PROTEINS: Structure, Function, and Genetics 40:343–354 (2000)

Protein Threading using PROSPECT: Design and Evaluation

Ying Xu* **and Dong Xu**
Computational Biosciences Section, Life Sciences Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee

ABSTRACT **The computer system PROSPECT for the protein fold recognition using the threading method is described and evaluated in this article. For a given target protein sequence and a template structure, PROSPECT guarantees to find a globally optimal threading alignment between the two. The scoring function for a threading alignment employed in PROSPECT consists of four additive terms: i) a mutation term, ii) a singleton fitness term, iii) a pairwise-contact potential term, and iv) alignment gap penalties. The current version of PROSPECT considers pair contacts only between core (α-helix or β-strand) residues and alignment gaps only in loop regions. PROSPECT finds a globally optimal threading efficiently when pairwise contacts are considered only between residues that are spatially close (7 Å or less between the C_β atoms in the current implementation). On a test set consisting of 137 pairs of target-template proteins, each pair being from the same superfamily and having sequence identity ≤ 30%, PROSPECT recognizes 69% of the templates correctly and aligns 66% of the structurally alignable residues correctly. These numbers may be compared with the 55% fold recognition and 64% alignment accuracy for the same test set using only scoring terms i), ii), and (iv), indicating the significant contribution from the contact term. The fold recognition and alignment accuracy are further improved to 72% and 74%, respectively, when the secondary structure information predicted by the PHD program is used in scoring. PROSPECT also allows a user to incorporate constraints about a target protein, e.g., disulfide bonds, active sites, and NOE distance restraints, into the threading process. The system rigorously finds a globally optimal threading under the specified constraints. Test results have shown that the constraints can further improve the performance of PROSPECT. Proteins 2000;40:343–354.** © 2000 Wiley-Liss, Inc.

Key words: protein threading; fold recognition; protein structure prediction; constrained threading; combinatorial optimization

INTRODUCTION

Protein threading [1–9] is expected to play a significant role in protein structure determination in the post-genome era as genome projects continue to produce protein sequences at a rate magnitudes higher than that at which protein structures are currently being determined experimentally. In particular, protein threading can help to i) select a "small" representative set of proteins, which have "unique" folds, for experimental structure determination, and ii) computationally model the structures of the rest of the proteins based on the experimental structures of the selected ones, in the structural genomics projects.[10] Developing more effective threading methodology to meet these needs represents a significant challenge. Challenging research problems are posed by the following four key components of a threading approach,[11] i.e., construction of a structural-template library; development of a scoring function for threading alignment; design of a search algorithm for the best threading alignment; and evaluation of a best-scoring threading alignment. Our main focus in this article is on the development of a mathematically rigorous and computationally efficient threading algorithm. We have previously published a divide-and-conquer algorithm[8] for efficiently finding the optimal threading alignment when considering both pair contacts between spatially nearby residues and variable length alignment gaps. Throughout this article, the optimal threading alignment means the globally optimal threading alignment.) A key contribution of that work is to have demonstrated that such an optimal threading problem can be solved efficiently when a widely-accepted cutoff distance for pair contacts is used.

In this article, we report an improved and generalized version of the divide-and-conquer algorithm, implemented as a computer program called PROSPECT (PROtein Structure Prediction and Evaluation Computer Toolkit; a detailed manual of the program can be found at http://compbio.ornl.gov/structure/prospect/), and present an analysis of its threading performance on a large set of proteins. The key improvements in the current threading algorithm include the following: 1) it rigorously generalizes the previous algorithm, which considered only core residues, to deal with loop alignments as well; 2) it significantly improves the computational efficiency; and 3) it allows known (partial) structural information about a target protein to be used as constraints in the threading

Grant sponsor: Office of Health and Environmental Research, U.S. Department of Energy; Grant number: DE-AC05-96OR22464.

*Correspondence to: Ying Xu, Computational Biosciences, Oak Ridge National Laboratory, 1060 Commerce Park Drive, Oak Ridge, TN 37830-6480. E-mail: xyn@ornl.gov

Received 21 October 1999; Accepted 3 March 2000

process, and finds an optimal threading under these constraints.

To thoroughly evaluate PROSPECT's performance on fold recognition and threading-alignment accuracy, we have run it on a set of 499 pairs of proteins. This set is divided into four subsets: training and testing sets (with 175 and 137 pairs, respectively), consisting of proteins that are evolutionarily related, and training and testing sets (with 100 and 87 pairs, respectively), consisting of proteins that are not evolutionarily related. The training data are used to determine the relative scaling factors of the scoring terms. Similar performance results are achieved for both the training and testing sets. For evolutionarily related protein pairs, PROSPECT correctly recognizes 68–69% of the templates and correctly aligns 66-70% of alignable residues. For protein pairs that are not evolutionarily related, PROSPECT correctly recognizes 19–20% of the templates and correctly aligns 20–23% of alignable residues. These numbers are further improved when secondary structures predicted by PHD 12 are used in scoring.

METHOD

In this section, we present an algorithm for finding the optimal threading alignment between a target sequence and a template fold, measured by a statistics-based "energy" function. Currently the energy function has the following form:

$$E_{\text{total}} = \omega_{\text{mutate}}E_{\text{mutate}} + \omega_{\text{single}}E_{\text{single}} + \omega_{\text{pair}}E_{\text{pair}} + \omega_{\text{gap}}E_{\text{gap}}. \quad (1)$$

The mutation energy E_{mutate} is the sum of the compatibility measurements $e_{\text{mutate}}(a_1; a_2)$ for substituting template amino acid a_1 by target amino acid a_2. We use the PAM250 matrix[13] for $e_{\text{mutate}}()$. The singleton energy Esingle represents the sum of the preferences esingle(a; s; t) for aligning amino acid a of the target sequence onto a template position with a structural environment defined by secondary structure s and solvent accessibility t. E_{pair} is the sum of pair-contact potentials $e_{\text{pair}}(a_1; a_2)$ between amino acids a_1 and a_2 of the target sequence when they are aligned to template positions that are spatially close. In the current version of PROSPECT, the cutoff is set at 7 Å between the C_β atoms of a_1 and a_2. E_{gap} is the sum of the penalties $e_{\text{gap}}(g) = 10.8+0.6*(g-1)$ for an alignment gap of length g.[13] All statistics for estimating these terms are collected from the FSSP database (released in March 1998).[14] For more detailed information on the calculation of the energy function terms, we refer the reader to reference 8. All the ω terms are scaling factors, which are determined through optimizing the threading alignments of our training set (see Results) against the structure-structure alignments.

In the current version of PROSPECT, pair-contact potentials are calculated only between residues of core secondary structures (α-helices or β-strands). Also we assume that alignment gaps are confined to loop regions. Based on these assumptions, the threading problem can be schematically represented in Figure 1. The goal is to find an alignment between the template and the target sequence so that E_{total} in Eq. (1) is minimized.

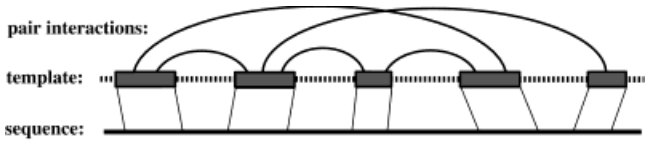

Fig. 1. A schematic of sequence-structure alignment. The line at the bottom shows the target sequence. Each box represents a core secondary structure (α-helix or β-strand) of the template. The dotted lines between boxes represent loop regions. An arc between two core secondary structures indicates that there exists at least one pairwise interaction between the two cores. The two lines between a core and the target sequence represent a gapless alignment between the core and the sequence.

A Divide-and-Conquer Approach

Our threading algorithm employs a divide-and-conquer strategy to solve the optimal threading problem. For this purpose, we pre-process the template by repeatedly dividing (bi-partitioning) it into sub-structures until each substructure contains only one core secondary structure. Dividing the template cuts an interaction between two cores into two open links, represented as an arc with one of its ends being a hollow circle as shown in Figure 2. The divide-and-conquer algorithm works correctly on any bipartition of the template. However, the way how a template is partitioned affects the computing time (see Computational Efficiency issues).

The algorithm solves the entire optimal alignment problem by recursively solving a series of sub-alignment problems between sub-structures and sub-sequences, under various constraints, and then combining these subalignments in a consistent and optimal way. Figure 3 illustrates the basic idea, using an example from the last partition step in Figure 2. In this example, the substructure AB is partitioned into two cores, A and B. The interaction link between A and B in the partition is cut into two open links, i.e., a_3 and b_1.

