \mathrm{d} < \mathrm{d_{max}}2^{-\mathrm{w_{max}}} \\ 0 & \mathrm{d} > \mathrm{d_{max}} \\ -\log_2(\mathrm{d}/\mathrm{d_{max}}) & \text{otherwise} \end{array}\right\}$$

We have used $d_{\max} = 12$ and $w_{\max} = 4.5$ to get results a little bit smaller than the standard GDT method, but without the quantization problems.

A similar measure with a slightly different goodness function, TM-score, has recently been proposed.[18] The main difference between smooth GDT and TM-score is that TM-score has a correction for protein length lacking in smooth GDT.

The smooth GDT measure fixes one problem of GDT—that of quantization error producing very different results for only slightly different conformations. TM-score fixes that problem and also the problem that the fraction of superposable atoms is generally larger for small proteins than for large, even for unrelated proteins. Neither measure addresses some of the other major problems of GDT:

- Smooth GDT and TM-score still suffer from having no penalty for badly predicted residues; it may be better to have a goodness function that goes slightly negative beyond 10 or 12 Å.
- Both GDT and smooth GDT tend to give too high a score to overcompacted structures, as there is no penalty for physically unrealizable conformations, and shrinking a wrong conformation may make it easier to superimpose with a correct one.
- Both smooth GDT and GDT rely on sampling many superpositions of the conformations to compute how any atoms can be superimposed at each distance. Different sampling techniques can result in different results (though smooth GDT is a little less sensitive to this problem than GDT).

Automatic Versus Alignment

We have a fairly complicated procedure for automatically creating a model from a set of alignments and fragments using UNDERTAKER. One of the first questions to ask is whether this method is any better than just doing side-chain replacement on the best-scoring alignment.

Comparing the smooth GDT scores for both (Figure 2), we see that the automatic method helps more often than it hurts. There is generally more improvement on the harder models, where the best-scoring alignment is less likely to be correct. The points along the y axis represent domains that are not present in the best alignments, but which are included from other alignments by the automatic method.

There are several targets for which just doing side-chain replacement would produce better models than the automatic SAM-T04 method. Let's examine T0234, T0213, T0223_1, T0199_2, and T0228 to see what went wrong.

In T0234, we had good alignments to 1dnlA, but the secondary structure prediction from the neural nets was poor, so the automatically generated constraints moved the conformation away from the alignment. We replaced the automatic constraints with constraints based on the

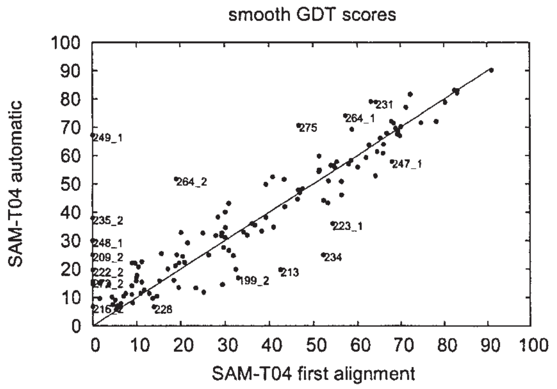

FIG. 2. Smooth GDT scores for models built by the automatic SAM-T04 method versus the models built by sidechain replacement using a single alignment with no further optimization. GDT and smooth GDT are measures of the percentage of the C_α atoms that can be made to be close when superimposing two conformations, so the best score is 100 and the worst is 0. See text for the difference between GDT and smooth GDT. The points along the *y* axis represent domains that are missing from the alignment chosen, but picked up from other alignments by the automatic method.

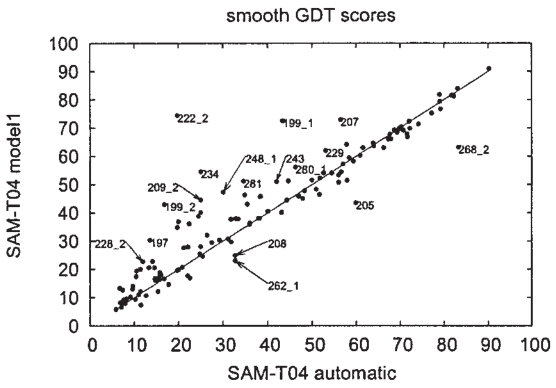

FIG. 3. Smooth GDT scores for models with human intervention versus the models built by our fully automatic method.

