12 Protein Structure Prediction by Protein Threading

Ying Xu, Zhijie Liu, Liming Cai, and Dong Xu

12.1 Introduction

The seminal work of Bowie, Lüthy, and Eisenberg (Bowie et al., 1991) on "the inverse protein folding problem" laid the foundation of protein structure prediction by protein threading. By using simple measures for fitness of different amino acid types to local structural environments defined in terms of solvent accessibility and protein secondary structure, the authors derived a simple and yet profoundly novel approach to assessing if a protein sequence fits well with a given protein structural fold. Their follow-up work (Elofsson et al., 1996; Fischer and Eisenberg, 1996; Fischer et al., 1996a,b) and the work by Jones, Taylor, and Thornton (Jones et al., 1992) on protein fold recognition led to the development of a new brand of powerful tools for protein structure prediction, which we now term "protein threading." These computational tools have played a key role in extending the utility of all the experimentally solved structures by X-ray crystallography and nuclear magnetic resonance (NMR), providing structural models and functional predictions for many of the proteins encoded in the hundreds of genomes that have been sequenced up to now.

What has made protein threading particularly attractive as a protein structure prediction tool is the observation that the number of unique structural folds in nature is a few orders of magnitude smaller than the number of proteins in nature (Finkelstein and Ptitsyn, 1987; Bairoch et al., 2005). Although this is still not a fully resolved issue, both theoretical and statistical studies (Murzin et al., 1995; Brenner et al., 1996; Li et al., 1996, 1998, 2002; Wang, 1996; Orengo et al., 1997; Holm and Sander, 1996a; Zhang and DeLisi, 1998) suggest that the number of unique structural folds in nature ranges from a few hundred to a few thousand. Clearly this is a significantly smaller number than the number of proteins in nature — as we understand now, the number of different living organisms on earth could range from millions to hundreds of millions (May, 1988). Since each organism often has at least thousands of different proteins encoded in its genome, the total number of different proteins in nature is at least in the tens of billions or possibly significantly higher, even without considering protein variants such as alternatively spliced proteins. This disparity suggests an effective paradigm for possibly solving all protein structures through combining experimental and computational approaches, that is, to solve structures of proteins with unique structural folds using the expensive and time-consuming experimental techniques

and computationally model the rest of the proteins using the experimental structures as templates. This is the key strategy currently being employed by the worldwide Structural Genomics efforts (Gaasterland, 1998; Skolnick et al., 2000; Baker and Sali, 2001).

The basic idea of protein threading is to place (align or thread) the amino acids of a query protein sequence, following their sequential order and allowing gaps, into structural positions of a template structure in an optimal way measured by fitness scores outlined above. This procedure will be repeated against a collection of previously solved protein structures for a given query protein. These sequence–structure alignments, i.e., the query sequence against different template structures, will be assessed using statistical or energetic measures for the overall likelihood of the query protein adopting each of the structural folds. The "best" sequence–structure alignment provides a prediction of the backbone atoms of the query protein, based on their placements in the template structure. Currently, protein threading is being widely used in molecular biology and biochemistry labs, often for initial studies of target proteins, as it may quickly provide structural and functional information, which could be used to guide further experimental design and investigation.

As a prediction technique, protein threading has a number of highly challenging computational and modeling problems. These include (a) how to effectively and accurately measure the fitness of a sequence placed in a template structure, (b) how to accurately and efficiently find the best alignment between a query sequence and a template structure based on a given set of fitness measures, (c) how to assess which sequence–structure alignment among the ones against different template structures represents a correct fold recognition and an accurate (backbone) structure prediction, and (d) how to identify which parts of a predicted structure are accurate and which parts are not. As researchers find more effective solutions to these and other challenging problems, we expect that protein threading will play an increasingly significant role in structural and functional studies of proteins.

12.2 Protein Domains, Structural Folds, and Structure Space

As of now, over a million protein sequences have been determined (Bairoch et al., 2005), among which ~30,000 have had their tertiary structures experimentally solved (Dutta and Berman, 2005). Given that there could be at least tens of billions of different proteins in nature as discussed above, one interesting question, particularly relevant to the idea of protein threading, is how many unique protein structures or structural folds these proteins might have adopted.

To answer this question, we need to first look at the basic structural units of proteins, called protein *domains* (Wetlaufer, 1973; Richardson, 1981). Protein domain is extensively discussed in Chapter 4 of this book. Here we describe briefly the concept of a domain from the perspective of threading. A structural domain is a distinct and compact structural unit that could fold independently of other domains.

While many proteins are single-domain proteins, there are proteins with two, three, or even more structural domains (Ekman et al., 2005). Our study shows that in the FSSP (Fold classification based on Structure–Structure alignment of Proteins) nonredundant set (Holm and Sander, 1996b) of the PDB, 67% of the proteins have single domains, 21% have two domains, 7% have three domains, and the remaining 5% have four or more domains (Xu et al., 2000a). This distribution may not necessarily reflect the actual domain distribution for all the proteins in nature as the set of known protein structures in PDB might have overrepresented small proteins due to the relative ease in solving these structures. For eukaryotic organisms, it was estimated that at least two-thirds of their proteins are multidomain proteins (Gerstein and Hegyi, 1998; Gerstein, 1997, 1998; Doolittle, 1995; Apic et al., 2001a,b). This number is somewhat smaller for bacterial and archaeal organisms but still represents a significant percentage of all their proteins. Previously domain partition of multidomain proteins was typically done manually. To keep up with the rate at which protein structures are being solved, there is a clear need for more automated domain-partitioning methods to process the newly solved structures. Currently computer programs are being used for partitioning a protein structure into individual domains. Popular programs for this purpose include DALI (Dietmann and Holm, 2001), DomainParser (Xu et al., 2000a), and PDP (Alexandrov and Shindyalov, 2003).

A protein domain could be part of different protein structures, through combining with other domains. Figure 12.1 shows an example of a domain in different proteins. Both proteins, RNA 3′-terminal phosphate cyclase and glutathione S-transferase, have the thioredoxin fold domain, which has two layers, one with two α-helices and one with four antiparallel β-strands, although some details differ in the two proteins. The parts other than the thioredoxin fold domain in the two proteins have no structural relationship. Since domains are the basic structural units of proteins, current studies on the number of unique structural folds in nature have been carried out on protein domains rather than whole proteins (hereafter, the term "proteins" will refer to single-domain proteins for simplicity of discussion).

To estimate the number of unique folds of proteins, one popular approach is through examining all protein families and the relationships between protein families and unique structural folds. Using the definition of SCOP classification scheme (Murzin et al., 1995; Brenner et al., 1996), a protein *family* represents a group of orthologous proteins (Makarova et al. 1999; Gerlt and Babbitt, 2000; Tatusov et al., 2000; Gelfand et al., 2000). The number of protein families in nature could possibly be estimated through finding orthologous gene groups covering all the genes in the genomes that have been sequenced. One such estimate suggests that there are 23,100 such protein families (Orengo et al., 1994). This number has been used in later studies on estimating the number of unique structural folds in nature. Other studies estimate this number to be in the tens of thousands (Koonin et al., 2002). Whether it is 23,100 or some other number ranging from 10,000 to 50,000, the idea is that all proteins in nature fall into one of these families. The estimation on the number of unique structural folds is obtained through estimating the number of

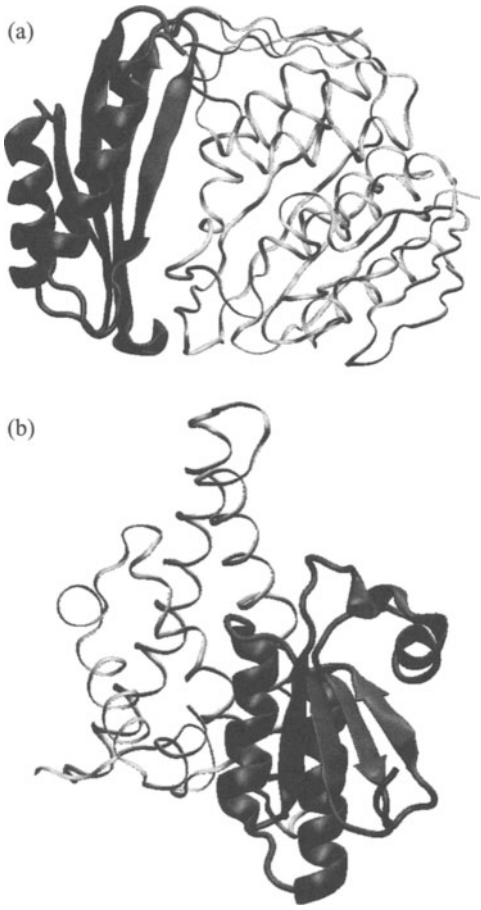

Fig. 12.1 An example of the same domain (thioredoxin fold domain shown in red, thick ribbons) appearing in different protein structures. (a) RNA 3′-terminal phosphate cyclase (PDB code 1qmi, chain A), with the thioredoxin fold domain in residues 185 to 279. (b) Glutathione S-transferase (PDB code 1k0d, chain A), with the thioredoxin fold domain in residues 109 to 200.

families that each structural fold covers and studying the coverage distribution by all the known structural folds.

One of the estimates by Zhang and DeLisi (1998) suggests that the number N of unique structural folds is probably around 700. This estimation is based on the observation that the number of structural folds covering X number of protein families follows a power-law distribution, with X being a variable. In essence it says that a few structural folds each cover many families (e.g., TIM barrel fold covers 31 protein families) while many structural folds each cover only a small number of families; more generally, the number of structural folds decreases as their coverage of protein families increases. Specifically, Zhang and DeLisi proposed a formula

which matches well with the known protein families and structural folds. Let M and N represent the number of families and the number of unique folds in nature, respectively. The probability that a fold covers exactly x families is given by

$$P_x = (1 - M/N)^{x-1} M/N.$$

Let M_s and N_s be the numbers of protein families and unique structural folds currently having solved structures, respectively. Through a simple algebraic transformation, Zhang and DeLisi showed that

$$N \approx \frac{M_s N_s}{M_s - (1 - M_s/M) N_s},$$

which is used to estimate the number of unique structural folds. Using the known numbers of $M_s = 736$ and $N_s = 361$ at the time of the estimation, they estimated that N is roughly about 700, which many researchers will argue to be too small (see following for more discussion).

Similar estimations have been made by other researchers, estimating the size of N ranging from a few hundred to a few thousand (Orengo et al., 1994; Wang, 1996), based on somewhat different assumptions. Coulson and Moult (2002) recently developed a new model for estimating N, based on the work of Zhang and DeLisi. Using the more recent data on the numbers of genes, gene families, and structural folds, they argued that there are two "special" cases that have not been treated well by previous estimation models. Based on their argument, they consider that there are three classes of structural folds, which are termed *unifolds, mesofolds*, and *superfolds*. Unifolds represent structural folds that each covers only one family of proteins, superfolds represent structural folds, each of which covers many structural folds, and mesofolds represent structural folds in between. For example, TIM barrel covers 31 families, while many unifolds exist in SCOP. Based on their observation, they argued that previous models such as Zhang and DeLisi did not fit well with the data of unifolds and superfolds. So a new piecewise model was then developed which treats unifolds, mesofolds, and superfolds, separately.

Using this new model, Coulson and Moult (2002) estimated that less than 20% of the protein families belong to unifolds, while 20% of the families belong to a few dozen superfolds and the rest of the protein families belong to mesofolds. Considering that the estimated number of protein families ranges from 10,000 to 50,000 (or 23,100 as one of the popular estimates suggests), we can infer that the number of unifolds ranges from 2,000 to 10,000. The number of mesofolds could be estimated using the Zhang and DeLisi model, based on the sizes of M_s and N_s, after excluding the unifolds and superfolds. Hence, Coulson and Moult concluded that the most probable size for the number of mesofolds is about 400. The number of superfolds is believed to be very small, possibly in the range of low dozens. *Overall this model suggests that over 80% of the protein families fold into a little over 400*

structural folds, the majority of which are already known, while the rest of the protein families each belongs to a unique unifold.

The implication of this estimation is that about 80% of the protein families are amenable for structural modeling using protein threading techniques, assuming that at least one protein in each of the meso- and superfolds has its structure solved. If experimental facilities for structure solution will strategically select their solution targets to maximally cover all the meso- and superfolds, we could expect that at least 80% of the protein families will be structurally modelable in the near future. This is exactly the strategy that the National Institute of Health (NIH) is using in its Protein Structure Initiative (*http://www.nigms.nih.gov/psi/*). For the remaining 20% of protein families, it might take some time to have at least one solved structure in each of the unifolds. Hence, the threading technique will be less applicable for this class of proteins, at least in the near future.

There are a number of popular schemes and associated databases for classification of proteins into structural folds, including SCOP (Murzin et al., 1995), CATH (Orengo et al., 1997), and FSSP (Holm and Sander, 1996b). These classification schemes classify all solved protein structures into different structural folds and subclasses of structural folds. The classification of protein structures is essentially achieved through grouping protein structures into clusters of similar structures, which can be done computationally through structure–structure alignments (Holm and Sander, 1996a).

SCOP (Murzin et al., 1995; Brenner et al., 1996; Andreeva et al., 2004) groups all protein structures essentially into a three-level classification tree. At the top level, SCOP (SCOP1.65) currently consists of about 800 structural folds, each of which is further divided into superfamilies and then into families. While a family represents a group of orthologous proteins, a superfamily represents a group of homologous proteins, possibly made of multiple families. Currently SCOP consists of about 1300 superfamilies and about 2400 families. Among the 800 structural folds, 489 have only one family each, which might represent unifolds; and 9 cover a large number of families each, which are considered as superfolds by Coulson and Moult. One thing worth noting is that among the 800 SCOP folds, only 36 represent membrane proteins. This is a reflection of the fact that only 2% of all the solved protein structures are membrane proteins (*http://blanco.biomol.uci.edu/Membrane_Proteins_xtal.html*). This suggests that threading is generally not applicable to structure prediction of membrane proteins, at the present time.