The algorithm first calculates the alignment score between A (similarly B) and each sequence position (meaning that the leftmost residue of A is aligned with that position). Since we assume that there is no alignment gap within a core alignment, this score can be calculated by simply adding the singleton scores, E_{single}, of the aligned residues and structural positions, plus the E_{pair} scores. The calculation of E_{pair} is tricky since we do not know which sequence positions are aligned to the cores at the other ends of the open links a_1, a_2, and a_3 (i.e., the first, fourth, and third cores in Fig. 2). To overcome this, we simply consider all possible legal alignments of these cores. Note that not every combination of the alignments of these cores makes a legal (overall) alignment since some of them may 1) violate the relative order of these cores (e.g., the first core is aligned to a sequence position that is to the right of the aligned sequence position of the fourth core); 2) overlap with each other; and 3) violate the allowed minimum and maximum length difference in loop alignments (we allow a user to specify these numbers in PROSPECT). Though now we have to consider many possible aligned positions of the cores connected with a_1; a_2; a_3 (from now on, we simply call them the assignments to a_1, a_2, and a_3),

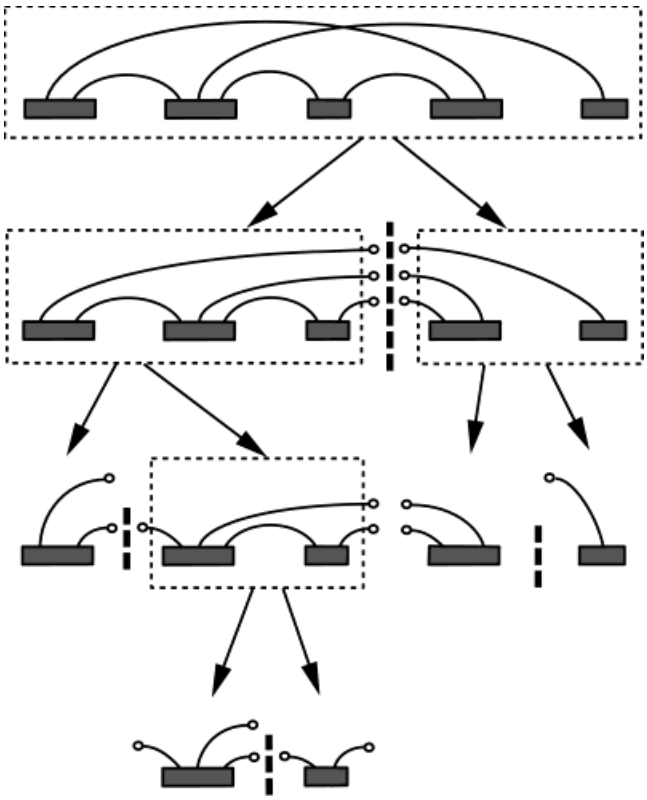

Fig. 2. A partition of a template structure. The first row shows the template with five core secondary structures of Figure 1. The second row shows a partition of the template into two substructures, one with three cores and the other with two cores. A broken arc ended with a circle is called an open link. The third and fourth rows show further partition of the template until each sub-structure contains only one core secondary structure. Note that a partition forms a tree structure as indicated by the arrows.

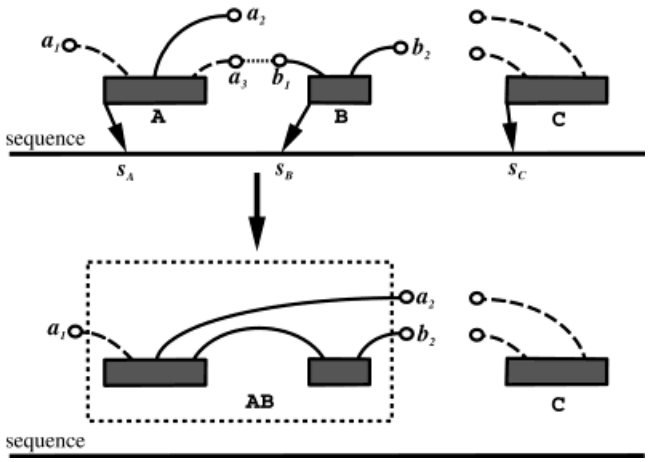

Fig. 3. A schematic example of the divide-and-conquer algorithm (see the text in the article for details).

we have enough information to calculate the pair contact term E_{pair} for each fixed assignment to the open links.

To avoid double counting, we treat the two open links created from each interaction differently. We use a dashed arc to represent one open link and a solid arc to represent the other. We only calculate the pair contact energy E_{pair} for the solid arc, while both open links are assigned to the aligned position of the core with a dashed arc. Here only a_2 is used to calculate Epair, and the corresponding E_{pair} terms for a_1 and a_3 will be calculated when the alignment for the first and the third cores are calculated, respectively. In addition, $a_3 = b_1 = s_A$, where s_A is the aligned position of the leftmost residue of core A on the sequence.

After the algorithm calculates the core alignment scores for each core under various assumptions that their open links are assigned to particular sequence positions, it calculates the alignment scores for larger sub-structures consisting of multiple cores. We continue to use the above example to illustrate the basic idea. We now want to calculate the optimal alignment score for AB under the assumption that a_1, a_2, and b_2 are assigned with particular sequence positions. Calculation of this optimal score consists of two parts: i) the sum of the alignment scores for A and for B under the condition that the two partial alignments for A and B are consistent, and ii) the alignment score of the loop between A and B. The optimal alignment score for AB is the lowest combined score of i) and ii) among all possible legal assignments to a_3 and b_1. To find the lowest combined score, the algorithm goes through all the alignment scores for A and B under the conditions a) that a_3 and b_1 have the same sequence-position assignment, and b) that $a_2 = b_2$, since they point to the same core (i.e., core C, see Fig. 3). For each such assignment, the algorithm calculates the optimal loop alignment (now the aligned positions of both A and B are fixed) using the dynamic programming method,[15] and adds up the total alignment score for AB. The assignment to a_3 (b_1) which has the lowest combined score gives the optimal alignment score for AB under the specified assignment condition.

This process continues for merging AB and C until the top level of the partition tree is reached, i.e., the whole template is considered. Note that in the lower level calculations, the algorithm repeatedly solves constrained alignment problems, e.g., finding the optimal alignment score for a sub-structure under the condition that the open links of the sub-structure have particular assignments. On the top level, the whole structure has no open links, and the optimal alignment obtained gives the final solution to our threading problem.

It is worth mentioning that the algorithm is not an exhaustive search algorithm, i.e., it does not explicitly go through all possible alignments. A simple example is that the sub-optimal alignments in AB are examined independently of sub-optimal alignments in another merged block, and no combination between these sub-optimal alignments is ever examined. However, it can be shown mathematically[8] that this algorithm guarantees to find a globally optimal threading for an energy function in Eq. (1).

Computational Efficiency Issues

We have found that the computational bottleneck of this algorithm is the consideration of all legal combinations of link assignments. More specifically, the dominating term

TABLE I. Distribution of Topological Complexity†

TC\cutoff (Å)	5	6	7	8	9	10	11	12	13	14	15
0	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)	52 (7%)
1	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)	87 (12%)
2	218 (29%)	162 (22%)	135 (18%)	118 (16%)	112 (15%)	106 (14%)	102 (14%)	98 (13%)	96 (13%)	94 (13%)	92 (12%)
3	364 (49%)	318 (42%)	245 (33%)	189 (25%)	152 (20%)	131 (17%)	114 (15%)	104 (14%)	93 (12%)	83 (11%)	77 (10%)
4	29 (4%)	126 (17%)	216 (29%)	245 (33%)	218 (29%)	198 (26%)	185 (25%)	175 (23%)	161 (21%)	156 (21%)	154 (21%)
5	0 (0%)	5 (.7%)	15 (2%)	58 (8%)	121 (16%)	156 (21%)	162 (22%)	151 (20%)	136 (18%)	120 (16%)	105 (14%)
6	0 (0%)	0 (0%)	0 (0%)	1 (.1%)	8 (1%)	19 (3%)	47 (6%)	82 (11%)	117 (16%)	135 (18%)	141 (19%)
7	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (.1%)	1 (.1%)	1 (.1%)	8 (1%)	23 (3%)	38 (5%)
8	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	4 (.5%)

†Each column of Table I represents a cutoff distance, and each row represents the number of proteins having a particular topological complexity (TC).

in the algorithm's computational complexity (running time) increases exponentially with the maximum number of links among all division points. Fortunately, this (maximum) number is generally a small number if we only consider pair contacts between residues that are spatially close given the optimal way to divide a template. We have previously demonstrated that the maximum number of links changes with how a template structure is divided (e.g., where to cut first, and then second, etc), and presented an algorithm for finding a division scheme which minimizes the maximum link number among all division points.[8] The minimized maximum link number is called the topological complexity (TC) of the template. The topological complexity is a small number (≤ 8) in all the cases that we have studied when the cutoff distance between the C_β atoms is no more than 15 Å. Table I shows the distribution of the topological complexity versus the cutoff distance between C_β atoms for pair contacts. The statistics were compiled from 750 structures in the FSSP database 14 (release of March 1998). The lengths of these proteins range from 31 to 793, and the numbers of their core secondary structures range from 1 to 34. Figure 4 shows the distribution of TC values vs. the lengths for 120 proteins randomly selected from this set.