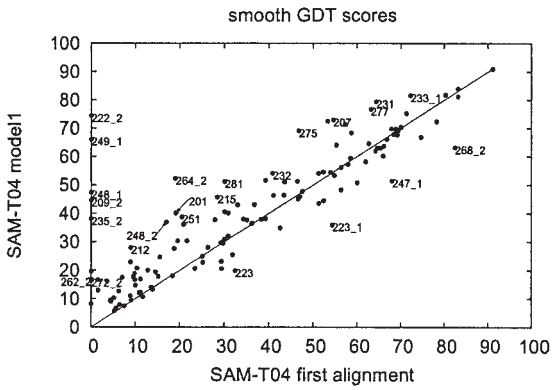

FIG. 4. Smooth GDT scores for models with human intervention versus the models built by side-chain replacement using a single alignment with no further optimization.

top-scoring alignment, and improved slightly on the initial alignment.

In T0213, the automatic run was done without 1t62A in the template library, but the alignments were redone after 1t62A was added, so the automatic run did not mess up the alignment—it didn't have it to work from.

In T0223_1, we had an okay alignment to 1nox, which is a domain-swapped dimer. Our secondary structure predictions were not a good match to the template, and we would have been better off discarding them and using just the template. Because we had initially classified T0223 as a comparative model (due to the strong hit to 1nox and other templates in the same family), we did not give it the attention it needed, and submitted a model no better than the automatic prediction.

In T0199_2, the automatic method was applied to the entire multidomain protein. The third domain was improved, but at considerable cost to the first two domains. For the hand prediction, we picked up some constraints from the alignments for sheets that appeared to have been damaged in the first two UNDERTAKER runs. We split the protein by hand into overlapping pieces (1–133, 133–end), that did not really match the domain boundaries (14–87; 116–142, 230–336; 145–226), but did help us find better alignments for the second and third domains. We recognized that the first domain ended sooner than our first partition and created predictions, for 1–95 and 115–end. The automatic prediction for 1–95 was pasted into predictions made for the whole model. The 115–end predictions were about as good as the whole-protein predictions for domains 2 and 3, so it does not matter that we did not use them. Pulling out 1–95 as a separate domain did help quite a bit on the first domain.

In T0228, we started with an adequate alignment to 1qpoA. Because the template does not quite form the hydrogen bonds necessary to keep the sheet together, UNDERTAKER did not manage to retain the structure. We manually added sheet constraints to form the sheets seen in the top alignment, which resulted in an adequate but not particularly good prediction.

Human Intervention Versus Automatic

Part of our human intervention was recognizing when the automatic method had lost a β-sheet that was in the alignments, adding constraints to retain the β-sheet. Other human interventions were to try to create β-sheets or helical bundles when they were not present in the alignments. In the past, our group's human interventions were about as likely to hurt as to help.[6] How well did we do this time, and were we just repairing the damage done by the automatic method? We compared our Model 1 submissions with both the automatic method (Fig. 3) and the best-scoring alignments (Fig. 4).

For the most part, we improved on the fully automatic method, often substantially, but there are a few targets for which the automatic prediction was better. What went wrong for targets T0268_2, T0205, T0208, and T0262_1, where our manual prediction is considerably worse than our automatic prediction?

T0268 had a good alignment to 1m6yA. The individual domains were not damaged much by the automatic method, though the relationship between the domains got a bit worse. Very little work was done on this target, but on the second try, we somehow added a helix between R182 and R189, though no such helix was predicted by the neural nets. This extraneous helix was combined with a misalignment starting around E167, turning the good prediction into a rather poor one.

T0205 started out with an adequate alignment, but the β-sheet came apart on the second optimization run.