SCOP's hierarchical classification of structural folds provides a convenient tool for applications of threading methods, as query proteins falling into a SCOP protein family are generally expected to have accurate structure predictions, while proteins with structural homologues in a SCOP superfamily will have a good chance to have the correct structural folds identified and some portions of their backbone structures predicted accurately. In general, it still represents a challenge to correctly identify the structural fold by a threading method if a query protein only has a structural analogue (i.e., similar structure but not homologous) in SCOP.

12.3 Fitting a Protein Sequence onto a Protein Structure

The realization that protein structures are clustered into structural folds in the structure space and the number of such clusters is possibly quite small has led to a new way of predicting protein structures in a more efficient and effective manner. The general belief is that different proteins fold into similar 3D shapes because at some level, they share similar interaction patterns among their residues and between the residues and the environments. It has been shown that these interaction patterns could possibly be captured using simple statistics-based energy models as exemplified by the earlier work of Eisenberg and colleagues (Bowie et al., 1991; Fischer et al., 1996a,b; Fischer and Eisenberg, 1996), the work of Sippl and colleagues (Sippl, 1990), and others (Jones et al., 1992; Rost et al., 1997). These simple statistics-based energy functions have been used, for many cases, to distinguish the correct structural folds from the incorrect ones and to distinguish the correct placements of the residues in a query protein into the structural positions of a correct structural template from the incorrect ones. Placing the (backbone atoms of) residues of a query protein into the correct structural positions in a correct structural fold gives a prediction of the backbone structure of the query protein. To accomplish this, one would need two capabilities: (a) an energy function whose global minimum will correspond to the correct placement of residues into the correct structural template, and (b) a computational algorithm that can find the global minimum of the given energy function. We explain the basic idea of developing such energy functions in this section and leave the algorithmic issues to the next one.

In their earlier work (Bowie et al., 1991; Fischer et al., 1996a,b, Fischer and Eisenberg, 1996), Eisenberg and colleagues demonstrated that simple residue-based, instead of atom-based, energy functions could provide substantial discriminating power in separating good from poor placements of individual residue types into different structural environments, justifying the usage of residue-based energy functions. In their work, structural environments are simply defined in terms of two parameters, solvent accessibility *sol* and secondary structure *ss*. Specifically the quantity *sol* of solvent accessibility is discretized into a number of intervals, say 30–40% exposed to the solvent. A secondary structure could be a helix, a beta-strand, or a loop, or it could be defined in terms of more refined categories of secondary structures, say including different types of turns. Then a structural environment for each residue in a template structure could be defined using (*sol, s*), say (0–10% exposed, alpha-helix). Statistics could be collected from a collection of solved protein structures about how frequent a particular type of amino acid appears in a particular structural environment as we just defined. This can be done by going through all protein structures under consideration to count the number of occurrences of each amino acid type in each encountered structural environment. If we consider, say, three levels of solvent accessibility, {exposed, intermediately exposed, buried}, and three types of secondary structures, we will have nine types of structural environment. Under this assumption, the result of counting the numbers of occurrences

above will be a 20 by 9 table, with each of the 20 rows representing an amino acid type and each of the 9 columns representing a structural environment.

Based on the collected statistics, we can build a simple *preference* model to measure how preferred a particular amino acid type is to a particular structural environment. This can be done using the following measure:

$$-\ln(O_{i,j}/E_{i,j}),$$

where $O_{i,j}$ represents the observed frequency of amino acid type i in structural environment j and $E_{i,j}$ represents the expected frequency of amino acid type i in structural environment j. In the work of Eisenberg and colleagues, $E_{i,j}$ is estimated using the frequency of amino acid type i in all proteins under consideration. Hence, if an amino acid type i has a higher frequency in a particular structural environment j than its frequency overall, it will be assigned a negative score $-\ln(O_{i,j}/E_{i,j})$; otherwise it will get a positive score or zero (when $O_{i,j} = E_{i,j}$). The higher $O_{i,j}$ is compared to $E_{i,j}$, the more negative the corresponding energy is. A popular name for this type of energy function is *singleton energy*. By performing statistical analysis on a database, one can get the scoring function using the above formulation for the 20 amino acid types appearing in the nine structural environments. Table 12.1 shows such a scoring function, as described in Xu et al. (1998).

Table 12.1 The scoring function for the 20 amino acid types in three secondary structure types with three solvent accessibility levels. The three solvent accessibility types are *buried*, *intermediate*, and *exposed*, from the left column to the right for each secondary structure type.

Residue	Helix			Sheet			Loop		
A	−0.741	−0.007	−0.181	−0.430	0.642	1.367	0.223	0.235	0.070
R	1.010	−0.687	−0.344	0.995	−0.479	0.340	1.406	−0.199	−0.190
N	0.840	0.221	0.042	0.483	0.344	0.743	0.656	−0.247	−0.795
D	1.113	0.243	−0.376	0.907	0.250	0.853	0.850	− 0.228	−0.783
C	−0.389	0.539	2.179	−1.208	0.492	2.651	−0.489	0.119	1.249
Q	0.587	−0.428	−0.573	0.618	0.058	0.221	1.220	0.067	−0.449
E	1.074	−0.442	−0.842	1.168	−0.029	0.282	1.773	0.218	−0.527
G	0.317	1.234	1.010	−0.161	0.714	1.167	−0.116	0.040	−0.849
H	0.335	−0.217	0.199	−0.127	−0.265	0.848	0.519	−0.387	− 0.117
I	−0.686	−0.001	1.198	1.316	−0.156	1.295	0.182	0.408	0.996
L	−0.902	−0.178	0.987	0.930	0.256	1.672	0.009	0.229	0.977
K	2.021	−0.212	0.743	1.818	− 0.209	−0.008	2.611	0.107	0.696
M	−0.858	0.378	0.872	−0.825	0.230	1.215	0.208	0.242	0.692
F	−0.585	0.089	1.146	−1.040	−0.168	1.085	−0.060	0.079	0.911
P	1.126	0.698	0.723	0.845	0.704	1.287	−0.041	−0.621	−0.646
S	0.328	0.454	0.111	−0.079	0.088	0.445	0.373	−0.073	−0.612
T	0.302	0.247	0.341	− 0.211	−0.419	0.126	0.350	−0.014	−0.377
W	−0.468	−0.270	1.493	−0.816	−0.307	1.843	− 0.082	0.041	0.797
Y	−0.149	−0.285	0.991	−0.690	−0.704	0.919	0.461	− 0.292	0.723
V	−0.491	0.251	1.053	−1.352	−0.324	0.983	0.196	0.382	0.737

When building such statistics-based energy functions, one needs to be careful in selecting the data set for statistics collection. For example, some proteins in SCOP (or in PDB) have more homologous structures than the others, which could possibly lead to biased statistics. To remove this type of statistic bias in our data set, one needs to remove homologous structures in the data set for statistics collection. There are a number of databases for this purpose, such as the nonredundant sequence representatives in FSSP, PDB-select (Hobohm et al., 1992), and PISCES (Wang and Dunbrack, 2003), where no two proteins have higher than a certain level of sequence similarity.

Another statistics-based energy function widely used in threading programs is often called *pairwise interaction energy*. It measures the preference of having two particular types of amino acids that are spatially close. One particular form of such an energy function was developed by Sippl (1990). The basic idea of this energy function is to compare the observed frequency of a pair of amino acids within a certain distance in solved protein structures with the expected frequency of this pair of amino acid types in a protein structure. The basic idea of such an energy function comes from statistical mechanics where the probability P_{ij} of having a pairwise interaction between residues i and j has the Boltzmann correlation to its energy g_{ij}^* (Gibbs free energy), defined as

$$P_{ij}^g = \exp\left(\frac{-g_{ij}^*}{kT}\right) \Big/ Z,$$

where k is the Boltzmann constant, T is the temperature, and Z is a partition function. When using a residue-averaged state as a reference state $\overline{g}(\overline{P})$, a knowledge-based potential can be derived using

$$g_{ij} = g_{ij}^* - \overline{g} = -kT \ln\left(\frac{P_{ij}^g}{\overline{P}}\right).$$

Specifically, if $N_O(i, j)$ and $N_E(i, j)$ represent, the observed and expected numbers of amino acid types i and$j$ within a certain distance, respectively, we can use the following to measure the preference of having these two types of amino acids close to each other:

$$-\ln\left(N_O(i, j)/N_E(i, j)\right).$$

While $N_O(i, j)$ can be collected by going through the protein structures in the sample set, an accurate estimation of $N_E(i, j)$ represents a challenge. There have been a number of proposed models for estimating this quantity. Among these models, the simplest one is the "independent reference state" model (Xu et al., 1998), in which $N_E(i, j)$ is estimated as follows:

$$N_E(i, j) = \frac{\sum_k N_O(i, k) \sum_k N_O(j, k)}{\sum_{i,j} N_O(i, j)}.$$

Table 12.2 Pairwise scoring functions between 20 amino acid types

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	−14																			
R	27	−2																		
N	10	−9	−43																	
D	22	−62	−42	2																
C	33	7	11	28	−192															
Q	3	−6	−20	7	19	−11														
E	12	−56	−14	14	12	1	7													
G	1	−8	−10	−27	9	−7	−3	−29												
H	6	−26	6	−45	−19	27	37	− 7	−45											
I	−11	11	35	32	15	24	29	18	29	−33										
L	−18	26	36	37	24	2	25	24	20	−16	−28									
K	12	31	−20	−56	25	−18	−67	10	5	19	18	12								
M	−7	30	31	21	5	3	14	1	−1	−1	−11	30	−49							
F	−6	6	20	28	3	7	23	11	16	−10	−20	−2	−27	−21						
P	17	−3	−21	−3	11	−8	−10	−7	−6	37	22	−5	3	−2	−21					
S	17	− 8	−22	−30	1	−16	−21	−19	−13	21	27	− 6	19	11	−16	−18				
T	6	6	−23	−20	37	−15	−21	−7	−24	11	22	−2	16	28	−10	−22	−21			
W	5	−15	−2	10	5	−1	16	−7	−21	−2	8	3	−1	3	−43	13	9	−2		
Y	5	−13	5	27	6	−9	27	6	3	−16	−9	−31	−17	−1	−8	10	16	−10	−1	
V	−11	17	30	43	20	18	23	20	20	−23	−22	27	−5	−4	5	27	7	10	−11	−32

Table 12.2 shows a scoring function for the preference between 20 types of amino acids using the above formulation (Xu et al., 1998).

There are more sophisticated models for defining the reference state, one of which is the *uniform distribution model* (Sippl, 1990; DeWitte and Shakhnovich, 1996; Lu and Skolnick, 2001; Samudrala and Moult, 1998), as discussed later. These more complex models take more factors into consideration in building the reference state, hence making the energy models more likely to be accurate.

In addition to using physics- or statistics-based energy functions, researchers have incorporated evolutionary information into energy model building. One of the earlier major improvements in energy function modeling is the incorporation of sequence profile information (Panchenko et al., 2000; Zhou and Zhou, 2005) derived from homologous proteins into the energy models outlined above. It was noticed that when using a sequence profile of a protein family (or superfamily) rather than a single (query) protein sequence, threading accuracy could be significantly improved (Panchenko et al., 2000; Zhou and Zhou, 2005). The very basic idea of this generalized approach is that rather than asking the question "will protein sequence A fit well with structural fold B?" we ask the more general question "will the whole family of protein A fit well with structural fold B?" Clearly if done properly, this approach could iron out some of the spurious predictions, caused by the appearance of specific individual sequences. Now a threading problem becomes a fitting problem between a sequence profile and a structural fold. A sequence profile is defined in terms of a multiple sequence alignment of the members of the query protein's family (or superfamily), with each element being the frequency distribution of the 20 amino acid types in this aligned position rather than a specific amino acid. To generalize the aforementioned energy functions to take into account the profile information, we can simply use the relative frequency of each amino acid in the position-dependent distribution as a weight factor when calculating the energy values for each amino acid

or amino acid pairs, and then sum over all possible amino acids or amino acid pairs. Specifically, let P_i be the relative frequency of amino acid type i in a particular aligned position with $\sum_i P_i = 1.0$. Then the Eisenberg type of energy can be calculated as

$$-\sum_i P_i \ln(O_{i,j}/E_{i,j}).$$

Similarly, pairwise interaction energy could be generalized as follows:

$$-\sum_{i,j} P_i P_j \ln(N_O(i,j)/N_E(i,j)).$$

Other types of energy functions have also been used in existing threading programs, including fitness scores between specific amino acids and the secondary structures in the template structure and threading alignment gap penalties. Typically these energy scores are combined using a simple weighted sum while often the scaling factors are empirically determined, based on some training data.