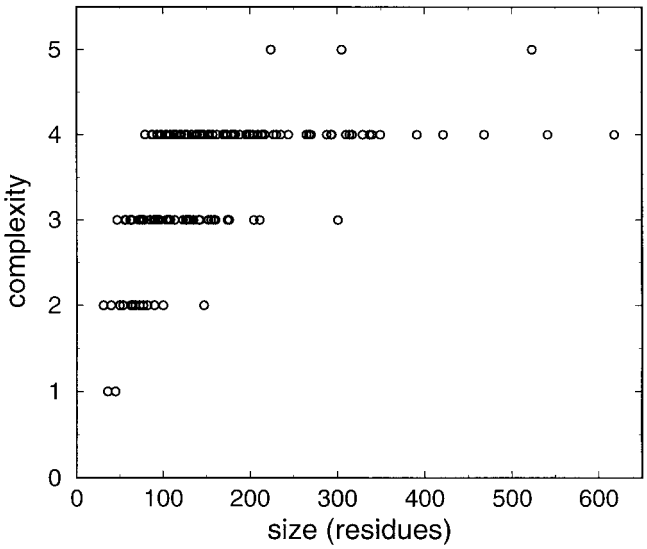

Fig. 4. Distribution of TC values vs. protein lengths. The x-axis is the protein-length axis, and the y-axis is the TC axis.

By using a similar analysis to that of reference 8, we can prove that our improved threading algorithm runs in a time proportional to $mn + M\, n^{TC} N^{TC/2}$, where m and n represent the lengths of the template and target protein sequences, M is the number of core secondary structures of the template, and N is the maximum allowed difference between the lengths of two aligned loops (the default value of N is 20 in the current version of PROSPECT). This function shows how various parameters affect the running time of the algorithm. The actual threading time on a DEC/alpha workstation typically ranges from a few seconds to several hours. A unique feature of our threading algorithm is that it runs in exponential time only with the topological complexity of a template rather than with the length of the template or target sequence as for other similar algorithms.[16, 17]

Information-Constrained Threading

PROSPECT allows a user to incorporate the following types of structural information into the threading process: disulfide bonds, active sites, NOE distance restraints, or any geometric constraints defined in terms of a pairwise relationship between residues.

Such a need may exist if some partial structural information is known about a target protein before its full structure is solved. This type of information puts some constraint on a threading alignment and possibly improve the alignment accuracy and fold recognition. For example, a user may specify that residues x, y, and z of the target sequence form an active site satisfying a certain pairwise geometric relationship among them, or that the cysteines at positions p and q form a disulfide bond and hence they are at certain distance range.

PROSPECT finds an optimal threading alignment satisfying the specified constraints. It achieves this by enforcing that a pair of residues, specified by a constraint, are aligned only to structural positions of the template that satisfy the constraint. Considering the potential structural difference between the native structure of a target sequence and the template, PROSPECT uses hard constraints to rule out alignments that significantly deviate

from the specified constraints, and also applies soft constraints (penalty functions) to force the alignments to be close to the specified constraints.

PROSPECT also provides a framework for a user to easily add position-dependent information about a target sequence as soft constraints in the threading process by adding extra penalty (or reward) terms to the energy function (Eq. 1). The relative weights of the added penalty terms can be specified by the user or determined through a training process. We have tested this using a) the secondary structure predicted by PHD[12] and b) the position-dependent profile of the target sequence calculated by the SAM program,[18,19] which provides the probability of having each of the 20 amino acids in a particular position based on the multiple-sequence alignment.

RESULTS

We have selected 499[†] pairs of structurally-aligned proteins from the data set of 20 to evaluate PROSPECT's performance on fold recognition and threading alignments. This set consists of 312 pairs of proteins, which are annotated to be evolutionarily related and 187 pairs which are not evolutionarily related 20. For convenience, we call the first subset the superfamily set and the second one the fold-family set. We further divide each of the two data sets into two subsets: a training set and a testing set. Training sets are used to optimize the scaling factors, $\{\omega_{single}; \omega_{mutate}; \omega_{pair}; \omega_{gap}\}$, of our energy function (Eq. 1). The superfamily training set contains 175 protein pairs, consisting of 175 target sequences and 149 unique templates (some pairs share the same templates), and the testing set contains 137 pairs with 137 target sequences and 123 templates. The fold-family training and testing sets have 100 and 87 pairs, respectively. The sequence-template pairs can be found in Appendices. The selection of our data set from the data list of 20 is basically arbitrary, with only one condition: proteins of each pair share less than 30% sequence identity in threading. In selecting the training pairs, we generally wanted to exclude large proteins with the consideration that training takes multiple rounds and large proteins tend to be slow to thread.

Our evaluation consists of two parts: fold recognition and threading alignment accuracy. To evaluate fold recognition, we put one protein of each pair into the target-sequence list and the other one into the template list. We run PROSPECT for each target sequence against the whole template list and assess how high it ranks the original mate by each target sequence. For evaluating threading-alignment accuracy, we compared the threading alignment of each pair with the structure-structure alignment determined by the SARF program.[7] A residue is termed correctly aligned if it is aligned to within a 4-residue shift from the SARF's structure-structure alignment position.

[†]We originally selected 550 pairs but then found that 51 pairs have sequence identity level higher than our cutoff, 30%, for the alignable portions of their structures though the global sequence identity of each pair is less than 25%.[20] We removed those 51 pairs from our data set.

Weighing Energy Terms

The scaling factors in our energy function were selected to "optimize" the average alignment accuracy for all training pairs. The alignment accuracy was defined by the number of correctly aligned residues, divided by the number of structurally-alignable positions defined by SARF.[7]

To search for the optimal parameter values, we first set a range for each parameter, based on preliminary threading results. Our search program for the optimal parameter values employed the orthogonal array method.[21] This procedure starts with a coarse search grid and can quickly approach the local optimal point by reducing the search range for each parameter and using a finer search grid in the reduced search region. For the superfamily set, the following values maximized the averaged alignment accuracy, with wmutate being set to be 1:

$$\omega_{single} = 0.31;\ \omega_{pair} = 0.31;\ \omega_{gap} = 10.4.$$

and for the fold-family set, we have decided on the following values:

$$\omega_{single} = 0.27;\ \omega_{pair} = 0.42;\ \omega_{gap} = 9.8.$$

Table II shows the contribution from each of four energy terms for 15 examples, in which the templates were all ranked as number 1.

Threading Performance for the Superfamily Set

PROSPECT correctly aligned 70.0% of the (16,936) alignable residues, for the 175 pairs of training data, using the scaling factors above. For the test set of 137 pairs, PROSPECT correctly aligned 66.3% of the (19,711) alignable residues. We believe that the 3.7% difference in the alignment accuracies between the training and testing sets is not caused by over-training (memorization of the data) as the number of trained parameters is rather small. This discrepancy may possibly be explained by the difference between the averaged sequence lengths of the training and the testing sets: 166.6 and 256.4 residues, respectively. As we have mentioned, we intentionally chose smaller proteins for the training set to save training time. Figure 5 shows that for the training set, the alignment accuracy decreases slightly as the length of a protein increases. Similar behavior was observed for the testing set.

While further study will be conducted to understand this phenomenon, one possible explanation for the decreased accuracy for larger proteins is that as the protein size increases the search space size in general increases. As we have discussed, the dominating factor in threading time (and similarly, the threading search space) is the topological complexity of a protein structure, which generally increases with its size (see Fig. 4). The increased search space may make it more difficult for our energy function to identify the correct alignment.