We stuck it back together using sheet constraints, but we had shifts of +2 and +4 on the two final strands, reducing the overall quality of the model. We decided not to fuss with the alignment of the β-sheet—clearly a mistake in retrospect.

For T0208, we worked hard to try to create a triose phosphate isomerase (TIM)-barrel out of the alternating strands and helices using sheet constraints, but the initial automatic model was more nearly correct. Residue-residue contact prediction and clustering of conserved cysteine and histidine residues did not help much, since the incorrect model satisfies the constraints as well or better than the real structure.

For T0262_1, our best model was from the alignment for the second-highest scoring template (1s29A), which did not cover any of the second domain. One mistake we made fairly early on T0262 was to try to cluster conserved histidines H85, H147, H166, and H190. We partitioned the protein in various ways, but none of them corresponded to the correct domain boundaries. Had we figured out the domain boundaries and attempted to cluster the conserved residues, we might have done much better, as they clustered well within each domain. More belief in our residue-residue contact predictions might have helped, though they were not very accurate.

There are still several targets for which our alignment to the best-scoring template provided a better model than our hand intervention, so we did not succeed in recognizing and undoing all the damage done by the automatic method. We already considered T0223_1; let's also look at T0247_1, where the automatic prediction is worse than the alignment, and the manual prediction is worse still.

For T0247, we had a decent alignment to 1pj5A, which covered the entire protein. We observed some missing hydrogen bonds and fixed them up with constraints, improving slightly the third domain. We made the first domain worse by attaching the hairpin at A16-L28 to the sheet, this was probably the most damaging change we made. We tried clustering H51, E196, R224, and R228 to make an ion-binding site (based on alignment to 1nrk), which pulled H51 in a bit too close and got the rotamer wrong for R228, but did no real harm to the backbone modeling.

Two successes for our hand method relative to the automatic method and alignments are T0222_2 and T0207.

The top alignment for target T0222 covered just the TIM-barrel domain and not T0222_2, and the automatic alignment did not improve the second domain much. We initially spent our effort fixing up small problems with the TIM barrel, and our best model for T0222_1 (try5-opt1, not submitted) did a fairly good on the TIM barrel, but did a terrible job on T0222_2. We then tried separate predictions for 1–274, 236–end, and 273–end. The best model for T0222_2 was from the 236–end predictions (try3-opt1-scwrl, not submitted) and was based on the second-best template hit (to 1msi). We pasted together predictions from 1–274 and 273–end, and did minor polishing for our final submission. The domains were not properly packed against one another, but were individually fairly good. Breaking the target into domains was essential for the quality of the T0222_2 prediction.

T0207 was not included in the CASP6 assessment, because information about the structure was leaked before the deadline. We had finished our predictions before the leak, so our results are not contaminated, but we cannot fairly compare with other predictors, since many predictors stopped working when the leak was reported. The initial alignment had some unfortunate gaps, but there were enough other alignments to members of the family for the automatic method to produce a configuration with only a few bad breaks. We were able to improve considerably on our initial alignment and automatic model, mainly by trying to pack things tighter and resolve clashes.

CONCLUSION

We successfully used a single protocol for modeling all targets, from the comparative modeling targets with easily found templates to the hardest new-fold targets.

We were interested in knowing whether this protocol was more effective for some classes of targets than others. The assessors judged our performance on the template-based models unexceptional, but among the top few groups on the analogous fold and new fold targets. When we plotted our GDT rank among all predictors provided by the CASP6 website versus our smooth GDT score (not shown), we found only a little correlation between our performance relative to others and the difficulty of the targets.

We had excellent GDT ranks (≤ 5) on targets across the entire spectrum of difficulty: T0199_1, T0229_1, and T0264_2 (comparative modeling targets); T0199_2, T0237_1, T0237_2, and T0237_3 (easy fold recognition); T0199_3, T0230, and T0248_1 (hard fold recognition), and T0201 and T0241_1 (new fold targets).

On the other hand we had terrible ranks (≥ 70) primarily on comparative modeling targets: T0205, T0264, T0268 and T0268_2, T0274, T0279 and T0279_1, and T0282 (all comparative modeling) and T0198 (hard fold recognition).