It has been observed that distance-dependent pairwise interaction energy could provide more accurate threading results than distance-independent models as outlined above. A distance-dependent energy could be estimated as follows:

$$\overline{u}_{(i,j,r)} = -\ln \frac{N_O(i,j,r)}{\mathrm{N}_E(i,j,r)},$$

where r is the distance between residues i and j (possibly measured between their C-beta atoms), $N_O(i,j,r)$ is the observed number of pairs of residues (i,j) within a distance bin from $r - \Delta r/2$ to $r + \Delta r/2$ in a database of folded structures for some bin width Δr, and $N_E(i,j,r)$ is the expected number of pairs (i,j) within the same distance bin. The challenging issue in accurately estimating the interacting energy $\overline{u}_{(i,j,r)}$ is how to estimate $N_E(i,j,r)$. Under the assumption that we are dealing with an ideal infinite liquid-state system within a volume V and residues are distributed uniformly (Sippl, 1990; DeWitte and Shakhnovich, 1996; Lu and Skolnick, 2001; Samudrala and Moult, 1998), $\mathrm{N}_E(i,j,r)$ can be estimated using

$$N_E(i,j,r) = N_i \cdot N_j \cdot \left(4\pi r^2 \Delta r\right)/V,$$

where N_i and N_j are the numbers of amino acid types i and j in the protein database, respectively. Researchers have realized that this model needs to be corrected when dealing with finite systems like a protein structure, to make the model more accurate when used in threading programs. Two particular corrections are made in the DFIRE (distance-scaled ideal gas reference state) energy model (Zhou and Zhou, 2002), a popular energy function for threading. In the first correction, DFIRE used r^α instead of r^2, considering that the number of interaction pairs in a finite system could not actually reach the level of r^2 as in an infinite system, where $\alpha < 2$ is

determined through minimizing the distribution fluctuation of interaction distance on a set of training data. In the second correction, DFIRE assumes that only short-range interactions need to be considered. That is, interaction energy becomes zero when the distance between the interacting pairs is beyond a cutoff distance r_{cut}. After these corrections, the interaction energy could be estimated using the following formula:

$$\bar{u}_{(i,j,r)} = \begin{cases} -\eta \ln \dfrac{N_O(i,j,r)}{\left(\dfrac{r}{r_{cut}}\right)^{\alpha} \dfrac{\Delta r}{\Delta r_{cut}} N_O(i,j,r_{cut})}, & r \le r_{cut} \\ 0, & r > r_{cut} \end{cases}$$

where constant η is related to the system temperature and can be determined empirically.

These simple energy models have played key roles in making threading programs as popular and as useful as we have seen today. While they have been used to help to solve many structure prediction problems, the limitations of these simple models have also become quite clear as we have seen from the recent CASP prediction results—the improvement in prediction accuracy has been only incremental in the past few CASPs (Kinch et al., 2003). One of the key reasons for the incremental improvement comes from the crudeness of the threading energy functions. Currently, the algorithmic techniques for protein threading have advanced to a stage that should be able to handle more sophisticated energy models in the threading framework, which could lead to more accurate predicted structures. We can expect that more detailed and more physics-based energy functions will emerge in the near future as the field is clearly in need of more accurate energy models for protein threading.

12.4 Calculating Optimal Sequence–Structure Alignments

The general form of threading energy function could be written as follows:

$$\alpha E_s + \beta E_p + E_{gap},$$

where E_s measures the overall fitness of putting individual residue types into specific structural environments, E_p measures the total interaction energy among pairs within the cutoff distance, and E_{gap} represents the total penalty for the gaps in a sequence–structure alignment. The scaling factors α and β are typically determined empirically through optimizing the performance of a threading program on a representative set of protein pairs. With the optimized α and β values, the goal of threading is to find an alignment (or placement) between a query protein sequence and a template structure that optimizes the energy function.

In a sense, protein threading is like sequence–sequence alignment as it finds an alignment between a sequence of amino acids and a sequence of structural positions

in a 3D structure. What makes threading more difficult than a sequence alignment problem is the pairwise interaction energy term E_p. For a sequence alignment problem, a simple dynamic programming can guarantee to find the global optimal alignment between two sequences as the problem formulation follows the *principle of optimality* (Brassard and Bratley, 1996). This type of simple dynamic programming algorithm does not work for a threading problem as the global optimal threading alignment could not be easily reduced to a small number of optimal threading alignments for the partial problems as in a sequence alignment problem. Intuitively, a simple dynamic programming approach, like the one used for sequence alignment that goes from the starts of the sequences to their ends and extends partial optimal alignments to include more elements at each step, will not work for protein threading as at each current point, we do not know what residues will be available to be assigned to future structural positions, which might have interactions with the previously aligned positions. Such interactions will have a global impact on a sequence–structure alignment. It is such a global nature of the problem that makes the threading problem more challenging from the algorithmic point of view.

There have been a number of studies attempting to understand the "intrinsic" difficulty, or computational complexity, of the threading problem. Under the assumption that all pairwise interactions need to be considered, the threading problem was proved to be an NP-hard problem by a number of authors (Lathrop, 1994; Calland, 2003). While these mathematical proofs provide some evidence that the problem is computationally difficult, they might not be particularly relevant to the true difficulty of a threading problem as previous studies have shown that pairwise interactions beyond certain cutoff distances (e.g., 10–12 Å between C-beta atoms) do not contribute to fold recognition and threading alignment (Lund et al., 1997; Melo and Feytmans, 1997; Xu et al., 1998; Zhang and Skolnick, 2004), and hence need not be considered. As of now, it remains an open problem regarding whether the threading problem, using a distance cutoff for pairwise interactions, is polynomial-time solvable. The challenging issue in studying the "intrinsic" computational complexity of a threading problem with a cutoff for pairwise interactions is that we do not have a good and realistic characterization for the overall interaction patterns for such a threading problem.

Because of the algorithmic challenges, earlier threading programs have employed various heuristic strategies for solving the optimal sequence–structure alignment problem. One particular strategy is called "frozen approximation" (Westhead et al., 1995). The basic idea is that it uses a dynamic programming approach to find a sequence–structure alignment and uses an approximation scheme to calculate the interaction energy. When the algorithm assigns an amino acid to a specific structural position from the beginning to the end of the query sequence during the dynamic programming procedure, it calculates the relevant interaction energy using the amino acids in the template structure rather than assigned amino acids from the query protein, for all the unassigned structural positions up to the current point of the dynamic programming procedure, within a certain cutoff distance. Intuitively the

algorithm should work to some degree in capturing some of the interaction "patterns" encoded in the query protein sequence as some of the position-equivalent residues between the native structure and the native-like template structure should have similar physicochemical properties, suggesting the validity of the frozen approximation scheme. Practical applications have also confirmed that the frozen approximation, while not guaranteeing global optimal threading alignment, does have an advantage over threading programs that do not consider pairwise interactions (Westhead et al., 1995; Skolnick and Kihara, 2001; Zhang et al., 1997).

A number of rigorous threading algorithms have been developed that guarantee to find the global optimal threading alignments, measured in terms of energy functions that consider pairwise interactions. These include a divide-and-conquer algorithm employed in the PROSPECT threading program (Xu and Xu, 2000) and an integer programming algorithm used in the RAPTOR program (Xu et al., 2003a,b). It was convincingly demonstrated, through applications of these programs at the CASP contests (Xu et al., 2001; Xu and Li, 2003), that threading programs with guaranteed global optimality do have an advantage over programs without this property. One particular advantage of such programs is that they can be used to rigorously benchmark a proposed energy function. When using programs without such a guarantee to assess an energy function, it will be difficult to decide whether it is the energy function or the lack of rigor in the threading algorithm that has resulted in a subpar performance by a particular energy function. The following provides some detailed information about three types of threading algorithms.

12.4.1 PROSPECT

The basic idea of the divide-and-conquer algorithm in PROSPECT (Xu and Xu, 2000) can be outlined as follows. The algorithm first divides the query protein sequence into two subsequences and also divides the template structure into two substructures by cutting at one of its loop regions. Then it tries to find the globally optimal threading alignments between the first subsequence/substructure pair and between the second subsequence/substructure pair, respectively. When calculating pairwise interaction energy for each "half" of the sequence–structure alignment problem, we might need information about which amino acids are assigned to which structural positions in the other "half" of the problem. To facilitate this calculation, the algorithm uses a simple data structure for each structural position L in the current substructure, that keeps a list of structural positions in the other substructure that are close enough to L to have interactions with the amino acid to be assigned to it. Figure 12.2 depicts schematically the situation where each of the two substructures has a number of structural positions, which are close enough to structural positions in the other substructure so that their alignments with amino acids need to be considered when calculating the interaction energy in the other substructure. These structural positions can be considered as extended parts of the other substructure (shown as extended arms for each substructure in Fig. 12.2). The difference between these extended parts and the original positions of a substructure is that when doing alignment

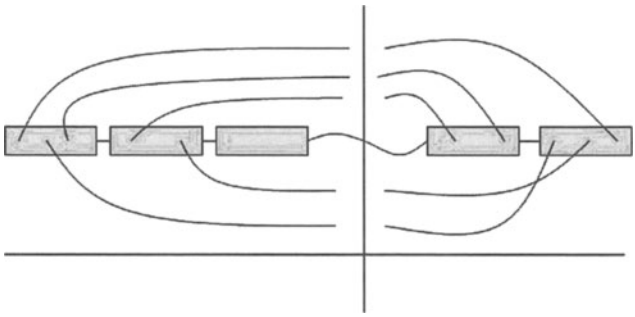

Fig. 12.2 Schematic illustration of the divide-and-conquer algorithm of PROSPECT. The blocks represent core secondary structure elements, and curved lines indicate interactions between linked secondary structures.

between the substructure and the corresponding subsequence, we do not have any knowledge about which amino acids are assigned to these positions in the other substructure. Hence, the optimal threading alignment for each substructure/subsequence pair depends on the optimal threading alignment for other substructure/subsequence pair. This codependence relationship makes the problem challenging.

In PROSPECT, this problem is overcome using the following strategy: consider all possible combinations of amino acids possibly assigned to these extended positions (in the other "half" of the problem); and then solve an optimal threading alignment problem for each substructure/subsequence pair under each possible combination of amino acid assignments to these extended structural positions. Assume that we can solve the optimal threading problem for each pair of (extended) substructure/subsequence and for each combination of such an assignment. Then it can be checked that the global optimal threading alignment for the whole structure and sequence must be the union of two optimal threading alignments for the two extended subproblems, under one specific combination of the amino acid assignment to the extended part of each subproblem. This realization lays the foundation for the divide-and-conquer algorithm of PROSPECT as it allows reducing a whole threading problem to two smaller threading problems. If we can solve the smaller threading problems, the whole threading problem can be solved by simply going through all combinations of the amino acid assignments to the extended parts for each subproblem to find the one that gives the overall best combined score. During the "conquer" step, it needs to make sure that the two optimal solutions, to be combined, to the subproblems is consistent. To solve a smaller threading problem, we can apply the same divide-and-conquer strategy to reduce it to even smaller problems. This procedure can continue until the size of the problem is small enough that it can be solved using a brute force exhaustive search strategy.

The trick is how to make this algorithm run efficiently. Note that if not done carefully, the number of possible combinations of assignments needed to be considered could be very large. In PROSPECT, (sub)structures are divided in such a way which minimizes such a number of combinations, through cutting at the "weakest" link with

the least interactions between two substructures (Xu et al., 1999). The overall computational complexity of the divide-and-conquer algorithm is dominantly determined by the thickest link among all weakest links throughout the bipartitioning of a protein structure into a series of small structures during the whole divide-and-conquer scheme. Intriguingly, we found that this thickest link is generally a small number for the vast majority of the solved protein structures (Xu et al., 1999), making the actual computing time of PROSPECT threading practically acceptable. This observation has also raised an interesting question: "Is there something special about the topology of protein structures, which could be further and more rigorously exploited for efficient threading algorithm development?"

12.4.2 RAPTOR

RAPTOR uses a more general framework to rigorously solve the threading problem (Xu et al., 2003a,b) than PROSPECT. It formulates a threading problem as a linear integer programming (LIP) problem. It uses an integer variable ("0" or "1") to represent if a particular residue in the query protein sequence is assigned to a particular structural position. Then the set of all feasible solutions to a threading problem could be defined in terms of a set of equalities or inequalities (called *constraints*), each of which is defined in terms of the above and other integer variables. The global optimal threading problem is then defined to find a feasible threading alignment that optimizes a given energy function. Generally a linear integer programming problem requires an exponential computing time to find an optimal solution, and hence intractable for large-size problems. Branch-and-bound represents a popular technique for solving linear integer programming problems. Typically, an integer programming problem is first relaxed to a linear programming problem, i.e., variables could take real values as possible solutions. There are efficient algorithms for solving linear programming problems as they are polynomial-time solvable (Papadimitriou and Christos, 1998). If by chance the solution to the relaxed linear programming problem is all integral, a solution to the original linear integer programming is found. Otherwise the linear solution will be used to constrain the search space, through fixing one variable to "0" or "1" and then the algorithm iterates this process until all solutions have integral values. An interesting observation made is that for the vast majority of threading problems, this relaxation procedure stops after a few iterations, solving the threading problem efficiently and also indicating that threading problems seem to have a special structure in terms of the integer programming formulation. Such special characteristics could possibly be utilized for developing more efficient threading algorithms, using more specialized algorithmic techniques.

12.4.3 Tree-Decomposition-Based Threading Algorithm

One particularly interesting technique which is being actively investigated by a number of researchers is based on the idea of tree decomposition of an interaction

graph representing possible alignments between a query sequence and a template structure (Song et al., 2005; Xu et al., 2005). In a sense this type of technique is a generalization of the divide-and-conquer outlined above. Tree decomposition technique has been widely used for various graph-related optimization problems, for example finding the maximum independent set and dominating set (Arnborg and Proskurowski, 1989). We now provide a detailed description of one such algorithm for solving the protein threading problem.

Using a tree decomposition algorithm, both the template structure and the query sequence are represented as graphs; vertices denote core secondary structures (or simply cores) and edges represent interactions between cores (two cores are considered to be in interaction if their shortest distance is within a predefined cutoff distance). A sequence–structure alignment problem essentially corresponds to finding an isomorphic mapping from the structure graph to a subgraph of the sequence graph. The efficiency of the alignment hinges on the *tree width* of the structure graph.