We then ran PROSPECT for each target sequence against the template list using the trained scaling factors. For the training set, PROSPECT ranked the correct template for each target sequence at the top in 56.0% of the cases and in

TABLE II. Energy Term Contribution in Optimal Alignments†

Pair	Class	Seq. identity	Align accuracy	Total score	Single score	Mutate score	Pair score	Loop score
1bfma-1tafa	superfamily	23.1%	63/64	−647.3	−7.8	−134.0	−34.0	−472.6
1sxl-1urna	superfamily	21.9%	73/73	−1531.6	−321.6	−705.0	−70.6	−426.6
1isdb-1rsy	superfamily	20.5%	87/100	−1690.4	−434.7	−634.0	−108.6	−513.2
1hxpa-1kpaa	superfamily	17.2%	83/90	−1123.7	−399.6	−259.0	−112.1	−355.1
1bgc-2gmfa	superfamily	16.7%	0/81	−900.2	−246.6	−504.0	−12.9	−146.7
1lbu-1vhh	superfamily	16.5%	54/82	−998.1	−399.3	−456.0	−56.2	−89.6
1cds-3ebx	superfamily	16.4%	39/53	−913.0	16.8	−432.0	−325.6	−172.2
1erv-1gp1a	superfamily	14.2%	43/89	−928.1	−379.2	−281.0	−44.2	−193.8
1slta-2ayh	superfamily	12.1%	86/111	−851.3	−679.8	−250.0	−220.7	299.2
1prtd-1prtc	superfamily	12.1%	60/86	−845.1	−379.8	−505.0	−144.8	184.5
1dih-1scua	fold-family	16.0%	76/115	−1524.3	−1327.6	−610.0	−287.3	684.3
1hgxa-1opr	fold-family	14.6%	0/90	−1392.1	−813.2	−453.0	−171.6	45.8
1ris-1spbp	fold-family	13.4%	50/62	−549.1	−256.0	−353.0	−161.0	206.7
1udii-1yua	fold-family	13.1%	38/45	−658.7	−230.0	−74.0	−114.6	−240.0
1led-2ayh	fold-family	12.4%	29/139	−1568.1	−879.4	−401.0	−376.3	88.6

†Pair is the name of a target-template pair and Class is the set a pair belongs. Seq. identity is the percentage of the identical aligned residues out of the total number of aligned residues of each alignment. Align accuracy gives the number of residues aligned correctly vs. the total number of alignable residues. Total score is the alignment score of a pair. The other four scores are the contributions to the Total score from the four energy terms, respectively.

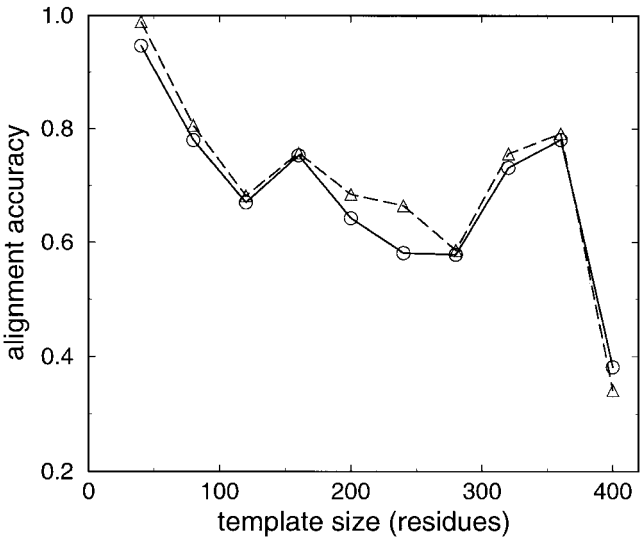

Fig. 5. The threading-alignment accuracy as a function of the template length for the superfamily training set. The dashed line with triangles is for alignment accuracy with the pair potential and the solid line with circles for accuracy without the pair term. Each data point is an average over a window size of 0.2.

the top five in 69.1% of the cases. If we use the convention of CASP3,[22] for which the top five ranked templates were submitted, we can consider that PROSPECT recognized 69.1% of the fold templates for the training set. A similar level of performance is achieved for the test set, for which 57.7% of the correct templates were ranked at the top, and 68.6% in the top five.

Table III summarizes the template ranking, when its original mate is used as the target sequence. Table IV summarizes the fold recognition accuracy and threading-alignment accuracy vs. the sequence identity level, for both the training and testing sets (train and test). It can be seen that the overall threading performance for the two sets are very similar.

As expected, PROSPECT's performance in both fold recognition and threading alignment improves as the sequence identity level increases. Based on the statistics from Table IV, we expect that PROSPECT will have a good threading performance (high fold recognition rate and $\geq$ 60% threading-alignment accuracy) when the template and target have at least 15% sequence identity.

To evaluate the contribution of pair-contact potentials to fold recognition and threading alignments, we conducted training and testing using the same data but excluding the pair-contact potential term. Different scaling factors for the remaining three energy terms were obtained using a similar procedure to the one described in the Weighing energy terms section. A comparison of PROSPECT's performance using the two energy functions (one with and one without the pair-contact potential term) is given in Tables III and IV. All results for the energy function with the pair potentials are labeled -A, and without the pair potentials, -S. While the performance on both fold recognition and threading alignment improves with pair-contact potentials, the improvement in fold recognition is more significant. From Tables III and IV, we can see that with the pair potential term PROSPECT's alignment accuracy improves about 2–3% for both the training and testing set, and its performance on fold recognition improves 11–22%, depending on different measurements. Actually, using the following formula to compare the overall rankings with and without pair potentials, the improvement is even greater:

$$\frac{1}{L}\sum_{\text{pair}}\frac{\text{rank}_s(\text{pair}) - \text{rank}_p(\text{pair})}{\max\{\text{rank}_s(\text{pair}), \text{rank}_p(\text{pair})\}}, \qquad (2)$$

TABLE III. Rank Distribution of the Correct Templates†

	1–5	6–10	11–15	16–20	21–25	26–30	31–35	3–40	41–45	46–50	51–75	76–100	101-
Train-A	121	7	5	3	4	4	0	3	6	5	5	3	9
(175)	69%	4%	3%	2%	2%	2%	0%	2%	3%	3%	3%	2%	5%
Test-A	94	9	4	2	8	5	2	1	1	2	6	1	2
(137)	69%	7%	3%	1%	6%	4%	1%	1%	1%	1%	4%	1%	1%
Train-S	102	11	9	4	3	0	5	2	2	5	14	5	13
(175)	58%	6%	5%	2%	2%	0%	3%	1%	1%	3%	8%	3%	7%
Test-S	76	6	4	6	6	3	2	5	1	5	10	10	3
(137)	55%	4%	3%	4%	4%	2%	1%	4%	1%	4%	7%	7%	2%

†Each column represents the number of pairs whose templates are ranked in that range, e.g., 121 pairs out of 175 have their templates ranked in top five, etc. The rows give the ranking distribution of the templates for the training and testing sets, respectively. -A is for results using all four energy terms, and -S is for results using all but the pair-contact term.

