J. Mol. Biol. (1996) **255**, 641–665

JMB

Global Optimum Protein Threading with Gapped Alignment and Empirical Pair Score Functions

Richard H. Lathrop[1] and Temple F. Smith[2]*

[1]*Artificial Intelligence Laboratory, Massachusetts Institute of Technology Cambridge, MA 02139, USA*

[2]*BioMolecular Engineering Research Center Boston University, Boston MA 02215, USA*

**Corresponding author*

We describe a branch-and-bound search algorithm for finding the exact global optimum gapped sequence-structure alignment (''threading'') between a protein sequence and a protein core or structural model, using an arbitrary amino acid pair score function (e.g. contact potentials, knowledge-based potentials, potentials of mean force, etc.). The search method imposes minimal conditions on how structural environments are defined or the form of the score function, and allows arbitrary sequence-specific functions for scoring loops and active site residues. Consequently the search method can be used with many different score functions and threading methodologies; this paper illustrates five from the literature. On a desktop workstation running LISP, we have found the global optimum protein sequence-structure alignment in NP-hard search spaces as large as 9.6×10^{31}, at rates ranging as high as 6.8×10^{28} equivalent threadings per second (most of which are pruned before they ever are examined explicitly). Continuing the procedure past the global optimum enumerates successive candidate threadings in monotonically increasing score order. We give efficient algorithms for search space size, uniform random sampling, segment placement probabilities, mean, standard deviation and partition function. The method should prove useful for structure prediction, as well as for critical evaluation of new pair score functions.

Keywords: branch-and-bound search; protein structure prediction; protein folding; amino acid pair potentials; contact potentials

Introduction

The protein folding problem is one of the major challenges confronting molecular biology today. One important approach to this problem uses the known protein crystal structures as folding templates. For example, homologous extension modeling uses primary sequence similarity to guide the alignment of a sequence to a known structure (Sankof & Kruskal, 1983; Greer, 1990). Many evolutionarily unrelated sequences (non-homologs) also contain similar domain folds or structural cores, differing primarily in the surface loops (Orengo *et al*., 1994; Holm & Sander, 1993; Chothia, 1992; Greer, 1990; Richardson, 1981). Recently, approaches have been devised that exploit this fact by aligning a sequence directly to a structure or structural model (''inverse'' protein folding). Protein sequence-structure alignment (''threading'') has a large and readily available literature (among many others, Abagyan *et al*., 1994; Bauer & Beyer, 1994; Bowie *et al*., 1991; Bryant & Lawrence, 1993; Crippen, 1991; Fetrow & Bryant, 1993; Finkelstein & Reva, 1991; Godzik *et al*., 1992; Goldstein *et al*., 1992; Hendlich *et al*., 1990; Johnson *et al*., 1993; Jones *et al*., 1992; Karlin *et al*., 1994; Lüthy *et al*., 1992; Maiorov & Crippen, 1992; Matsuo & Nishikawa, 1994; Miyazawa & Jernigan, 1985; Ouzounis *et al*., 1993; Sippl, 1990, 1993; Sippl & Weitckus, 1992; Wilmanns & Eisenberg, 1993; and for reviews, see Bowie & Eisenberg, 1993; Jones & Thornton, 1993; Wodak & Rooman, 1993; Bryant & Altschul, 1995; Sippl, 1995).

The definition of threading used in this paper follows Greer (1990), Jones *et al*. (1992) and Bryant & Lawrence (1993), as illustrated in Figure 1. Formal analyses are given by White *et al*. (1994), Stultz *et al*. (1995) and Lathrop (1994). The structural model corresponds to an annotated backbone trace of the secondary structure segments in the conserved core

Present address: R. H. Lathrop, Department of Information and Computer Science, University of California, Irvine, CA 92717, USA.

Abbreviations used: SIMD, single instruction multiple data; MIMD, multiple instruction multiple data; 3D, three-dimensional.