Intuitively, the tree width of a graph measures how much the graph is "treelike." A graph can be represented as a "tree having thick trunks," where the "trunk thickness" is quantified by the tree width of the graph. This technique of "treelike" representation for graphs is called *tree decomposition.* In a tree decomposition of a graph, vertices of the graph are grouped into possibly overlapping subsets, each of which is associated with a node in the tree. The maximum size of such a subset corresponds to the tree width of the tree decomposition. Given a tree decomposition of a structure graph with tree width t, a dynamic programming algorithm can be employed to find the optimal sequence–structure alignment in time $O(k^t N^2)$, for some small integer parameter k and N being the number of amino acids in the template structure. The alignment algorithm is very efficient since the tree width for such structure graphs is small in general (by the nature of protein structures). For example, among 3890 protein tertiary structure templates compiled using PISCES (Wang and Dunbrack, 2003) only 0.8% of them have tree width $t > 10$ and 92% have $t < 6$, when using a 7.5-Å $C_\beta - C_\beta$ distance cutoff for defining pairwise interactions [see Fig. 12.5(a)]. We now provide the details of a tree-decomposition-based threading algorithm.

12.4.3.1 Graph Representation

A sequence–structure alignment can be formulated as a *generalized* subgraph isomorphism optimization problem, for which both the template structure and the query sequence are represented as *mixed* graphs that contain both directed and undirected edges. We use $V(G)$, $E(G)$, and $A(G)$ to denote the vertex set, the undirected edge set, and the directed edge (arc) set of a mixed graph G, respectively.

The graph H for the template structure is constructed as follows: each vertex in $V(H)$ represents a core, each undirected edge in $E(H)$ represents the interaction between two cores, and each directed edge (arc) in $A(H)$ represents the loop between two consecutive cores (from the N-terminal to the C-terminal). For technical

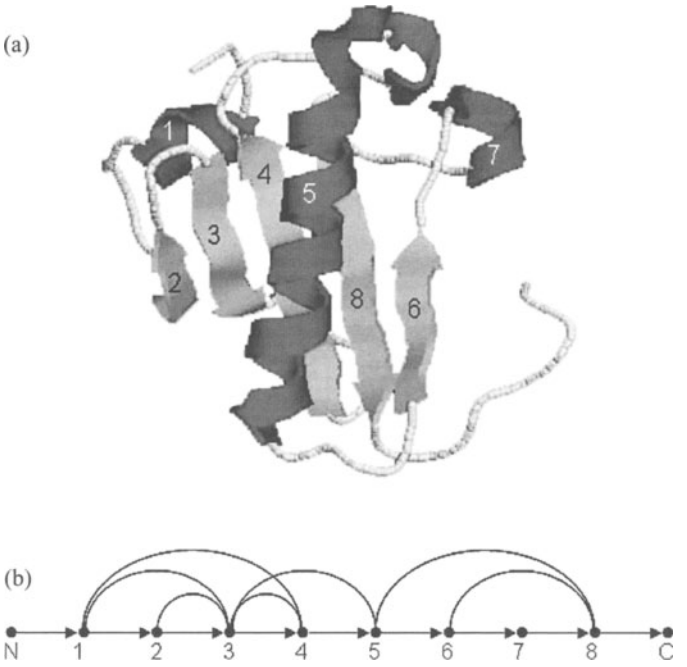

Fig. 12.3 (a) Folded chain B of protein kinase C (PDB-ID 1AV) protein with eight core secondary structures; (b) its corresponding structure graph.

convenience, both N- and C-terminals are presented as vertices. Figure 12.3 gives a protein tertiary structure and its corresponding structure graph representation.

A query sequence is preprocessed so that for each core in the template, all substrings (called *candidates*) of the query sequence that align well with the core are identified (Xu et al., 2000). By representing each candidate as a vertex, a query sequence can also be represented as a mixed graph. That is, each edge in $E(G)$ connects a pair of candidates that may possibly interact but do not overlap in the sequence, and each arc in $A(G)$ connects two candidates (from the N-terminal to the C-terminal) that do not overlap. As in a structure graph, both N- and C-terminals are represented as vertices in the sequence graph. Figure 12.4 illustrates the sequence graph with a simple example.

The relationship between a core v in the template and its candidates in the query sequence can be constrained using a mapping function M such that $M(v)$ contains all possible candidates of v. The less restricted M is, the more accurate the alignment is expected to be and the more time it may take to compute. Xu et al. (1998) used a similar approach in their divide-and-conquer threading algorithm, which can find all suitable candidates for each core. The maximum size $k = |M(v)|$ over all cores v is called the map width of M, an important parameter for the alignment algorithm.

Now a sequence–structure alignment problem can be formulated as a problem of finding an isomorphism mapping f between the structure graph and a subgraph of

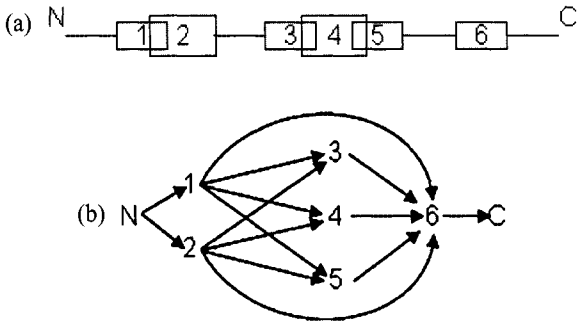

Fig. 12.4 (a) Five alignment candidates of cores found after a preprocessing of the sequence, among which overlapping candidates are <1,2>, <3,4>, and <4,5>, respectively; (b) the corresponding sequence graph (edges are not shown, and arcs incident on N- and C-terminals are not shown).

the sequence graph G such that the following sum of the alignment energy functions

$$\sum_{u \in V(H)} E_{core}(u, f(u)) + \sum_{(u,v) \in E(V)} E_{pair}((u, v), (f(u), f(v))) + \sum_{<u,v> \in} E_{loop}(< u, v >, < f(u), f(v) >)$$

achieves the minimum, where E_{core} is the alignment energy score (singleton type of energy) between a core u in the template and its candidate $f(u)$ in the sequence, E_{pair} represents interaction energy (pairwise interaction energy) between residues $(f(u), f(v))$ assigned to cores (u, v), and E_{loop} is the alignment score between the loop $< u, v >$ in the template and the corresponding $<f(u), f(v)>$ in the sequence.

12.4.3.2 Tree Decomposition of Structure Graph

For a structure graph H, its tree decomposition (T, X) is defined by a tree topology T, which is a binary tree, and a collection $X = \{X_1, X_2, \ldots, X_m\}$, where each X_i is a subset of $V(H)$,called a *bag* for tree node i in T. Any tree decomposition (T, X) of a graph H should satisfy the following conditions: (a) $\bigcup X_i = V(H)$; (b) for any u, v forming an (un) directed edge in the graph, u and v are contained in the same bag; and (c) the tree nodes whose bags contain the same vertex should form a connected subtree of T. Intuitively, a tree decomposition groups locally connected vertices together into bags, and the set of vertices in the bag for each tree node can actually separate the graph into two disconnected components. The tree width of a tree decomposition is the maximum size of the bag associated with a tree node minus one. The tree width of the graph is the minimum tree width over all possible tree decompositions for the graph (Robertson and Seymour, 1986; Bodlaender, 1996).

There are algorithms of time $O(c^t n)$ that can find an optimal tree decomposition for a graph of n vertices, provided that the graph has a tree width t. However, the constant c may be too large for the algorithm to be practically useful. For structure

graphs formulated in our current context, we found that fast heuristic algorithms may produce tree decompositions with a tree width reasonably close to the minimum tree width. The following describes one such heuristic algorithm for tree decomposition.

Given a structure graph, two undirected edges (representing two interactions between cores) *cross* each other if the four cores involved interleave in their sequential positions. To find a tree decomposition of the graph, an edge that crosses the greatest number of other edges is selected and removed from the graph (but the endpoints of the edge are kept). This process is repeated until the remaining graph does not contain any crossing edges. The remaining graph is called an *outerplanar* graph (Frederickson, 1991) that has tree width 2, whose optimal tree decomposition can be found in linear time $O(\,|\mathrm{V}(H)| + |E(H)| + |A(E)|)$.

Based on the tree decomposition for the derived outerplanar graph, a tree decomposition for the original graph can be constructed by adding u, for every removed edge (u, v), to every bag on the shortest path from the bag containing u to the bag containing v in the tree. For example, Fig. 12.5(b) gives a tree decomposition

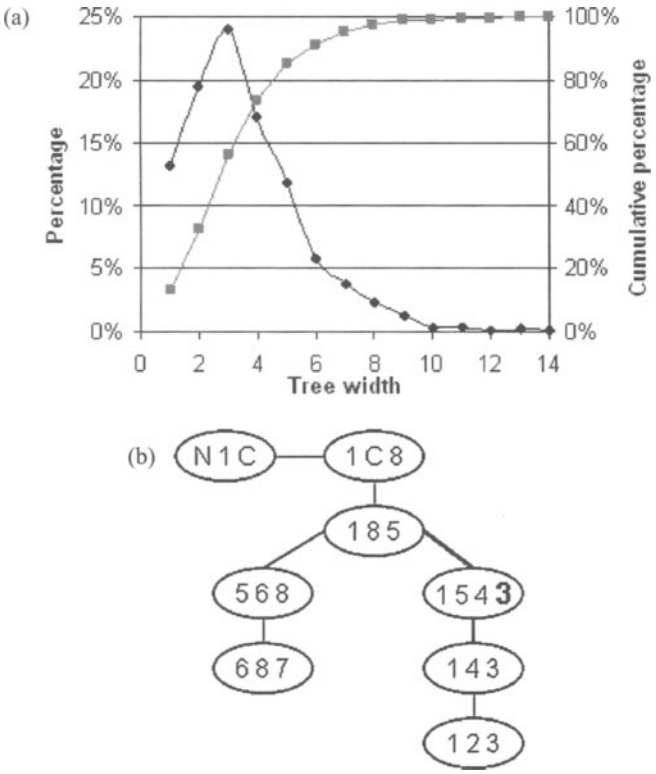

Fig. 12.5 (a) Tree width distribution of the structure graphs for 3890 protein tertiary structure templates compiled using PISCES (Wang and Dunbrack, 2003). (b) A tree decomposition for the structure graph given in Fig. 12.3(b), constructed by first removing edge (3,5), building a tree decomposition for the remaining graph, and then putting vertex 3 (in the bold font) in node $\{1,5,4\}$ on the path from node $\{1,4,3\}$ to $\{1,8,5\}$.

of the structure graph in Fig. 12.3(b). In Fig. 12.5(b) the crossing edge (3,5) is first removed and a tree decomposition is constructed for the derived outerplanar graph, which is then extended to a tree decomposition for the original graph by placing vertex 3 (in the bold font) in tree node bag {1,5,4} on the path from node {1,4,3} to node {1,8,5}. This heuristic strategy produces a tree decomposition of size at most $2 + c$ if there are c crossing edges removed. In actual applications, the obtained tree decomposition in general has a much smaller tree width than $2 + c$.

12.4.3.3 Tree-Decomposition-Based Alignment Algorithm

Once a tree decomposition (T, X) and a mapping constraint M are given, a dynamic programming can be designed to traverse the tree to find an optimal alignment between the template structure graph and a subgraph of the sequence graph G, as follows.

In a bottom-up fashion, the algorithm establishes one table for each tree node i. Such a table consists of $t + 3$ columns, where t is number of vertices (i.e., cores) contained in the bag for the tree node. Each row in the table is a combination f of candidates for the t cores constrained by M. The additional three columns store the validity, the score, and the optimality for each such combination. The computation of these three columns is performed by considering the cores in the present tree node bag that also occurs in its child node bags and in its parent node bag.

In particular, the validity asserts that the combination f of the cores' candidates is legal not only for all cores in the present tree node bag X_i, but also consistent with at least one valid combination in the tables of its two child nodes (if it has children). The score for each combination f in the present table m_k is calculated as the sum of the following three scores:

1. The alignment score of the core–candidate correspondences defined by the combination f for the cores in X_i only;
2. The maximum score over all combinations in the table m_k for the left child node k which are consistent with combination f; and
3. The maximum score over all combinations in the table m_j for the right child node j which are consistent with combination f.

The optimality column indicates if the score for each combination f is optimal over all combinations with the same candidate choices as f for the subset of cores of X_i that also occur in its parent node bag.

Figure 12.6 illustrates the computation for a row in the table of an internal node that has two child nodes. The optimal score over all combinations computed in the table for the root of the tree T is the best alignment score. A recursive routine can be used to trace back the corresponding optimal alignment.

12.4.3.4 Time Complexity Analysis

There are three steps to accomplish the structure template–target sequence alignment. The time complexity for each step is analyzed in the following.

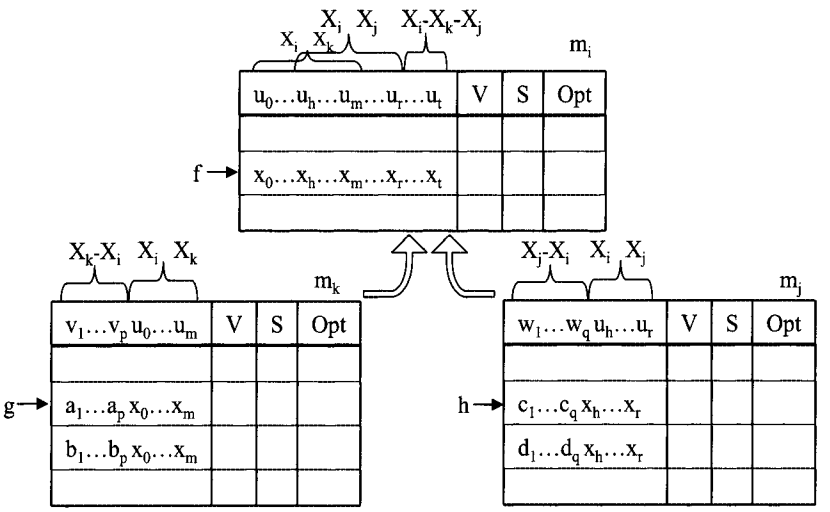

Fig. 12.6 Computing dynamic programming table m_i for internal node i based on the tables m_k and m_j for its two child nodes k and j. Row f in the table m_i is computed based on row g in m_k and row h in m_j. Rows g and h are consistent with f in the overlapping columns $\{u_0,\ldots,u_m\}$ and in columns $\{u_h,\ldots,u_r\}$, respectively; g and h are optimal over all rows in the respective tables which have the same candidate combination for these overlapping columns.