TABLE IV. Threading Performance Vs. Sequence Identity: Superfamily†

Seq identity	Total	1–6% (1)	7–9% (4)	9–12% (26)	13–15% (28)	16–18% (44)	19–21% (28)	22–24% (19)	25–27% (15)	28–30% (10)	Overall (175)
Train	Top-A1	0	0	2	9	25	25	13	15	9	98
				8%	32%	57%	89%	68%	100%	90%	56%
	Top-A5	0	0	4	15	32	28	17	15	10	121
		0%		15%	54%	73%	100%	89%	100%	100%	69%
	Align-A	25/88	143/421	797/2153	1564/2786	3127/4423	2704/3122	1589/1743	1022/1221	886/979	11857/16936
		28%	34%	37%	56%	71%	87%	91%	84%	91%	70%
	Top-S1	0	0	2	6	22	16	10	11	7	74
				8%	21%	50%	57%	53%	73%	70%	42%
	Top-S5	0	0	3	14	24	24	16	12	9	102
				12%	50%	55%	86%	84%	80%	90%	58%
	Align-S	25/88	164/421	860/2153	1462/2786	2898/4423	2668/3122	1549/1743	969/1221	889/979	11484/16936
		28%	39%	40%	52%	65%	85%	88%	79%	91%	67%
	Total	(1)	(3)	(24)	(36)	(19)	(15)	(18)	(16)	(5)	(137)
Test	Top-A1	0	0	6	18	7	11	17	16	4	79
				25%	50%	37%	73%	94%	100%	80%	58%
	Top-A5	0	0	9	24	10	12	18	16	5	94
				37%	67%	53%	80%	100%	100%	100%	69%
	Align-A	0/88	116/312	866/3023	3384/5650	1426/2392	1673/2106	3022/3412	2091/2253	489/505	13067/19723
		0%	37%	29%	60%	60%	79%	89%	93%	97%	66%
	Top-S1	0	0	2	10	4	8	10	11	4	49
				8%	28%	21%	53%	56%	69%	80%	36%
	Top-S5	0	0	4	18	7	11	17	15	4	74
				16%	50%	37%	73%	94%	94%	80%	55%
	Align-S	0/88	59/312	1024/3023	3165/5650	1343/2392	1623/2106	2849/3412	2033/2253	483/505	12627/19723
		0%	19%	33%	56%	56%	77%	83%	90%	95%	64%

†Each column represents a different range of sequence identity level, e.g., column 4 (4–6%) represents the pairs with the sequence identity level at 4–6%. Total is the total number of pairs in a particular sequence-identity range. Top-A1 and Top-A5 represent the numbers of pairs whose templates are ranked in top one and top five, respectively; and Align-A denotes the averaged alignment accuracy over all pair alignments in a particular sequence-identity range. All -S terms are defined similarly.

where $rank_p$(pair) and $rank_s$(pair) are ranks of a template by its target sequence given in a target-template pair with and without pair contact potentials, respectively; the sum is over all pairs in the data set; and L is the total number of pairs. Using this measure, pair potentials improve the overall ranking by 21% and 31% for the training set and testing set, respectively.

To understand why the pair potential that we are using improves the fold recognition performance more significantly than the threading-alignment accuracy, we have studied a number of cases from our training set and discovered that the key contribution of the pair potential is in ruling out bad templates. For example, for target sequence 1octc, PROSPECT, with and without the pair potential, found the same alignment accuracy of 1octc with its correct fold 1fj1a: 58 out of the 60 structurally-alignable residues within a shift of 4 from the SARF-predicted positions. However with the pair potential included, 1fj1a was ranked the number 1 template; without it, number 10, while number 1 was 1gria. As can be seen from Figure 6, the pairwise contact pattern in 1fjla is dominated by the packing between α-helices, which have a periodicity of 3.6 residues (per turn), while the pairwise contact pattern in 1gria is dominated by the packing between β-sheets, with a periodicity of 2. To make the pair contacts favorable in template 1gria, it requires a segment of 1octc to be

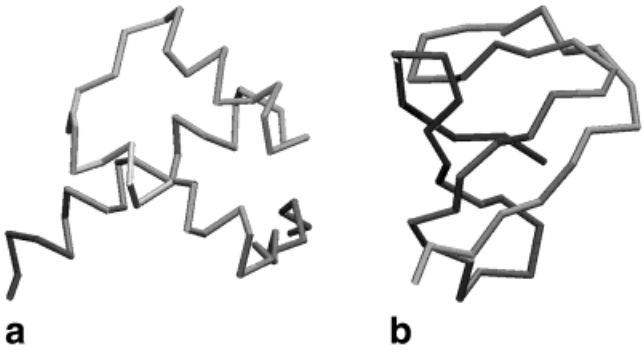

Fig. 6. The fold templates for the target sequence 1octc. **a:** The correct fold 1fjla. **b:** The wrong fold 1gria, which is ranked number 1 in threading without using the pair-contact potential. This figure was made using VMD.[27]

mounted on one β-strand of a β-sheet in contact with another segment of 1octc mounted on the other β-strand of the β-sheet. This is difficult for the fold of 1octc, whose favorable pairwise interactions are between α-helices. This probably explains why the pair terms find the pair-contact pattern exhibited in 1gria as unfavorable and down-ranked this template to the number 8, while our energy function without pair potentials cannot distinguish the two.

Threading Performance for the Fold-Family Set

Scaling factors are determined for the fold-family set through a separate training process, similar to the one above. PROSPECT aligned 23% and 20% of the alignable residues correctly for the training and testing sets, respectively. These numbers may be compared to 21% and 18%, respectively, without using the pair-contact potentials. For fold recognition, PROSPECT ranked 20% of the templates in the top five by the target sequences for both the training and testing sets. This number may be compared to 15% and 18%, respectively, without using the pair term. Table V summarizes the detailed performance for the fold-family set. It is interesting to note that PROSPECT's performance level is significantly lower for this set than for the superfamily set even comparing proteins with similar sequence identity levels. The poor performance for the fold-family echoes the general performance level in CASP3 in this category, where only a few predictions were successful in this category. More discussion on this is given in the Discussion.

Threading With Predicted Secondary Structure

The PHD program[12] predicts the secondary structures by assigning three numerical scores to each residue as the probabilities of its being in a {α-helix, β-strand, loop}, respectively. Each score ranges from 0 to 9: 0 for the lowest probability and 9 for the highest. We use the following function to measure the overall consistency of the PHD-predicted secondary structure of the target sequence with the PROSPECT-aligned secondary structure of the template:

$$E_{\mathrm{ss}} = -\sum_{\text{aligned residue } a_i} \text{score}(x|a_i \text{ aligned to secondary struct } x), \quad (3)$$

where score(x | a_i) is the PHD score for the secondary structure type x at residue a_i, and where x is the secondary structure type of the template position to which a_i is aligned. This reward function multiplied by its scaling factor ω_{ss} was added to our energy function (Eq. 1). The value $\omega_{ss} = 0.039$ was determined through optimizing the average alignment accuracy for the training set (with all the other parameters fixed). Using the PHD-predicted secondary structure, PROSPECT improves its average alignment accuracy from 66–70% to 73–74% for the superfamily set and from 20–23% to 28–29% for the fold-family set (there is a slight difference between the training and testing sets). Table VI provides detailed results.

Threading Performance With Constraints

All performance results presented above are achieved fully automatically by PROSPECT. Our experience, particularly with the CASP3 prediction exercise,[22,23] has shown that human expert input could further improve the threading performance if partial structural information exists for particular target proteins. We now give a few examples.

Example 1 (application of disulfide bond):

The protein pair 1bnd-2tgi is part of our superfamily training set. 1bnda (119 residues) is the brain derived neurotrophic factor; 2tgi (112 residues) is the transforming growth factor-β2. The sequence identity between the structurally alignable positions (49 in all) of the two is 16%. PROSPECT ranked 2tgi the 127th for the target sequence 1bnda and aligned none of the alignable residues correctly. We found in its PDB file that residues 58 and 109 of 1bnda form a disulfide bond. When this constraint was introduced into the threading process (Information-Constrained Threading section), PROSPECT correctly aligned 36 out of 49 structurally-alignable residues and ranked 2tgi the 24th.

Example 2 (application of active site)

This example is from our CASP3 predictions 23. The target protein is t0053 (CbiK protein with 264 residues). The correct template fold is 1ak1 (310 residues). The sequence identity between t0053 and 1ak1 is 11.2%. Though PROSPECT recognized 1ak1 as a template of t0053, it aligned only 1 out of its 13 core secondary structures with no residue shift. We found that 1ak1 has an active site at His-183.[24] Using the BLOCK search,[25] we identified His-145 of t0053 as the corresponding active site. We then ran PROSPECT using the constraint that His-145 of t0053 is aligned with His-183 of 1ak1. The new PROSPECT alignment has five core secondary structures aligned with no residue shift and seven more cores are aligned within four-residue shift from SARF's alignment.