The relatively poor performance on comparative modeling probably has two causes: (1) We did not put much human effort into the comparative modeling, and (2) our fold recognition method is designed for finding remote homologs and does not have any procedure for selecting between templates when multiple close templates are found.

We plan to work on improving our alignments to templates, doing a better job of selecting templates and alignments when several templates are found, providing improved ways to communicate information from the fold-recognition programs to UNDERTAKER, providing more conformation-change operators in UNDERTAKER, and improving UNDERTAKER's cost function for evaluating conformations.

A longer term goal is to use our neural nets and UNDERTAKER to do protein design, as well as protein structure prediction.

All our predictions, including all models generated and all our working notes, are available at http://www.soe.ucsc.edu/~karplus/casp6/

ACKNOWLEDGMENTS

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 laboratory, and conversations with members of that laboratory were fruitful in guiding our initial work on these targets.

REFERENCES

1. Karplus K, Sjölander K, Barrett C, Cline M, Haussler D, Hughey R, Holm L, Sander C. Predicting protein structure using hidden Markov models. Proteins 1997; Suppl 1:134–139.
2. Karplus K, Barrett C, Cline M, Diekhans M, Grate L, Hughey R. Predicting protein structure using only sequence information. Proteins 1999; Suppl 3:121–125.
3. Karplus K, Karchin R, Barrett C, Tu S, Cline M, Diekhans M, Grate L, Casper J, Hughey R. What is the value added by human intervention in protein structure prediction? Proteins 2001; 45(Suppl 5):86–91.
4. Karplus K, Barrett C, Hughey R. Hidden Markov models for detecting remote protein homologies. Bioinformatics 1998;14:846–856.
5. Eyrich VA, Marti-Renom MA, Przybylski D, Madhusudhan MS, Fiser A, Pazos F, Valencia A, Sali A, Rost B. EVA: continuous automatic evaluation of protein structure prediction servers. Bioinformatics 2001;17:1242–1243.
6. Karplus K, Karchin R, Draper J, Casper J, Mandel-Gutfreund Y, Diekhans M, Hughey R. Combining local-structure, fold-recognition, and new-fold methods for protein structure prediction. Proteins 2003;53(Suppl 6):491–496.
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. Altschul S, Gish W, Miller W, Myers E, Lipman D. Basic local alignment search tool. J Mol Biol 1990;215:403–410.
9. Altschul S, Madden T, Schaffer A, Zhang J, Zhang Z, Miller W, Lipman D. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997;25: 3389–3402.
10. Schäffer AA, Aravind L, Madden TL, Shavirin S, Spouge JL, Wolf YI, Koonin E, Altschul SF. Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. Nucleic Acids Res 2001;29:2994–3005.
11. Karchin R, Cline M, Mandel-Gutfreund Y, Karplus K. Hidden Markov models that use predicted local structure for fold recognition: alphabets of backbone geometry. Proteins 2003;51:504–514.
12. Karchin R, Cline M, Karplus K. Evaluation of local structure alphabets based on residue burial. Proteins 2004;55:508–518.
13. Simons KT, Bonneau R, Ruczinski I, Baker D. Ab initio protein structure prediction of CASP III targets using ROSETTA. Proteins 1999;Suppl3:171–176.
14. Bower M, Cohen F, Dunbrack R. Prediction of protein side-chain rotamers from a backbone-dependent rotamer library: a new homology modeling tool. J Mol Biol 1997;267:1268–1282.
15. Canutescu AA, Shelenkov AA, Dunbrack RL Jr. A graph-theory algorithm for rapid protein side-chain prediction. Protein Sci 2003;12:2001–2014.
16. Cid H, Bunster M, Canales M, Gazitua F. Hydrophobicity and structural classes in proteins. Protein Eng 1992;5:373–375.
17. Bystroff C, Thorsson V, Baker D. HMMSTR: a hidden Markov model for local sequence–structure correlations in proteins. J Mol Biol 2000;301:173–190.
18. Zhang Y, Skolnick J. Scoring function for the automated assessment of protein structure template quality. Proteins 2004;57:702–710.