The running time for the tree-decomposition-based structure–sequence alignment is $O(k^t t^2 n)$, where k is the maximum number of candidates of a core, t is the width of the tree decomposition, and $n = |V(H)|$. This is because the tree decomposition contains at most n nodes, each maintaining a table of at most $O(k^t)$ rows. For each row in the table, the combination of core–candidate correspondences needs to be validated and a score is computed according to the objective function given in Section 12.4.3.1 (by looking up precomputed values of the three alignment energy functions E_{core}, E_{pair}, and E_{loop}). The former step needs $O(t^2)$ and the latter $O(t^2 + 2t \log_2^t k)$ (note that the table for a child node can be ordered so as to facilitate binary search by the computation for its parent node).

It takes $O(nN)$ time to preprocess the target sequence of length N to construct the sequence graph. Simultaneously, this step precomputes the values of alignment functions E_{core} and E_{pair}. The values of function E_{loop} can then be precomputed, needing time $O(k \sum_{i=1}^{l} l_i^2) = O(knN)$, where l_i is the length of the segment (between two candidates of cores) to be aligned to the ith loop in the structure, and l is the total number of loops in the structure.

Putting together the time needed by the preprocessing, tree decomposition, and the alignment gives us a loose upper bound $O(k^t nN)$, or $O(k^t N^2)$, for the total time needed by the whole algorithm for the structure–sequence alignment part of the threading. Based on preliminary computational results, the tree-decomposition-based threading algorithm seems to run faster than both the divide-and-conquer and the integer programming-based threading programs.

12.5 Assessing Statistical Significance of Threading Alignments

Protein threading is used mainly for two purposes: (a) identification of the correct structural fold or folds from a collection of protein structures for a query protein and (b) prediction of the backbone structure through identifying the correct assignment (or alignment) of the amino acids of a query sequence to the structural positions in the correct structural fold. In a way, protein structure prediction through protein threading could be considered as a problem to find a global optimal threading result among many sequence–structure alignments. The threading algorithms outlined in the previous section could only provide the best alignment between a query sequence and a particular template structure, i.e., a local optimum. The problem of finding a global optimum from the many local optima for the threading problem represents a highly challenging problem. The key reason is that the alignment scores between a query sequence and different template structures are in general not comparable directly with each other. Some structures might tend to have higher baseline threading scores than the others, no matter what query sequences are used. We can think of this problem using the following analogy from local sequence–sequence alignment problems. Suppose we have two "template" DNA sequences (consisting of four different letters). We assume that one sequence is significantly longer than the other one. For simplicity of discussion, we can regard the length of the longer one to be infinite. It is not hard to convince ourselves that for a randomly selected DNA sequence, its best (local) alignment with the longer sequence will have a higher probability to have a better score than with the shorter sequence. The reason is that the longer sequence, by chance, should have a higher probability to have portions of its sequence similar to the query sequence, compared to the shorter sequence. This indicates that the background alignment-score distributions for the two "template" sequences are different. Hence, alignment scores against these two sequences could not be compared with each other directly in a meaningful way. Similar can be said about structural templates for protein threading. To compare threading alignment scores, we need to first "normalize" the scores so that the quality of threading alignments against different templates could be compared with each other.

For sequence–sequence alignments, there have been a number of methods developed for assessing the quality of an alignment through assessing the statistical significance of its alignment score. The p-value calculation for BLAST represents a popular method for such a purpose (Altschul and Gish, 1996). Developing rigorous and effective statistical models for threading has proven to be more challenging than sequence alignment problems. Hence as of now, there has not been a dominating method, like the *p-value* calculation for BLAST, in estimating the statistical significance of a threading alignment score.

Early studies on statistical significance analysis of threading results were mostly empirical in nature. A representative of these studies was by Bryant and co-workers

(Bryant and Altschul, 1995; Panchenko et al., 2000). In their model, they assume that threading scores between a template structure and a large set of sequences of the same length and with the same amino acid composition follow a Gaussian distribution, and have estimated the parameters of the Gaussian distribution through calculating these threading scores. Specifically, for a query sequence and a template structure, they have generated a large number of sequences with the same length and the same amino acid composition as those of the query sequence by randomly reshuffling the query sequence and have used these sequences to estimate the parameters of the Gaussian distribution. With the estimated Gaussian distribution, they then calculate the probability (p-value) that a random sequence would have a better threading score than the score between the template and the query sequence. The effectiveness of this p-value calculation was demonstrated through their successful performance at the CASP3 contest (Panchenko et al., 1999). While effective, the approach has a number of limitations which have limited its wider range of applications. These limitations include (a) its high demand for computing time as it requires solving a large number of threading problems involving the reshuffled sequences and (b) its not well justified assumption that the threading scores follow a Gaussian distribution.

It has been shown that the scores of optimal sequence–sequence and structure–structure alignments generally follow extreme-value distributions (Levitt and Gerstein, 1998). Based on the assumption that sequence–structure alignment scores also follow an extreme-value distribution, Sommer et al. (2002) have recently developed another scheme for p-value estimation, in which parameters of the model are estimated by fitting the threading scores against an extreme value distribution. A key characteristic of the work is that it attempts to have a unified model for threading problems involving different lengths and compositions of the proteins involved.

Based on the random energy model (REM), Shakhnovich and co-workers systematically studied the distribution of threading alignment scores (Chen et al., 2003; Mirny et al., 2000). REM provides an adequate description of equilibrium properties of random heteropolymers, which assumes that for a query sequence, the energy of a set of alignments, including the "native" alignment with energy E_N and M decoys, followed a Gaussian distribution with an average energy E_{av} and standard deviation Σ. Assisted by a null hypothesis where the optimal native sequence–template alignment should be the highest ranked, they proved that the threading scores for random sequence–structure pairs follow an extreme-value distribution. A critical energy E_c was derived, which was a quantity in the REM, for characterizing the probable lower boundary of the density of states and it represents the most probable energy value of an optimal random sequence–structure alignment, which is determined by Eq. (12.1). They used another parameter ε for the statistical significance [Eq. (12.2)] based on E_c, corresponding to the relative energy gap for an alignment. This ε score has been tested for both gapless threading and limited-gap threading results. They found that when $\varepsilon > 1$, especially when $\varepsilon > 1.2$, there was a significant

energy gap between the optimal alignment and the decoy alignments.

$$E_c = E_{av} - \sum \sqrt{2 \ln(M/\sqrt{2\pi})}, \tag{12.1}$$

$$\varepsilon = \frac{E_N - E_{av}}{E_c - E_{av}} = \frac{E_N - E_{av}}{\sqrt{2 \ln(M/\sqrt{2\pi})}}. \tag{12.2}$$

Hence, this quality could be used as a measure for assessing the significance of a threading alignment.

A good method for assessing the statistical significance of a threading score should not only allow comparing threading results on the same footing but also provide a way to indicate if a particular fold is possibly the correct fold for a query sequence, without using other reference information.

12.6 Structure Prediction Using Protein Threading

A typical threading program consists of four key components from the implementation perspective: (a) a database of template protein structures, (b) an energy function, often residue-based, (c) a threading algorithm that can find the optimal threading alignment between a query sequence and a template structure, and (d) a method for calculating "significance" scores of threading alignments.

12.6.1 Database of Template Structures

From the prediction accuracy point of view, the larger a template structure database is, the more accurate we can expect the threading prediction will be. However, it might not always be realistic to use the whole PDB database as the template database due to the amount of time required to thread a query sequence against each PDB structure. Often a template structure database consists of a representative subset of all the structures in PDB, say PDB-select, which consists of PDB structures with the "redundant" structures removed. Here "redundant" refers to structures that have high sequence similarities to other structures in the database. To make prediction more accurate, some threading programs employ a two-stage strategy: (a) thread a query sequence against a representative structure database and to identify a few possible native-like structures, and (b) thread the query sequence against all family/superfamily members of the identified structures in (a). Certain preprocessing of the template database might be needed for some threading programs. For example, as we discussed in Section 12.4, protein structure needs to be represented as structure graphs as required by the threading algorithm.

12.6.2 Threading Energy Function

The majority of the current threading programs employ the types of energy functions outlined in Section 12.3 or their variations. These energy functions are statistics, rather than physics, based. They are used to distinguish correct structural folds from the incorrect ones and to distinguish accurate alignments against the correct structural folds from inaccurate alignments. Threading programs have been using these simple energy forms, mainly because of the consideration of computational efficiency and also partly due to the constraints of limited available structural data for more sophisticated energy forms when such statistics-based energy functions were first developed. As those simple energies began to reach their limits, we began to see more physics-based energy functions developed, as we discussed in Section 12.3. We expect that as the threading algorithms become more efficient, we will see more physics-based energy functions. We expect that one particular type of extension to the existing energy functions is to consider multibody interactions, which have been mostly ignored by existing threading programs. Recent studies have shown that multibody interactions could help to improve the performance of threading programs (Munson and Singh, 1997; Li and Liang, 2005), and hence should be considered.

12.6.3 Threading Algorithm

Existing threading programs employ various algorithmic techniques for solving the sequence–structure alignment problem, including dynamic programming with enhanced heuristics (Westhead et al., 1995; Skolnick and Kihara, 2001; Zhang et al., 1997), divide-and-conquer algorithm (Xu and Xu, 2000), and integer programming (Xu and Li, 2003; Xu et al., 2003a,b). In this chapter, we presented a new class of threading algorithm based on a tree decomposition of sequence and structure graphs. While integer programming might represent the most general framework for handling sequence–structure alignments, particularly so for threading problems considering multibody interactions, tree-decomposition-based algorithm could prove to be more popular down the road because of its conceptual simplicity and computational efficiency. We expect that a class of more general threading algorithms will begin to emerge to deal with more complex threading problems as the existing threading algorithms become faster and faster. This general class of threading algorithms should be able to handle simultaneous backbone threading and side-chain packing problems, leading to significantly more accurate capabilities in fold recognition and protein structure prediction.

12.6.4 Assessing Prediction Reliability

Existing threading programs use various ideas and techniques to assess the "significance" of threading results. These methods include z-score calculation (Sommer et al., 2002), normalized threading scores using techniques such as support vector machines or neural network (Xu et al., 2002; Ding and Dubchak, 2001), and

Table 12.3 A list of major threading programs and their URLs

123D	*http://123d.ncifcrf.gov/123D+.html*
3D-PSSM	*http://www.sbg.bio.ic.ac.uk/~3dpssm/*
FUGUE	*http://www-cryst.bioc.cam.ac.uk/~fugue/*
GenTHREADER	*http://bioinf.cs.ucl.ac.uk/psipred/*
HMAP	*http://trantor.bioc.columbia.edu/hmap/*
LOOPP	*http://cbsuapps.tc.cornell.edu/loopp.aspx*
NCBI Threading Package	*http://ncbi.nih.gov/Structure/RESEARCH/threading.shtml*
PROSPECT	*http://csbl.bmb.uga.edu/downloads/#prospect*
PROSPECTOR	*http://cssb.biology.gatech.edu/skolnick/*
RAPTOR	*http://www.bioinformaticssolutions.com/products/raptor/*
UCLA-DOE Structure Prediction Server	*http://fold.doe-mbi.ucla.edu/*

more rigorous calculations of p-values based on statistical models (Panchenko et al., 2000; Bryant and Altschul, 1995). While useful to some degree, none of these methods have reached the level of performance comparable to p-value calculations for BLAST sequence alignments (Altschul and Gish, 1996). This is possibly due to a combination of the inadequacy of the existing threading energy functions for accurate threading prediction and the lack of general understanding about distinguishing characteristics between correct and incorrect native folds and between correct and incorrect placements of amino acids into structural positions. Overall, compared to other areas of protein threading, this is a somewhat underdeveloped area. New ideas and techniques are clearly needed to fill the holes in this area.

Because of the importance of solving protein structures for functional studies and the power of threading techniques, many protein threading programs have been developed. Using these programs, a large number of protein structures have been predicted prior to the solution of their experimental structures, providing highly useful information for guiding experimental design in investigation of these proteins. Examples of such predictions include an obese gene (Madej et al., 1995), vitronectin (Xu et al., 2001), and a SARS protein (von Grotthuss et al., 2003). Table 12.3 provides a list of popular threading programs and URLs for accessing these prediction tools.

We now summarize the highlights of some of these threading programs which use different energy functions and different computational techniques, each of which has its strengths and limitations.

PROSPECT (Xu and Xu, 2000; Kim et al., 2003): The PROSPECT program employs a divide-and-conquer algorithm for rigorously solving the global optimal threading problem, which employs a somewhat standard threading energy function, including a singleton energy term and a pairwise interaction energy term plus a secondary structure fitness score and a gap penalty score. For a typical threading problem, it can find the best alignments against a template structure database of 2000+ within a couple of days on a single CPU while it can virtually get a linear

speed-up using multiple CPUs when the number of CPUs is smaller than the number of structures in the template structure database. It achieves its computational efficiency by taking advantage of the fact that protein structures generally have small *topological complexities* (Xu et al., 1998) and through using a filtering procedure to filter out "improbable" alignment positions for each core secondary structure in the template structure. While this heuristic filtering works well for the vast majority of the threading cases, it might filter out the correct alignment positions for some cases. PROSPECT normalizes the threading scores, along with various parameters of the template structure and the query sequence, using a support vector machine. Then a z-score is calculated based on the "normalized" threading scores.