TABLE V. Threading Performance Vs. Sequence Identity: Fold-Family[†]

Seq identity	Total	1–6% (1)	7–9% (5)	9–12% (33)	13–15% (34)	16–18% (22)	19–21% (3)	22–24% (2)	Overall (100)
Train	Top-A5	0	0	5	10	3	0	2	20
				15%	29%	14%	0%	100%	20%
	Align-A	0/36	39/311	370/2063	528/2320	358/1863	52/186	69/78	1416/6280
		0%	13%	18%	23%	28%	28%	88%	23%
	Top-S5	0	0	3	7	3	0	2	15
				9%	21%	14%	0%	100%	15%
	Align-S	0/36	51/311	370/2063	524/2320	291/1863	52/186	69/78	1357/6280
		0%	16%	18%	23%	22%	28%	88%	21%
	Total	(1)	(3)	(27)	(36)	(13)	(5)	(1)	(87)
Test	Top-A5	0	0	3	9	4	0	1	17
				11%	24%	30%	0%	100%	20%
	Align-A	37/46	67/123	296/2027	672/2565	66/863	7/248	36/38	1181/5910
		80%	54%	15%	26%	8%	3%	95%	20%
	Top-S5	0	0	3	9	3	0	1	16
				11%	24%	23%	0%	100%	18%
	Align-S	32/46	67/123	293/2027	658/2565	16/863	0/248	36/38	1102/5910
		70%	54%	14%	25%	2%	0%	95%	18%

[†]Each column represents a different range of sequence identity level. Total is the total number of pairs in a particular sequence-identity range. Top-A5 represents the number of pairs whose templates are ranked in top five; and Align-A denotes the averaged alignment accuracy over all pair alignments in a particular sequence-identity range. All -S terms are defined similarly.

TABLE VI. Threading Performance With Predicted Secondary Structures[†]

Set	1–6%	7–9%	9–12%	13–15%	16–18%	19–21%	22–24%	25–27%	28–30%	Overall
Superfamily-train	68%	22%	58%	64%	73%	87%	89%	84%	89%	74%
Superfamily-test	39%	38%	49%	67%	67%	82%	89%	93%	98%	73%
Fold-family-train	0%	12%	26%	31%	29%	60%	90%	—	—	29%
Fold-family-test	80%	55%	22%	35%	17%	20%	92%	—	—	28%

[†]Each column represents a different range of sequence identity level. Each row represents the averaged alignment accuracy among pairs with a particular level of sequence identity for each of the four sets: the training and testing sets of the superfamily set, and the training and testing sets of the fold-family set.

Example 3 (application of position-dependent profile)

This example is also from our CASP3 predictions.[23] The target sequence is t0083 (cyanase with 156 residues). The correct template is 1adr (76 residues). There is 10% sequence identity for the 58 alignable residues. PROSPECT correctly identified 1adr as the template of t0083, but aligned none of the 58 residues correctly. We then generated a position-dependent profile of the target sequence using the SAM server,[18,19] and applied it in PROSPECT, using an additional pseudo-energy term for the profile:

$$E_{\text{profile}} = \sum_i \sum_j p(i,j) e_{\text{mutate}}(j, t_i), \qquad (4)$$

where i is a position on the target sequence, j is an amino acid type, ti is the amino acid type of the template residue aligned with the position i, p(i; j) is the probability of the amino acid type j on the position i, and $e_{\text{mutate}}(j; t_i)$ is the compatibility of substituting amino acid type j by amino acid type t_i. With the added information, PROSPECT correctly aligned 34 residues.

DISCUSSION

It is clear to us that our pair-contact potential terms are fairly noisy and could potentially be improved. Our pair terms are currently calculated using the following simple formula:

$$\log\left(\frac{\text{the number of observed } x\text{-}y \text{ pairs within our cutoff}}{\text{the number of expected occurrences of } x\text{-}y \text{ pairs within the cutoff}}\right), \qquad (5)$$

where the expected number of occurrences by the x-y pair is calculated with the assumption that x and y are independent. An implicit assumption used in our current calculation is that these numbers have uniform distributions across different environments (defined by secondary structure and solvent accessibility). This is generally not true as we have found. Figure 7 shows that the "information content" [defined by the absolute value of Eq. (5), multiplied by 1,000] is quite different for the Arg-Asp pair when both are in α-helices and both in β-strands. Mixing them, as we have been doing, as one case has definitely reduced the information content (and signal-to-noise ratio) and hence the discriminating power of the pair-contact potentials. In the same analysis, we have also found that the distance-range in which information content is high (defined by some cutoff) varies from pair to pair. This suggests that a uniform cutoff for all pairs under all

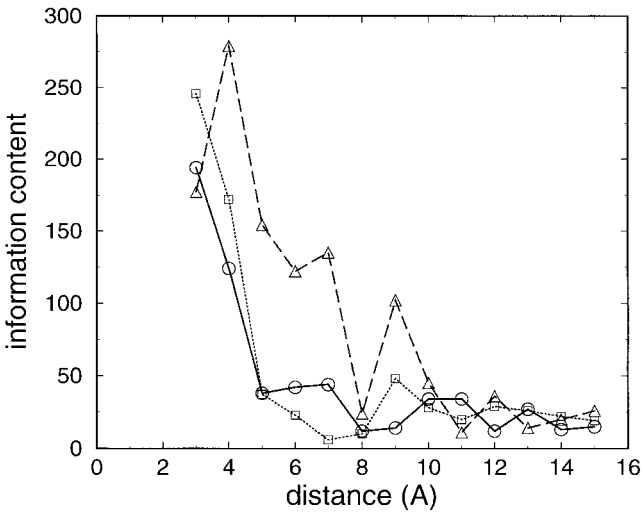

Fig. 7. Information content. The x-axis is the distance axis in Å and the y-axis is the information content. The solid line with circles is the information content measure for the Arg-Asp pair when both are in β-strands. The dashed line with triangles is for the same pair when both are in α-helices. The dotted line with squares is the information content when not distinguishing the secondary structure types. The distance is calculated between the C_β atoms.

different environments is probably not appropriate. Further research is planned to design more sensitive pair terms.

As we have shown, PROSPECT's performance is significantly worse for the fold-family set than for the superfamily set. A number of key factors contribute to the poor performance for this set.

Proteins of the same fold-family have much larger variation than proteins of the same superfamily. They generally have a larger RMSD between their alignable core elements than their counterparts from the same superfamily, and their peripheral elements of secondary structure and turn regions can have significantly different lengths and conformations[26] Hence, the solvent accessibility, the secondary structure type, and the C_β pairs of the template are less transferable to the target protein; and the threading energy has less discerning power.

The large variation in the secondary structures of peripheral elements, among proteins of the same fold-family, has also caused problems. Figure 8 shows an example in which the target protein has a short loop where the template has a long region containing two β-sheets. PROSPECT did not align a single residue correctly, mainly because deletions of secondary structures are not allowed. If we manually delete the long peripheral region in the template (residues 462–515), PROSPECT aligns 26 out of 63 structurally alignable residues correctly against this modified template. This suggests that PROSPECT's threading algorithm needs more flexibility.

In our analysis of the computational results, we also found that certain templates tend to be ranked high for many target sequences, indicating that the baseline threading scores (typical threading score against an arbitrary protein sequence) for these templates are relatively high compared to the other templates. We are currently attempting to find a method to determine the baseline score for each template. The goal is to replace our current raw threading score by a significance score through deducting the baseline scores.

In summary, we have described a new computer program for the globally-optimal threading problem and presented an analysis of its performance for a fairly large data set. We have also identified a number of research issues for further improvements.

ACKNOWLEDGMENTS

We thank Dr. J. Ralph Einstein and Dr. Edward C. Uberbacher for helpful discussions. Dr. Einstein read the first draft of the article, and his suggestions have helped improve the presentation of the article.