0022–2836/96/040641–25 $12.00/0

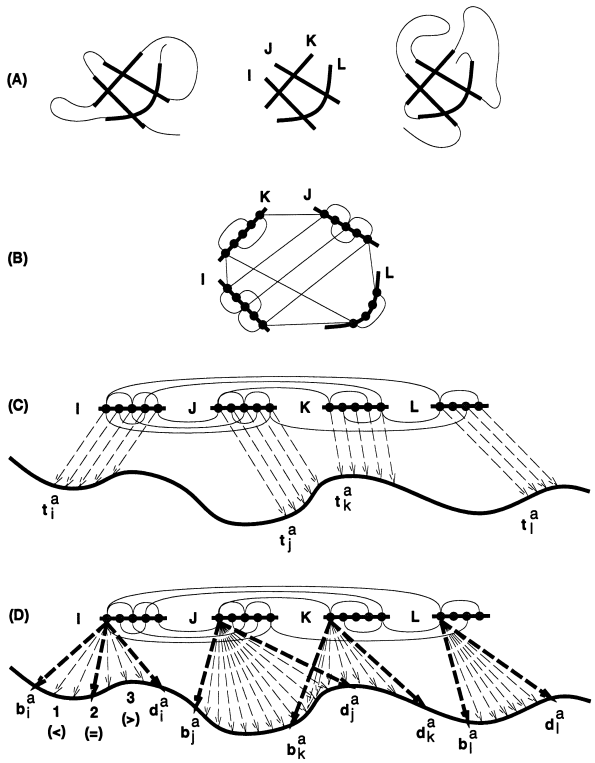

Figure 1. An illustration of the gapped protein threading methodology (Bryant & Lawrence, 1993; Greer, 1990; Jones *et al*., 1992) used in this work. (A) Conceptual drawing of two structurally similar proteins and a common core of four secondary structure segments (dark lines, I-L). To form the structural models used here, side-chains are replaced by a methyl group and loops are removed. (B) Abstract structural model showing spatial adjacencies (interactions). Small circles represent amino acid residue positions (core elements), and thin lines connect neighbors in the folded core. The structural environments and spatially neighboring positions will be recorded for later use by the score function. (C) One possible threading with a novel sequence. A sequence is threaded through the model by placing successive sequence amino acid residues into adjacent core elements. t_x^a indexes the sequence residue placed into the first element of segment X. Sequence regions between core segments become connecting turns or loops. (D) Defining and splitting sets of threadings. Sets used in the branch-and-bound search are defined by lower and upper limits (dark arrows, labeled b_x^a and d_x^a for segment X) on the sequence amino acid residue placed into the first core element of each segment. The set consists of all legal threadings such that the first element of each segment X is within the interval $[b_x^a, d_x^a]$. A set is split into subsets by choosing one core segment (here, segment I) and one split point (dark interior arrow). Its interval is split into sub-intervals: (1) less than; (2) equal to; and (3) greater than the split point.

fold. Core segments are connected by variable loop or coil regions. Loops are not considered part of the conserved fold, and are modeled by an arbitrary sequence-specific loop score function. The model's primitive core elements correspond to spatial locations that eventually will be occupied by sequence amino acid residues. Depending upon the requirements of the particular theory of protein structure adopted, the structural model may record local structural environments, spatial neighbors, degree of solvent exposure, distances between core elements, and so on. In this way, the annotated structural model organizes its core elements: each is embedded in an implied structural environment and interacts with structurally implied neighbors.

When a sequence is threaded through the structural model, successive core elements of each segment are occupied by adjacent amino acid residues from the sequence. Alignment gaps are confined to the connecting non-core loop regions (termed ''gapped alignment'' by Bryant & Lawrence, 1993), and so the loop lengths are variable. This gives rise to an exponentially large search space of possible threadings (alignments between sequence and structure). Each distinct threading is assigned a score by an assumed score function (e.g. contact potentials, knowledge-based potentials, potentials of mean force, etc.). We restrict attention to score functions that can be computed by considering no more than two core segments at a time.