RAPTOR (Xu and Li, 2003; Xu et al., 2003a,b): The RAPTOR program formulates a threading problem as a linear integer programming problem, and solves the problem using a branch-and-bound method plus a standard integer programming solver. RAPTOR employs the same energy functions of PROSPECT and uses a similar approach for assessing the "statistical significance" of threading results to that of PROSPECT. A unique feature of the program is that its threading algorithm is more rigorous than that of PROSPECT as it does not use a heuristic filter to filter out "improbable" alignment positions. For a typical threading problem, it takes minutes to hours to thread the query sequence against 2000 structures in its structure database. Since the program is data-parallelizable, its speed-up is virtually linear when running on multiprocessor computers.

GenTHREADER (Jones, 1999b) and an improved version mGenTHREADER (McGuffin and Jones, 2003) use PSI-BLAST profile (Altschul et al., 1997) and predicted secondary structures by PSIPRED (Jones, 1999a) for threading. It employs a double dynamic programming strategy (Jones et al., 1992) in its threading program. The algorithm does not treat pairwise interactions rigorously but its performance has been among the top threading programs, indicating the effectiveness of this strategy. A Web server for this program has been set up at *http://bioinf.cs.ucl.ac.uk/psipred/*. A user in general can expect the return of a threading prediction in minutes. The prediction program runs fast enough that it can be used for genome-scale protein structure predictions.

PROSPECTOR (Skolnick and Kihara, 2001) recognizes native-like structural folds using a hierarchical strategy for obtaining sequence profiles. It uses two types of sequence profiles, one type derived using close homologous sequences whose sequence identity lies between 35% and 90%, and another type constructed using more remote homologous sequences with a FASTA E-value less than 10. Both types of sequence profiles are incorporated into a typical threading energy as described in Section 12.3 to screen a structural database. The program uses a dynamic programming algorithm to find the best threading alignment, and employs z-scores for assessing the significance of each threading alignment.

FUGUE (Shi et al., 2001) uses a typical threading energy function as described in Section 12.3, with some unique features: (1) its structural environment singleton term includes a term for hydrogen bonding status, and the singleton term was derived from structural alignments in the HOMSTRAD database (de Bakker et al., 2001; *http://www-cryst.bioc.cam.ac.uk/homstrad/*); (2) its gap penalties are

structure-dependent based on solvent accessibility, its position relative to the secondary structure elements, and the conservation of the secondary structure elements; and (3) its alignment is based on multiple sequences against multiple structures to enrich the conservation/variation information. FUGUE uses dynamic programming as its threading algorithm and employs *z*-scores to assess the statistical significance of a threading result.

Since each of these threading programs has its own strengths and limitations, a popular strategy for predicting a protein structure is to use multiple prediction programs and combine their prediction results. Further discussion on this topic is given in Chapter 17. We now use PROSPECT as an example to illustrate how to use a threading program for predicting structures of SARS-CoV proteins (Wan et al., 2005), which play a role in the development of the SARS disease. We used the PROSPECT pipeline to survey all of the 11 Open Reading Frames (ORFs) in SARS-CoV strain Urbani (GenBank ID: 30027617), one of the first SARS-CoV genomes.

Among the 11 ORFs, the S and M proteins play a key role in the virus infection process. Interestingly, both the M protein and the S2 domain in the S protein are predicted to adopt the fold of Ig-like beta sandwich. The structural similarity suggests that the S2 domain and the M protein may be evolutionarily related through gene fusion and duplication, although their sequences do not have significant similarity after a long period of evolution. The threading results might explain how the M protein interacts with the S2 domain, for the virus assembly: since the S2 domain with the fold of Ig-like beta sandwich can interact with the S1 domain, the M protein with the same fold could possibly interact with the S1 domain. This suggests that the S1 domain may act as an on/off switch between the S2 domain and the M protein. Such a mechanism may suggest that the M protein could also be involved in the virus–host cell interaction. This hypothesis was supported by a recent study in the murine hepatitis coronavirus study, which showed that glycosylation of the M protein affected the interferogenic capacity of the virus (de Haan et al., 2004).

Threading programs have been used for genome-scale applications. A recent study (Guo et al., 2004) performed structure prediction for all of the ORFs of *Pyrococcus furiosus*, which is found in the marine sand surrounding sulfurous volcanoes and can grow at temperatures above 100°C. The microbe utilizes peptides, proteins, and some carbohydrates as carbon sources. Its entire genome is about 2 Mb in length with 2195 annotated ORFs. Out of a total of 2195 ORFs, 540 are predicted to be membrane proteins, and 753 proteins can be predicted with structures in high confidence, among which 190 ORFs cannot be detected using PSI-BLAST.

12.7 Improving Threading-Based Structure Prediction

Recent prediction results in CASPs indicate that even when the correct structural folds are identified, the threading alignments could often be off. From the same statistics, we found that the overall alignment accuracy has not been improved over the past few CASPs. In addition, the alignment accuracy of the best CASP models using templates with <30% sequence identity ranges from 60% to 90% (Venclovas

et al., 2003). All these statistics suggest that there is significant room for improvement in threading alignments. The prediction accuracy of the current threading programs is mainly limited by the inaccuracy of statistics-based and residue-based threading energy functions. While improving the threading energy functions represents one direction to take for improving threading performance, other approaches may also help, which include (a) application of partial experimental data as constraints in the threading process and (b) refinement of threaded structures using molecular dynamics and energy minimization. We refer the reader to Chapter 11 for related discussions.

12.7.1 Application of Experimental Data as Threading Constraints

Often partial structural data is available for specific proteins before the determination of the detailed structure of a protein. These partial structural data might be in the form of (a) residue–residue distances such as disulfides between specific cysteines, (b) specific residues involved in particular active sites, binding sites, or other functionally important sites, (c) specific residues known to be on the surface of a protein structure, or any other information providing geometric information about specific residues in a protein structure. In addition to such information about specific residues, there are experimental techniques that can be used to generate geometric information in a systematic manner.

NMR represents one such technique. NMR methods solve a protein structure through generating either distance restraints [called *NOE*, or nuclear Overhauser effect, distances (Prestegard, 1998)] between different residues (or more specifically different atoms) or orientations of certain chemical bonds in a protein, called *residual dipolar coupling* (Tolman et al., 1995). Then a protein structure is solved through finding structural models that are consistent with the collected geometric constraints and have their energy minimized. To accurately solve a protein structure using NMR technique, it typically requires 15–20 distance restraints per residue (Clore et al., 1993), which will require multiple NMR experiments. Partial distance restraints could possibly be collected using fewer NMR experiments, possibly involving labeling of specific amino acid types. Similar can be said about orientation information collected through residual dipolar coupling experiments. While these partial NMR data might not be sufficient for solving a protein structure accurately, they provide highly useful constraints for protein fold recognition and backbone structure prediction by a threading method.

Chemical cross-linking experiments provide another systematic approach to generating partial structural information for a protein. In such experiments, chemical cross-linkers, with customized arm lengths, are designed to link specific types of amino acids within a certain distance range (Cohen and Sternberg, 1980). Such experiments followed by tandem mass spectrometry experiments and data interpretation could provide distance information between certain amino acids. Such an approach has been used for structural data collection for both soluble and membrane proteins (Yan et al., 2005), which have then been used for protein structure prediction (Young et al., 2000).

One way to use such structural information, such as distances or structural locations, in a threading program is to add an energy term in the threading energy function, which measures the consistency between the collected structural data and a threading alignment. For example, if residues A and B are known to be within a certain distance, then an energy term could be specifically designed to penalize threading alignments which violate this particular geometric constraint. The energy term could be designed so that the bigger the violation, the larger the penalty. Similarly, if a residue X is known to be on the surface of a protein structure, an energy term could be specifically designed for this knowledge so that it penalizes threading alignments that do not put X into a surface position in the template structure, and the amount of penalty could be designed to reflect the degree of violation of this particular knowledge. To deal with all geometric constraints, we can design a new energy term E_G,which is the sum of the individual penalty functions for all the specific geometric constraints. The overall scaling factor for this new energy term in a threading energy function could be empirically determined based on a training data set, for which actual partial experimental data is available.

The effectiveness of applying such partial structural data in a threading program has been documented in a number of studies (Xu et al., 2000b,c; Young et al., 2000; Qu et al., 2004a,b). Table 12.4 shows a systematic study on improving the

Table 12.4 Improvement of threading performance using increasingly more NOEs

			RMSD Å / rank vs. percentage of assigned *ss*					
class	query	temp	0	20%	40%	60%	80%	100%
	1bbn	1cnt	**16.2/29**	6.5/11	6.5/4	6.5/2	6.4/3	**5.3/1**
	1itl	3inkC	**13.1/7**	6.6/2	9.2/1	6.6/2	3.5/1	**3.5/2**
α	1ner	1lmb3	**3.8/57**	3.0/11	3.0/6	3.0/5	3.0/4	**3.0/4**
	1il6	1bgj	**3.0**/1	2.8/1	2.8/1	2.8/1	2.8/1	**2.9**/1
	1ocd	1cyj	**3.4**/1	3.3/1	3.3/1	3.3/1	3.3/1	**3.3**/1
	1maj	1agdB	8.8/**37**	10.3/34	10.4/34	10.4/18	9.0/14	9.0/**9**
	1nct	1bec	**14.7**/1	14.2/1	14.2/1	14.2/1	14.2/1	**14.2**/1
β	1bla	1hce	**7.7**/1	3.2/1	3.2/1	2.4/1	2.4/1	**2.4**/1
	1vhp	1cd8	**4.5**/1	4.0/1	4.0/1	4.0/1	4.0/1	**4.0**/1
	1ghj	liyu	**2.0**/1	2.0/1	2.0/1	1.8/1	1.8/1	**1.8**/1
	1a7i	1qli	2.8/1	2.9/1	2.9/1	2.9/1	2.8/1	2.8/1
	1afi	2acy	**6.8/376**	7.8/369	7.8/235	5.6/133	5.7/53	**5.6/38**
	3trx	1a8y	**4.1**/1	2.9/1	2.7/1	2.8/1	2.7/1	**2.7**/1
α/β	1ikm	1dokB	**2.1/2**	2.0/2	2.0/1	2.0/1	2.0/1	**2.0/1**
	3phy	1bv6	**12.2/41**	4.1/24	3.1/14	3.1/9	3.4/5	**3.1/2**
	1crp	1byuB	**2.6**/1	2.2/1	2.2/1	2.2/1	2.2/1	**2.2**/1
	1fht	2ula	**4.5**/1	2.2/1	2.2/1	2.2/1	2.2/1	**2.2**/1

"Query" and "temp" represent the PDB codes of the query and template proteins, respectively. "RMSD/rank vs. percentage of assigned *ss*" are the Cα-RMSD between the experimental structure and the predicted structure using MODELLER based on threading alignments for the alignable portions in the structure–structure alignment between the query protein and the template, and the rank of the correct template structure among 667 templates. 0%, 20%, 40%, . . . , 100% represent the percentage of residues with secondary structure assignment, respectively. The highlighted numbers show improvement between using no secondary structure information and using full secondary structure assignments.

threading performance by incrementally increasing the number of NMR/NOE distance constraints used in a threading process. It is seen that distance constraints help both fold recognition and threading alignment accuracy.

12.7.2 Improving Structural Quality through Molecular Dynamics and Energy Minimization

In the best scenario, a threading program can provide an accurate prediction of the backbone atoms of a protein structure, which is still a long way from having a detailed all-atom structure. In the most general situation, a threading program could provide a somewhat accurate structure for the backbone atoms in the core secondary structures while predictions for the loop regions are often not accurate. The reason is that threading predicts a structure based on a known template structure. While the core secondary structures among homologous proteins are generally "well" conserved, loops are often not. Hence, template-based loop predictions are generally not accurate. Fortunately, existing methods for short loop prediction (<14 residues) have reached a level that the predicted loops could be as accurate as the predicted core structure. For example, the recent work by Jacobson et al. (2004) has achieved prediction accuracy of 0.43 Å for 5-residue loops, 0.84 Å for 8-residue loops, and 1.63 Å for 11-residue loops, using an accurate all-atom energy function and hierarchical refinement protocol.

Potentially the predicted full backbone structure, after adding loop structures, could be refined using an energy-based approach. To do this, one needs to put all the atoms, backbone and side chains, into a structural model. One can use alignments with the selected templates in fold recognition to produce a 3D atomic model through homology modeling tools, such as MODELLER (Sali and Blundell, 1990), which runs a protocol of energy minimization and molecular dynamics simulation to refine a structural model. After a structure model is generated, one can apply structure assessment tools such as WHATIF(Vriend, 1990) and PROCHECK (Laskowski et al., 1993) to evaluate the packing and backbone conformations, the inside/outside occupancies of hydrophobic and hydrophilic residues, and stereochemical quality of a predicted structure. Based on this assessment, a user can pick the best among the multiple structures derived from an alignment.

12.8 Challenging Issues

While widely used, the potential of protein threading as a protein structure and function prediction technique is far from being fully realized. There are a number of factors that have limited its wider range of applications. First, fold recognition for structural analogues and some remote homologues is still challenging (Kinch et al., 2003; Sippl et al., 2001). Such proteins might account for about 40% of all proteins encoded in a typical genome according to our studies (Xu et al., 2003; Guo et al., 2004). These structures are theoretically modelable using comparative

modeling techniques such as protein threading, but the predictions typically gave a low confidence level and the results may be wrong. Second, even when a correct fold is identified, the accuracy of threading alignment has been about 60–90% for proteins with less than 30% sequence identity with their template structures (Venclovas et al., 2003). Novel ideas and new techniques are clearly needed now to make a major jump in improving the prediction capability of the existing threading methods; this has become quite clear based on the slow and incremental improvements in threading performance in the past few CASP contests (Venclovas et al., 2003).

Energy Function

The current energy functions are generally coarse gained mainly to achieve fast predictions. Given the significant advances in computer hardware and algorithm development, it may be the time to use more sophisticated energy functions. For example, multibody interactions and more physical energy functions may help improve the threading prediction accuracy.