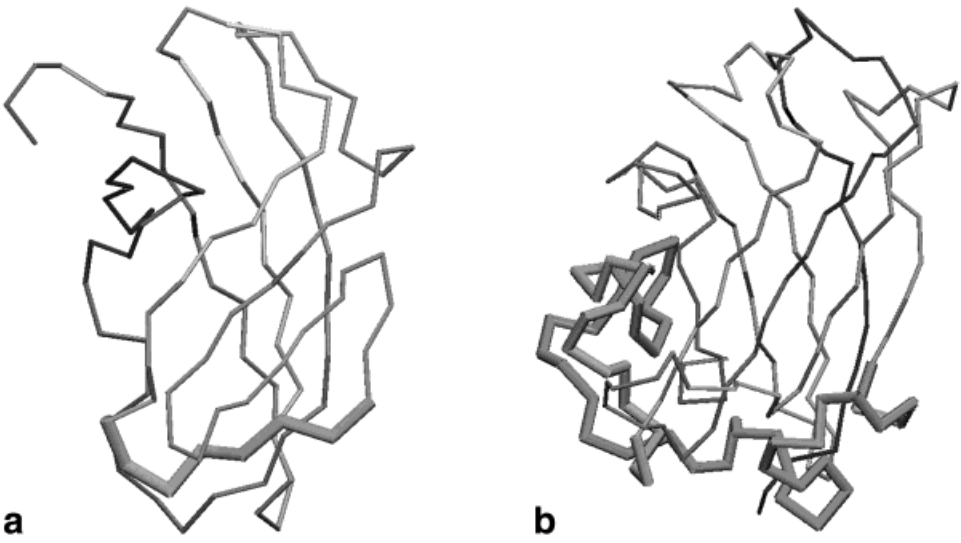

Fig. 8. A target-template pair with 1exg (**a**) as the target and 1knb (**b**) as the template. The thicker lines show the corresponding peripheral elements (residues 45–52 for 1exg and residues 462–515 for 1knb). This figure was made using VMD[27]

REFERENCES

1. Bowie JU, Luthy R, Eisenberg D. A method to identify protein sequences that fold into a known three-dimensional structure. Science 1991;253:164–170.
2. Sippl MJ, Weitckus S. Detection of native-like models for amino acid sequences of unknown three-dimensional structure in a data base of known protein conformations. Proteins 1992;13:258–271.
3. Jones DT, Taylor WR, Thornton JM. A new approach to protein fold recognition. Nature 1992;358:86–89.
4. Godzik A, Skolnick J, Kolinski A. A topology fingerprint approach to the inverse folding problem. J Mol Biol 1992;227:227–238.
5. Bryant SH, Altschul SF. Statistics of sequence-structure threading. Curr Opinion Struct Biol 1995;5:236–244.
6. Fischer D, Rice D, Bowie JU, Eisenberg D. Assigning amino acid sequences to 3-dimensional protein folds. FASEB J 1996;10:126–136.
7. Alexandrov NN, Nussinov R, Zimmer RM. Fast protein fold recognition via sequence to structure alignment and contact capacity potentials. In: Hunter L, Klein T, editors. Biocomputing: proceedings of the 1996 Pacific Symposium. Singapore: World Scientific Publishing Co.; 1996. p 53–72.
8. Xu Y, Xu D, Uberbacher EC. An efficient computational method for globally optimal threading. J Comp Biol 1998;5:597–614.
9. Crawford OH. A fast, stochastic algorithm for protein threading. Bioinformatics 1999;15:66–71.
10. National Institute of General Medical Sciences. Pilot projects for the protein structure initiative (structural genomics). http://www.nih.gov/grants/guide/rfa-files/RFA-GM-99-009.html, 1999, June:RFA GM-99-009.
11. Smith T, Conte LL, Bienkowska J, Gaitatzes C, Rogers R, Lathrop R. Current limitations to protein threading approaches. J Comp Biol 1997;4:217–225.
12. Rost B, Sander C. Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol 1993;232:584–599.
13. Gonnet GH, Cohen MA, Benner SA. Exhaustive matching of the entire protein sequence database. Science 1992;256:1443–1445.
14. Holm L, Sander C. Mapping the protein universe. Science 1996;273:595–602.
15. Smith TF, Waterman MS. Comparison of biosequences. Adv Appl Math 1981;2:482–489.
16. Bryant SH, Lawrence CE. An empirical energy function for threading protein sequence through the folding motif. Proteins 1993;16:92–112.
17. Lathrop RH, Smith, TF. Global optimum protein threading with gapped alignment and empirical pair score functions. J Mol Biol 1996;255:641–665.
18. Karplus K, Barrett C, Hughey R. Hidden Markov models for detecting remote protein homologies. Bioinformatics 1998;14:846–856.
19. Park J, Karplus K, Barrett C, Hughey R, Haussler D, Hubbard T, Chothia C. Sequence comparisons using multiple sequences detect twice as many remote homologues as pairwise methods. J Mol Biol 1998;284:1201–1210.
20. Holm L, Sander C. Decision support system for the evolutionary classification of protein structures. ISMB 1997;5:140–146.
21. Sun Z, Xia X, Guo Q, Xu D. Protein structure prediction in a 210-type lattice model: parameter optimization in the genetic algorithm using orthogonal array. J Protein Chem 1999;18:39–46.
22. CASP. Protein structure prediction issue. Proteins 1999;Suppl 3:1–237.
23. Xu Y, Xu D, Crawford OH, Einstein JR, Larimer F, Uberbacher EC, Unseren MA, Zhang G. Protein threading by PROSPECT: a prediction experiment in CASP3. Protein Eng 1999;12:899–907.
24. Al-Karadaghi S, Hansson M, Nikonov S, Jonsson B, Hederstedt L. Crystal structure of ferrochelatase: the terminal enzyme in heme biosynthesis. Structure 1997;5:1501–1510.
25. Henikoff S, Henikoff JG. Protein family classification based on searching a database of blocks. Genomics 1994;19:97–107.
26. Murzin AG, Brenner SE, Hubbard T, Chothia C. Scop: a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol 1995;247:536–540.
27. Humphrey WF, Dalke A, Schulten K. VMD: visual molecular dynamics. J Mol Graph 1996;14:33–38.

APPENDIX I: THE TRAINING SET IN THE SUPERFAMILY[†]

1aba-1grx	1aboa-1ckaa	1acf-1pne	1aera-1ddt	1agg-1eit	1agt-1gpt	1ahsa-1bvp1	1ash-2fal
1baba-1mbd	1bbha-2ccya	1bbpa-1hbq	1bbt1-1pov1	1bbt1-2mev1	1bbt2-2stv	1bcfa-1riba	1bco-1itg
1bdo-1ghj	1bera-1lea	1bfma-1tafa	1bgc-1hula	1bgc-1lki	1bgc-1rcb	1bgc-1rmi	1bgc-2gmfa
1bia-1lea	1bmda-1ldm	1bnda-2tgi	1bova-1prtf	1brha-1rgea	1broa-1thta	1btn-1dyna	1btn-1mai
1c53-1fcdc	1c53-451c	1cc5-1cyj	1cdka-1csn	1cds-3ebx	1cem-1clc	1cewi-1mola	1cfb-1ten
1chka-1192	1cis-2seci	1cmba-1myla	1cnsa-1sly	1cpca-1cpcb	1csma-1ecma	1ctf-1ptf	1cus-3tgl
1cyj-1ycc	1cyj-2mtac	1cyj-451c	1cyx-1occb	1din-1whtb	1dlha-1dlhb	1dlhb-1hila	1dlhb-1hsba
1dtr-1hsta	1dtr-1lea	1eaf-3cla	1ecfb-1opr	1egf-1hae	1epaa-1hbq	1epaa-1mup	1erd-2erl
1erv-1gp1a	1erv-1thx	1etd-2hts	1fc1a-1hila	1fim-1otga	1fipa-1rnl	1fjla-1lfb	1fjla-1yrnb
1flp-1itha	1flp-1mbd	1fmb-1hvc	1fmb-2rspa	1fnc-1ndh	1fnf-1ten	1fnf-2hft	1fnf-3hhrb
1frd-1put	1gdha-2dlda	1gky-1ukz	1gln-2ts1	1gpt-1pnh	1gtpa-1gtqa	1hae-3tgf	1hcna-1pdga
1hcnb-1pdga	1hcra-1idy	1hfc-1iae	1hfc-1sat	1hfh-1tpg	1higb-2ilk	1hlb-1mbd	1hleb-7apib
1hmcb-3inkc	1hmf-1hrya	1hnf-3cd4	1hpt-2ovo	1hula-2ilk	1huma-3il8	1hura-1tag	1huw-1lki
1hxpa-1kpaa	1hyha-1ldm	1ica-1gpt	1igd-2ptl	1isdb-1rsy	1itg-1vsd	1kifa-1pbe	1kte-1thx
1l92-1sly	1lbu-1vhh	1lcl-1slta	1lmb3-1neq	1lmb3-1r69	1lpe-2asr	1ltsd-1prtf	1lzr-1sly
1mbd-2gdm	1mbg-1pdnc	1mfa-3cd4	1mfa-7fabh	1mmod-1riba	1mola-1stfi	1mup-1obpa	1myla-1fjla
1ncm-1tlk	1ncm-1vcaa	1ncx-1rro	1ncx-1sra	1ncx-2scpa	1ncx-4icb	1npk-1ris	1ntr-1rnl
1octc-1fjla	1orta-8atca	1osa-1rec	1prs-2bb2	1prtd-1prtc	1pvua-1rvaa	1qrda-1rcf	1r69-1octc
1rcb-3inkc	1rcf-4fx2	1rgea-9rnt	1riba-1xsm	1ris-1urna	1rnl-1troa	1rnl-3chy	1rpa-3pgm
1ryc-1scha	1slta-2ayh	1svpa-1try	1svr-1vil	1sxl-1urna	1tgxa-3ebx	1try-2pkab	1tuc-1gria
1vcaa-1wit	1vhia-2pii	1vin-1vola	1wdcb-1wdcc	1whta-1ysc	1xnb-2ayh	1yppa-2prd	1ytfc-1ytfd
1zaac-2drpa	2bpa2-2stv	2cbh-1lpba	2dri-8abp	2i1b-2ila	2omf-2por	2pcda-2pcdm	

[†]The first entry of each pair is used as a target protein sequence and the second as a template structure.

APPENDIX II: THE TESTING SET IN THE SUPERFAMILY

153l-1sly	1aac-1plc	1aba-1kte	1aera-1ddt	1aky-1ukz	1alo-1frra	1amg-1bpla	1amp-2ctc
1apyb-1pmap	1arb-1hava	1arv-1ryc	1aszb-1lyla	1atla-1hfc	1baba-2hbg	1bbha-1cpq	1bbrh-1try
1bbt2-1pvc2	1bbt3-1pvc3	1bbt3-1smva	1bcfa-1rci	1bdo-1ctm	1bdo-1htp	1bec-1ncm	1bec-8faba
1bera-1rgs	1bfg-2ilb	1bfma-1tafb	1bhs-1cyda	1bmv1-1dhx	1bplb-1smd	1broa-1din	1broa-1ede
1broa-1ysc	1btl-3pte	1btn-1pls	1cd8-1ctn	1cd8-1hila	1cdka-1hcl	1cem-1gai	1ceo-1ghr
1ceo-1xyza	1cewi-1std	1cfb-1ctn	1cid-1mfa	1cnv-1nar	1ctn-2ebn	1cur-2azaa	1cus-1tca
1cwpa-1smva	1cyda-1hdca	1dai-1nipa	1dbqa-1tlfa	1ddt-1mspa	1deka-1gky	1dhr-1hdca	1dlha-1hsbb
1dpe-2olba	1dsba-1thx	1dtr-1ecl	1ecpa-1pbn	1eft-5p21	1enp-1eny	1eny-1hdca	1esfa-1tssa
1esl-1lit	1esl-1rtm1	1exg-1qba	1fca-1fxd	1fim-1otfa	1flp-2fal	1fnc-2pia	1frpa-1imba
1gca-2dri	1gdha-1psda	1gdha-2nada	1gen-1hxn	1glcg-1kay	1gria-1lkka	1gsa-2dln	1gym-1isdb
1hae-1tpg	1hcl-1irk	1hilb-8faba	1hmt-1obpa	1hmy-1vid	1hpi-1isua	1hura-5p21	1igs-1ubsa
1ilr2-2i1b	1imba-1inp	1jkw-1vin	1ldm-2cmd	1led-1scs	1lkka-1mil	1lpe-256ba	1ltsd-1tiid
1lxa-1thja	1mal-2omf	1mbg-1yrna	1mfa-2dblh	1mina-1minb	1mjc-1tssa	1mml-1rtha	1mmob-1mmod
1msc-1qbea	1muca-2mnr	1ncm-1tit	1onc-7rsa	1osa-1wdcb	1paz-1plc	1pea-2dri	1pea-2liv
1plq-2pola	1pmaa-1pmap	1pmaa-1pnkb	1pona-1ponb	1ppn-1thea	1prcl-1prcm	1prn-2por	1prtd-1prtf
1pru-1r69	1ptva-1ytw	1pvc2-1pvc3	1pvc3-2mev1	1rtha-2rn2	1saca-2ayh	1scua-1scub	1smva-2tbva
1sva2-2stv	1thg-1thta	1tlfa-2dri	1vhra-1ytw	1vid-1vpt	1wdcc-1syma	2bpa1-2stv	2fal-3sdha
2sas-2scpa							

APPENDIX III: THE TRAINING SET IN THE FOLD FAMILY

1aboa-1pse	1aca-1dkza	1acf-1pda	1acf-2phy	1acp-1knya	1aep-1rci	1amm-1wkt	1aps-1taq
1axn-1mzm	1axn-1occe	1bbha-1knya	1bbpa-1smpi	1bcfa-1lki	1bgh-1gpc	1bgh-1prtd	1bgw-1lba
1bia-1dcpa	1bia-1umua	1bmfg-1prcl	1bmta-1ecma	1bmta-3chy	1bnca-2rsla	1bpi-1tcp	1bw4-1ptf
1bw4-2eng	1c5a-1hula	1cgme-1occc	1cid-3ssi	1cola-1cpca	1cpca-2hbg	1ctf-1pba	1ctt-2chsa
1dar-1pkp	1dar-2pii	1dcpa-2fua	1deka-1ra9	1dera-1tif	1dhr-1qrda	1dih-1scua	1dkza-1spzz
1dlc-1lpe	1efub-1ecma	1ehs-1jud	1epaa-1sria	1eps-1tig	1etb1-1fnf	1etb1-1thv	1exg-1knb
1fc2c-1whta	1fipa-1ret	1fjla-1scha	1fnf-1kum	1fps-1lis	1frd-1guab	1hdj-1mmog	1hgxa-1opr
1hiwa-1hula	1hnf-8ruci	1hoe-1wkt	1huma-1sap	1hvc-1ytfc	1iso-1svr	1kifa-1phr	1knya-1srsa
1kpaa-1tig	1lcl-1msaa	1led-2ayh	1lepa-1qora	1lgr-1ris	1lpe-1occc	1ltsd-3il8	1lyla-1prtf
1mjc-2prd	1mli-1ris	1mspa-3dpa	1nal3-2mnr	1occa-2brd	1pdo-2liv	1prch-1whi	1psda-1ptf
1psda-1ris	1qbea-1sria	1ris-1spbp	1ris-2chsa	1rvv1-1eria	1sfe-1vsd	1sh1-2bds	1sly-2erl
1smna-1sria	1thm-2liv	1tnra-2stv	1tpg-1fbr	1troa-1utg	1udii-1yua	1xsoa-2mcm	1xxaa-2olba
1ytfb-1ecma	256ba-2mhr	2asr-1occh	2mcm-1fnf				

APPENDIX IV: THE TESTING SET IN THE FOLD FAMILY

1abrb-1bfg	1alo-1ubi	1aps-1dar	1aps-1vih	1axn-1hyp	1bbha-1dkza	1bcfa-1occc	1bfg-1hce
1bgw-1cus	1bgw-1etd	1bhs-1vid	1bia-1pse	1bia-1tnt	1bip-1hyp	1bmfd-1mtnc	1bmfg-1vsga
1bmta-1scub	1bnca-1gara	1brha-1imba	1brsd-1rlr	1cewi-1oaca	1cfb-1slua	1chd-2dri	1cis-1orda
1cpq-1lpe	1cus-8abp	1dai-1deka	1dlc-1glqa	1ecma-1mbd	1eit-1lpba	1erv-1gca	1esl-1prea
1etb1-2pcda	1fkj-1grj	1frvb-3mong	1fwp-1mla	1fwp-1psda	1gdoa-1pmaa	1gln-6inse	1gria-1ihva
1grj-1rpo	1guab-1igd	1guab-1ubi	1hgxa-2liv	1hjra-2yhx	1hoe-3dpa	1hqi-2bopa	1icea-8abp
1iceb-1mla	1ido-1rnl	1igs-1luca	1jud-1nbaa	1lpe-1occa	1mml-1regx	1mola-1qbea	1mola-2bbkh
1muca-1tig	1nal3-1tph2	1obpa-2cae	1occe-1mmog	1occf-3cd4	1orc-1ret	1orda-3chy	1otga-2ctc
1ovaa-1udii	1pda-1stu	1pdga-2tgi	1pft-1tfi	1pne-2blta	1prtf-1snc	1qbea-2cba	1rci-3mdea
1rie-1svb	1rmi-1vnc	1rnl-1tag	1rvv1-1tpt	1rvv1-2dri	1sesa-2spca	1stfi-1udii	1tafa-1tfe
1tig-3dni	1tpg-1pth	1vhia-2bopa	1vsd-2yhx	1wba-2i1b	2ayh-2stv	2spca-1hgeb	