Threading Algorithm and Implementation

Although many theoretical studies have been carried out for threading algorithms, there is still significant room for further improving the computational efficiency of threading programs. Better search methods against structural templates using advanced database techniques have not been explored thoroughly. More work can be done at the implementation level in a similar way to the implementation of BLAST, where many low-level operations were implemented in a highly efficient manner. In addition, algorithmic development needs to address new types of energy functions, such as new threading algorithms that could handle simultaneous backbone threading and side-chain packing to fully take advantage of more detailed energy function forms. Threading algorithms that could handle multibody interactions and energy functions capturing more global properties (e.g., compactness) of proteins are clearly underdeveloped.

Statistical Significance Analysis of Threading Results

Existing confidence assessments are either too time-consuming in computation or not sufficiently accurate. More rigorous and faster assessment techniques for threading are clearly needed to achieve comparable performance to that of BLAST. Assessments of different alignments using the same fold were basically not studied. Furthermore, identification of "reliable" versus "unreliable" parts of a threaded structure, and quantitative assessment of the structural deviations in terms of RMSD for regions of predicted structures have not been achieved.

Consensus Building and Subdomain Threading

It has been found that using multiple fold recognition programs to build consensus of structural template is an effective way to increase the prediction accuracy (Lundstrom et al., 2001). Furthermore, one can thread subdomain structures and use these substructures from different templates to build a new structure through

a shotgun approach (Fischer, 2000, 2003). Currently a consensus was built by a simple scheme of majority vote. Much statistics can be done to do this in a more scientifically sound way. In addition, how to piece different substructures together to form a global protein structure is another challenging issue. Further discussion on consensus building and subdomain threading can be found in Chapter 17.

12.9 Summary

As a structure prediction technique, threading potentially applies to at least 80% of all protein families. However, the application of threading to membrane proteins has been very limited due to the lack of available structural templates. Threading techniques have been widely used for various purposes in biological studies, including (a) functional studies of proteins and experimental design (e.g., targeted mutagenesis) (Madej et al., 1995; Xu et al., 2001; von Grotthuss et al., 2003), (b) genome annotation (Xu et al., 2003; McGuffin et al., 2004), (c) helping solve experimental structures (Ye et al., 2004), (d) modeling protein complex structures (Lu et al., 2003), (e) prediction of misfolded structure (see Chapter 9), and (f) protein design (Sorenson and Head-Gordon, 1999). To further increase the utility of threading techniques to meet the needs for genome-scale protein structure prediction to keep up with the rate of genome sequencing and gene prediction, we clearly need a new generation of threading energy functions, threading algorithms, methods for assessing the statistical significance of threading results, and refinement of threaded structures.

Suggested Further Reading

There are several comprehensive reviews and books on various aspects of threading. Recent reviews related to threading include Fetrow et al. (2002) and Godzik (2003). A number of books also provide some general coverage of threading and protein structure predictions (Tsigelny, 2002; Jiang et al. 2002; Bourne and Weissig, 2003). For scoring function, the readers can find more information in Chapters 2 and 3 of this book. For more information about general protein structure and structure–function relationship, we recommend Branden and Tooze (1999) and Lesk (2001).

Acknowledgments

This research was sponsored in part by the U.S. Department of Energy's Genomes to Life program (*www.doegenomestolife.org*) under project "Carbon Sequestration in *Synechococcus sp.*: From Molecular Machines to Hierarchical Modeling" (*www.genomes2life.org*). YX and ZJL's work was also supported in part by NSF/DBI-0354771, NSF/ITR-IIS-0407204, and a "Distinguished Cancer Scholar" grant from the Georgia Cancer Coalition. DX's work was also partially funded by NSF/EIA-0325386.

References

Alexandrov, N., and I. Shindyalov. 2003. PDP: protein domain parser. *Bioinformatics* 19:429–430.

Altschul, S.F., and W. Gish. 1996. Local alignment statistics. *Methods Enzymol.* 266:460–480.

Altschul, S.F., T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. 1997. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Res.* 25:3389–3402.

Andreeva, A., D. Howorth, S.E. Brenner, T.J. Hubbard, C. Chothia, and A.G. Murzin. 2004. SCOP database in 2004: Refinements integrate structure and sequence family data. *Nucleic Acids Res.* 32:D226–D229.

Apic, G., J. Gough, and S.A. Teichmann. 2001a. Domain combinations in archaeal, eubacterial and eukaryotic proteomes. *J. Mol. Biol.* 310:311–325.

Apic, G., J. Gough, and S.A. Teichmann. 2001b. An insight into domain combinations. *Bioinformatics* 17 (Suppl. 1):83–89.

Arnborg, S., and A. Proskurowski. 1989. Linear time algorithms for NP-hard problems restricted to partial k-tree. *Discrete Appl. Math.* 23:11–24.

Bairoch, A., R. Apweiler, C.H. Wu, W.C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane, M.J. Martin, D.A. Natale, C. O'Donovan, N. Redaschi, and L.S. Yeh. 2005. The Universal Protein Resource (UniProt). *Nucleic Acids Res.* 33:D154–D159.

Baker, D., and A. Sali. 2001. Protein structure prediction and structural genomics. *Science* 294:93–96.

Barton, G.J., and M.J. Sternberg. 1987. A strategy for the rapid multiple alignment of protein sequences. Confidence levels from tertiary structure comparisons. *J. Mol. Biol.* 198:327–337.

Barton, G.J., and M.J. Sternberg. 1990. Flexible protein sequence patterns. A sensitive method to detect weak structural similarities. *J. Mol. Biol.* 212:389–402.

Bodlaender, H.L. 1996. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25:1305–1317.

Bourne, P.E., and H. Weissig (eds.). 2003. *Structural Bioinformatics.* New York, Wiley–Liss.

Bowie, J.U., R. Luthy, and D. Eisenberg. 1991. A method to identify protein sequences that fold into a known three-dimensional structure. *Science* 253:164–170.

Branden, C., and J. Tooze. 1999. *Introduction to Protein Structure*, 2nd ed. New York, Garland Publishing.

Brassard, G., and P. Bratley. 1996. *Fundamentals of Algorithmics*. Upper Saddle River, NJ, Prentice–Hall, pp. 265–266.

Brenner, S.E., C. Chothia, T.J. Hubbard, and A.G. Murzin. 1996. Understanding protein structure: Using scop for fold interpretation. *Methods Enzymol.* 266:635–643.

Bryant, S.H., and S.F. Altschul. 1995. Statistics of sequence–structure threading. *Curr. Opin. Struct. Biol.* 5:236–244.

Calland, P.Y. 2003. On the structural complexity of a protein. *Protein Eng.* 16:79–86.
Chen, W., L. Mirny, and E.I. Shakhnovich. 2003. Fold recognition with minimal gaps. *Proteins* 51:531–543.
Clore, G.M., M.A. Robien, and A.M. Gronenborn. 1993. Exploring the limits of precision and accuracy of protein structures determined by nuclear magnetic resonance spectroscopy. *J. Mol. Biol.* 231:82–102.
Cohen, F.E., and M.J. Sternberg. 1980. On the use of chemically derived distance constraints in the prediction of protein structure with myoglobin as an example. *J. Mol. Biol.* 137:9–22.
Coulson, A.F., and J. Moult. 2002. A unifold, mesofold, and superfold model of protein fold use. *Proteins* 46:61–71.
de Bakker, P.I., A. Bateman, D.F. Burke, R.N. Miguel, K. Mizuguchi, J. Shi, H. Shirai, and T.L. Blundell. 2001. HOMSTRAD: Adding sequence information to structure-based alignments of homologous protein families. *Bioinformatics* 17:748–749.
de Haan, C.A., K. Stadler, G.J. Godeke, B.J. Bosch, and P.J. Rottier. 2004. Cleavage inhibition of the murine coronavirus spike protein by a furin-like enzyme affects cell–cell but not virus–cell fusion. *J. Virol.* 78:6048–6054.
DeWitte, R.S., and E.I. Shakhnovich. 1996. SMoG: de novo design method based on simple, fast, and accurate free energy estimates .1. Methodology and supporting evidence. *J. Am. Chem. Soc.* 118:11733–11744.
Dietmann, S., and L. Holm. 2001. Identification of homology in protein structure classification. *Nat. Struct. Biol.* 8:953–957.
Ding, C.H., and I. Dubchak. 2001. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics* 17:349–358.
Doolittle, R.F. 1995. The multiplicity of domains in proteins. *Annu. Rev. Biochem.* 64:287–314.
Dutta, S., and H.M. Berman. 2005. Large macromolecular complexes in the Protein Data Bank: A status report. *Structure* 13:381–388.
Ekman, D., A.K. Bjorklund, J. Frey-Skott, and A. Elofsson. 2005. Multi-domain proteins in the three kingdoms of life: Orphan domains and other unassigned regions. *J. Mol. Biol.* 348:231–243.
Elofsson, A., D. Fischer, D.W. Rice, S.M. Le Grand, and D. Eisenberg. 1996. A study of combined structure/sequence profiles. *Fold. Des.* 1:451–461.
Fetrow, J.S., A. Giammona, A. Kolinski, and J. Skolnick. 2002. The protein folding problem: A biophysical enigma. *Curr. Pharm. Biotechnol.* 3:329–347.
Finkelstein, A.V., and O.B. Ptitsyn. 1987. Why do globular proteins fit the limited set of folding patterns? *Prog. Biophys. Mol. Biol.* 50: 171–190.
Fischer, D. 2000. Hybrid fold recognition: Combining sequence derived properties with evolutionary information. *Pacific Symp. Biocomputing*, Hawaii, pp. 119–130, World Scientific.
Fischer, D. 2003. 3D-SHOTGUN: A novel, cooperative, fold-recognition meta-predictor. *Proteins* 51:434–441.

Fischer, D., and D. Eisenberg. 1996. Protein fold recognition using sequence-derived predictions. *Protein. Sci.* 5:947–955.

Fischer, D., A. Elofsson, D. Rice, and D. Eisenberg. 1996a. Assessing the performance of fold recognition methods by means of a comprehensive benchmark. *Pac. Symp. Biocomput.* 300–318.

Fischer, D., D. Rice, J.U. Bowie, and D. Eisenberg. 1996b. Assigning amino acid sequences to 3-dimensional protein folds. *FASEB J.* 10:126–136.

Frederickson, G.N. 1991. Planar graph decomposition and all pairs shortest paths. *J. Assoc. Comput. Mach.* 38:162–204.

Gaasterland, T. 1998. Structural genomics: Bioinformatics in the driver's seat. *Nat. Biotechnol.* 16:625–627.

Gelfand, M.S., E.V. Koonin, and A.A. Mironov. 2000. Prediction of transcription regulatory sites in Archaea by a comparative genomic approach. *Nucleic Acids Res.* 28:695–705.

Gerlt, J.A., and P.C. Babbitt. 2000. Can sequence determine function? *Genome Biol.* 1(5):reviews 0005.1–0005.10.

Gerstein, M. 1997. A structural census of genomes: Comparing bacterial, eukaryotic, and archaeal genomes in terms of protein structure. *J. Mol. Biol.* 274:562–576.

Gerstein, M. 1998. How representative are the known structures of the proteins in a complete genome? A comprehensive structural census. *Fold. Des.* 3:497–512.

Gerstein, M., and H. Hegyi. 1998. Comparing genomes in terms of protein structure: Surveys of a finite parts list. *FEMS Microbiol. Rev.* 22:277–304.

Godzik, A. 2003. Fold recognition methods. *Methods Biochem Anal.* 44:525–546.

Guo, J.T., K. Ellrott, W.J. Chung, D. Xu, S. Passovets, and Y. Xu. 2004. PROSPECT-PSPP: An automatic computational pipeline for protein structure prediction. *Nucleic Acids Res.* 32(Web Server issue):W522–525.

Hobohm, U., M. Scharf, R. Schneider, and C. Sander. 1992. Selection of representative protein data sets. *Protein Sci.* 1:409–417.

Holm, L., and C. Sander. 1996a. Mapping the protein universe. *Science* 273:595–603.

Holm, L., and C. Sander. 1996b. The FSSP database: Fold classification based on structure–structure alignment of proteins. *Nucleic Acids Res.* 24:206–209.

Jacobson, M.P., D.L. Pincus, C.S. Rapp, T.J. Day, B. Honig, D.E. Shaw, and R.A. Friesner. 2004. A hierarchical approach to all-atom protein loop prediction. *Proteins* 55:351–367.

Jiang, T., Y. Xu, and M. Zhang (eds.). 2002. *Current Topics in Computational Molecular Biology.* Cambridge, MA, MIT Press.

Jones, D.T. 1999a. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* 292:195–202.

Jones, D.T. 1999b. GenTHREADER: An efficient, and reliable protein fold recognition method for genomic sequences. *J. Mol. Biol.* 287:797–815.

Jones, D.T., W.R. Taylor, and J.M. Thornton. 1992. A new approach to protein fold recognition. *Nature* 358:86–89.

Kim, D, D. Xu, J.T. Guo, K. Ellrott, and Y. Xu. 2003. PROSPECT II: Protein structure prediction program for genome-scale applications. *Protein Eng.* 16:641–650.

Kinch, L.N., J.O. Wrabl, S.S. Krishna, I. Majumdar, R.I. Sadreyev, Y. Qi, J. Pei, H. Cheng, and N.V. Grishin. 2003. CASP5 assessment of fold recognition target predictions. *Proteins* 53(Suppl.6):395–409.

Koonin, E.V., Y.I. Wolf, and G.P. Karev. 2002. The structure of the protein universe and genome evolution. *Nature* 420:218–223.

Laskowski, R.A., M.W. MacArthur, D.S. Moss, and J.M. Thornton. 1993. PROCHECK: A program to check the stereochemical quality of protein structures. *J. Appl. Crystallogr.* 26:283–291.

Lathrop, R.H. 1994. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Eng.* 7:1059–1068.

Lesk, A. 2001. *Introduction to Protein Architecture: The Structural Biology of Proteins.* London, Oxford University Press.

Levitt, M., and M. Gerstein. 1998. A unified statistical framework for sequence comparison and structure comparison. *Proc. Natl. Acad. Sci. USA* 95:5913–5920.

Li, H., R. Helling, C. Tang, and N. Wingreen. 1996. Emergence of preferred structures in a simple model of protein folding. *Science* 273:666–669.

Li, H., C. Tang, and N.S. Wingreen. 1998. Are protein folds atypical? *Proc. Natl. Acad. Sci. USA* 95:4987–4990.

Li, H., C. Tang, and N.S. Wingreen. 2002. Designability of protein structures: A lattice-model study using the Miyazawa–Jernigan matrix. *Proteins* 49:403–412.

Lu, H., and J. Skolnick. 2001. A distance-dependent atomic knowledge-based potential for improved protein structure selection. *Proteins* 44:223–232.

Li, X., and J. Liang. 2005. Geometric cooperativity and anti-cooperativity of three-body interactions in native proteins. *Proteins* 60:46–65.

Lu, L., A.K. Arakaki, H. Lu, and J. Skolnick. 2003. Multimeric threading-based prediction of protein–protein interactions on a genomic scale: Application to the Saccharomyces cerevisiae proteome. *Genome Res.* 13(6A):1146–1154.

Lund, O., K. Frimand, J. Gorodkin, H. Bohr, J. Bohr, J. Hansen, and S. Brunak. 1997. Protein distance constraints predicted by neural networks and probability density functions. *Protein Eng.* 10:1241–1248.

Lundstrom, J., L. Rychlewski, J. Bujnicki, A. Elofsson. 2001. Pcons: A neural-network-based consensus predictor that improves fold recognition. *Protein Sci.* 10:2354–2362.

Madej, T., M.S. Boguski, and S.H. Bryant. 1995. Threading analysis suggests that the obese gene product may be a helical cytokine. *FEBS Lett.* 373:13–18.

Makarova, K.S., L. Aravind, M.Y. Galperin, N.V. Grishin, R.L. Tatusov, Y.I. Wolf, and E.V. Koonin. 1999. Comparative genomics of the Archaea (Euryarchaeota): Evolution of conserved protein families, the stable core, and the variable shell. *Genome Res.* 9:608–628.

May, R.M. 1988. How many species are there on earth. *Science* 241:1441–1449.

McGuffin, L.J., and D.T. Jones. 2003. Improvement of the GenTHREADER method for genomic fold recognition. *Bioinformatics* 19:874–881.

McGuffin, L.J., S.A. Street, K. Bryson, S.A. Sorensen, and D.T. Jones. 2004. The Genomic Threading Database: A comprehensive resource for structural annotations of the genomes from key organisms. *Nucleic Acids Res.* 32(Database issue):D196–199.

Melo, F., and E. Feytmans. 1997. Novel knowledge-based mean force potential at atomic level. *J. Mol. Biol.* 267:207–222.

Mirny, L.A., A.V. Finkelstein, and E.I. Shakhnovich. 2000. Statistical significance of protein structure prediction by threading. *Proc. Natl. Acad. Sci. USA* 97:9978–9983.

Munson, P.J., and R.K. Singh. 1997. Statistical significance of hierarchical multi-body potentials based on Delaunay tessellation and their application in sequence–structure alignment. *Protein Sci.* 6:1467–1481.

Murzin, A.G., S.E. Brenner, T. Hubbard, and C. Chothia. 1995. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247:536–540.

Orengo, C.A., D.T. Jones, and J.M. Thornton. 1994. Protein superfamilies and domain superfolds. *Nature* 372:631–634.

Orengo, C.A., A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. 1997. CATH—A hierarchic classification of protein domain structures. *Structure* 5:1093–1108.

Orengo, C.A., and W.R. Taylor. 1993. A local alignment method for protein structure motifs. *J. Mol. Biol.* 233:488–497.

Panchenko, A., A. Marchler-Bauer, and S.H. Bryant. 1999. Threading with explicit models for evolutionary conservation of structure and sequence. *Proteins Suppl.* 3:133–140.

Panchenko, A.R., A. Marchler-Bauer, and S.H. Bryant. 2000. Combination of threading potentials and sequence profiles improves fold recognition. *J. Mol. Biol.* 296:1319–1331.

Papadimitriou, C., and H. Christos. 1998. *Combinatorial Optimization: Algorithms and Complexity*. New York, Dover Publications.

Prestegard, J.H. 1998. New techniques in structural NMR–anisotropic interactions. *Nat. Struct. Biol.* 5(Suppl.):517–522.

Qu, Y., J.T. Guo, V. Olman, and Y. Xu. 2004a. Protein fold recognition through application of residual dipolar coupling data. *Pac. Symp. Biocomput.* pp. 459–470.

Qu, Y., J.T. Guo, V. Olman, and Y. Xu. 2004b. Protein structure prediction using sparse dipolar coupling data. *Nucleic Acids Res.* 32:551–561.

Richardson, J.S. 1981. The anatomy and taxonomy of protein structure. *Adv. Protein Chem.* 34:167–339.

Robertson, N., and P.D. Seymour. 1986. Graph minors .2. algorithmic aspects of tree-width. *J. Algorithm* 7:309–322.

Rost, B., R. Schneider, and C. Sander. 1997. Protein fold recognition by prediction-based threading. *J. Mol. Biol.* 270:471–480.

Sali, A., and T.L. Blundell. 1990. Definition of general topological equivalence in protein structures. A procedure involving comparison of properties and

relationships through simulated annealing and dynamic programming. *J. Mol. Biol.* 212:403–428.

Samudrala, R., and J. Moult. 1998. An all-atom distance-dependent conditional probability discriminatory function for protein structure prediction. *J. Mol. Biol.* 275:895–916.

Shi, J., L. Blund, and K. Mizuguchi. 2001. FUGUE: Sequence–structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *J. Mol. Biol.* 310:243–257.

Sippl, M.J.. 1990. Calculation of conformational ensembles from potentials of mean force. An approach to the knowledge-based prediction of local structures in globular proteins. *J. Mol. Biol.* 213:859–883.

Sippl, M.J., P. Lackner, F.S. Domingues, A. Prlic, R. Malik, A. Andreeva, and M. Wiederstein. 2001. Assessment of the CASP4 fold recognition category. *Proteins* Suppl. 5:55–67.

Skolnick, J., J.S. Fetrow, and A. Kolinski. 2000. Structural genomics and its importance for gene function analysis. *Nat. Biotechnol* 18:283–287.

Skolnick, J., and D. Kihara. 2001. Defrosting the frozen approximation: PROSPECTOR: A new approach to threading. *Proteins* 42:319–331.

Sommer, I., A. Zien, N. von Ohsen, R. Zimmer, and T. Lengauer. 2002. Confidence measures for protein fold recognition. *Bioinformatics* 18:802–812.

Song, Y., K. Ellrott, C. Liu, J. Guo, Y. Xu, and L. Cai. 2005. Tree decomposition based protein threading. Submitted.

Sorenson, J.M., and T. Head-Gordon. 1999. Redesigning the hydrophobic core of a model beta-sheet protein: Destabilizing traps through a threading approach. *Proteins* 37:582–591.

Tatusov, R.L., M.Y. Galperin, D.A. Natale, and E.V. Koonin. 2000. The COG database: A tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Res.* 28:33–36.

Taylor, W.R., and C.A. Orengo. 1989. Protein structure alignment. *J. Mol. Biol.* 208:1–22.

Tolman, J.R., J.M. Flanagan, M.A. Kennedy, and J.H. Prestegard. 1995. Nuclear magnetic dipole interactions in field-oriented proteins: Information for structure determination in solution. *Proc. Natl. Acad. Sci. USA* 92:9279–9283.

Tsigelny, I.F. (eds.). 2002. *Protein Structure Prediction: Bioinformatic Approach.* La Jolla, CA, International University Line Publishers.

Venclovas, C., A. Zemla, K. Fidelis, and J. Moult. 2003. Assessment of progress over the CASP experiments. *Proteins* 53(Suppl. 6):585–595.

von Grotthuss, M., L.S. Wyrwicz, and L. Rychlewski. 2003. mRNA cap-1 methyltransferase in the SARS genome. *Cell* 113:701–702.

Vriend, G. 1990. WHAT IF: A molecular modelling and drug design program. *J. Mol. Graph.* 8:52–56.

Wan, X.F., D. Ataman, and D. Xu. 2005. Application of computational biology in understanding emerging infectious diseases: Inferring the biological function for S-M complex of SARS-CoV. In *Progress in Bioinformatics*. New York, Nova Science Publishers, pp. 55–80.

Wang, G., and R.L. Dunbrack, Jr. 2003. PISCES: A protein sequence culling server. *Bioinformatics* 19:1589–1591.

Wang, Z.X. 1996. How many fold types of protein are there in nature? *Proteins* 26:186–191.

Westhead, D.R., V.P. Collura, M.D. Eldridge, M.A. Firth, J. Li, and C.W. Murray. 1995. Protein fold recognition by threading: Comparison of algorithms and analysis of results. *Protein Eng.* 8:1197–1204.

Wetlaufer, D.B. 1973. Nucleation, rapid folding, and globular intrachain regions in proteins. *Proc. Natl. Acad. Sci. USA* 70:697–701.

Xu, D., K. Baburaj, C.B. Peterson, and Y. Xu. 2001. Model for the three-dimensional structure of vitronectin: Predictions for the multi-domain protein from threading and docking. *Proteins* 44:312–320.

Xu, D., D. Kim, P. Dam, M. Shah, E.C. Uberbacher, and Y. Xu. 2003. Characterization of protein structure and function at genome scale with a computational prediction pipeline. In *Genetic Engineering, Principles and Methods*, Vol. 25, J.K. Setlow (ed.). New York, Kluwer Academic/Plenum Publishers, pp. 269–293.

Xu, D., M.A. Unseren, Y. Xu, and C. Uberbacher. 2000. Sequence–structure specificity of a knowledge based energy function at the secondary structure level. *Bioinformatics* 16:257–268.

Xu, J., F. Jiao, and B. Berger. 2005. A tree decomposition approach to protein structure prediction. *Proceedings of 2005 IEEE Computational Systems Bioinformatics Conference*, pp. 247–256.

Xu, J., and M. Li. 2003. Assessment of RAPTOR's linear programming approach in CAFASP3. *Proteins* 53(Suppl. 6):579–584.

Xu, J., M. Li, D. Kim, and Y. Xu. 2003a. RAPTOR: Optimal protein threading by linear programming. *J. Bioinform. Comput. Biol.* 1:95–117.

Xu, J., M. Li, G. Lin, D. Kim, and Y. Xu. 2003b. Protein threading by linear programming. *Pac. Symp. Biocomput.* pp. 264–275.

Xu, Y., and E.C. Uberbacher. 1996. A polynomial-time algorithm for a class of protein threading problems. *Comput. Appl. Biosci.* 12:511–517.

Xu, Y., and D. Xu. 2000. Protein threading using PROSPECT: Design and evaluation. *Proteins* 40:343–354.

Xu, Y., D. Xu, O.H. Crawford, and J.R. Einstein. 2000c. A computational method for NMR-constrained protein threading. *J. Comput. Biol.* 7:449–467.

Xu, Y., D. Xu, O.H. Crawford, J.R. Einstein, F. Larimer, E. Uberbacher, M.A. Unseren, and G. Zhang. 1999. Protein threading by PROSPECT: A prediction experiment in CASP3. *Protein Eng.* 12:899–907.

Xu, Y., D. Xu, O.H. Crawford, J.R. Einstein, and E. Serpersu. 2000b. Protein structure determination using protein threading and sparse NMR data. *Annual Conference on Research in Computational Molecular Biology*, pp. 299–307.

Xu, Y., D. Xu, and H.N. Gabow. 2000a. Protein domain decomposition using a graph-theoretic approach. *Bioinformatics* 16:1091–1104.

Xu, Y., D. Xu, and V. Olman. 2002. A practical method for interpretation of threading scores: An application of neural network. *Stat. Sinica.* 12:159–177.

Xu, Y., D. Xu, and E.C. Uberbacher. 1998. An efficient computational method for globally optimal threading. *J. Comput. Biol.* 5:597–614.

Yan, B., C. Pan, V.N. Olman, R.L. Hettich, and Y. Xu. 2005. A graph-theoretic approach for the separation of b and y ions in tandem mass spectra. *Bioinformatics* 21:563–574.

Ye, X., P.K. O'Neil, A.N. Foster, M.J. Gajda, J. Kosinski, M.A. Kurowski, J.M. Bujnicki, A.M. Friedman, and C. Bailey-Kellogg. 2004. Probabilistic cross-link analysis and experiment planning for high-throughput elucidation of protein structure. *Protein Sci.* 13:3298–3313.

Young, M.M., N. Tang, J.C. Hempel, C.M. Oshiro, E.W. Taylor, I.D. Kuntz, B.W. Gibson, and G. Dollinger. 2000. High throughput protein fold identification by using experimental constraints derived from intramolecular cross-links and mass spectrometry. *Proc. Natl. Acad. Sci. USA* 97:5802–5806.

Zhang, B., L. Jaroszewski, L. Rychlewski, and A. Godzik. 1997. Similarities and differences between nonhomologous proteins with similar folds: Evaluation of threading strategies. *Fold. Des.* 2:307–317.

Zhang, C., and C. DeLisi. 1998. Estimating the number of protein folds. *J. Mol. Biol.* 284:1301–1305.

Zhang, Y., and J. Skolnick. 2004. Scoring function for automated assessment of protein structure template quality. *Proteins* 57:702–710.

Zhou, H.Y., and Y.Q. Zhou. 2002. Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. *Protein Sci.* 11:2714–2726.

Zhou, H., and Y. Zhou. 2005. Fold recognition by combining sequence profiles derived from evolution and from depth-dependent structural alignment of fragments. *Proteins* 58:321–328.