Searching for the best threadings

A given sequence is threaded through a given structure by searching for a sequence-structure alignment that places sequence amino acid residues into preferred structural environments and near other preferred amino acid types. The two key conditions that determine the complexity of this search (Lathrop, 1994) are whether (1) variable-length gaps are admitted into the alignment, and (2) interactions between neighboring amino acid residues from the sequence being threaded are admitted into the score function.

If variable-length gaps are not permitted (Crippen, 1991; Hendlich *et al*., 1990; Maiorov & Crippen, 1992; Sippl, 1990; Sippl & Weitckus, 1992) then alignments are restricted to substructures of equal length that are extracted from a database. In a predictive setting, ignoring variable-length gaps means that the structure and a novel sequence almost invariably will be partially out of hydrophobic registration (Novotný *et al*., 1988).

Variable-length gaps may be permitted while interactions between amino acid residues from the threaded sequence are not allowed. Here, amino acid interactions may be ignored altogether and only the local environment considered (Bowie *et al*., 1991; Johnson *et al*., 1993; Lüthy *et al*., 1992); interactions may be assigned to generic bulk peptide instead of to specific amino acid types (Ouzounis *et al*., 1993); or interactions may be evaluated with respect to the structure's original native sequence instead of the sequence actually being threaded (Sippl, 1993; Wilmanns & Eisenberg, 1993). In these cases, the global optimum threading can be found using the dynamic programming alignment method (Sankof & Kruskal, 1983).

Dynamic programming alignment employs an affine gap penalty that biases the search to prefer loop lengths present in the model structure's original sequence, and so would make distant structural homologs more difficult to recognize if their loop lengths differed substantially (Russell & Barton, 1994). Additionally, ignoring amino acid interactions means giving up a potentially rich source of structural information.

Alternatively, if both variable-length gaps and interactions between neighboring amino acid residues are allowed, then finding the global optimum threading is NP-hard (Lathrop, 1994). This means that in order to find an optimal solution, any known algorithm must require an amount of time that in the worst case is exponential in protein size. Consequently, any current search algorithm must adopt one of two choices: (1) it may find the optimal solution in many cases and very good solutions in others, but sometimes must fail to find the optimal; or (2) it may terminate rapidly in many cases, but sometimes must require an exponential amount of time.

Several researchers have adopted the first choice. Most modify the dynamic programming alignment method to yield an approximate solution in polynomial time; if an affine gap penalty is employed, of course, the search will be biased to favor the model structure's original loop lengths. Godzik *et al*. (1992) substitute the original motif residues, or previous aligned sequence residues in subsequent iterative steps, for the neighbors (their ''frozen approximation''). Finkelstein & Reva (1991) and Goldstein *et al*. (1992) also use this iterative approach. Jones *et al*. (1992) use a modified dynamic programming routine from Taylor & Orengo (1989, see also Orengo & Taylor, 1990), which employs a secondary level of dynamic programming to fix the neighbors for the first level. A Monte Carlo search, not based on dynamic programming, has been used (Bryant & Altschul, 1995). In general, all these methods find a good but approximate solution rather than the optimal one.

Others have adopted the second choice, which guarantees to find the global optimum threading while allowing both variable-length gaps and pairwise interactions. Here, the only other work that we are aware of is that of Bryant & Lawrence (1993) and colleagues (Bryant & Altschul, 1995). They exhaustively enumerated all legal threadings, and reported 5×10^5 evaluations per hour ($=1.4 \times 10^2$ per second) on a Silicon Graphics 4D-35 workstation. In order to make the search practical, bounds were placed on the loop lengths considered based on observed loop lengths in aligned homologous sequences. This will miss less homologous sequences having more divergent loops (it is possible to remove loops of up to 140 amino acid residues from some proteins and retain specific activity; Starzyk *et al*., 1987). In any case, exhaustive search rapidly becomes impractical for larger proteins and more diverse families.

Finding the global optimum

We describe the first practical method of finding the mathematically exact global optimum threading when both variable-length gaps and pairwise interactions are allowed. Given a fixed structural model, sequence and score function, our branch-and-bound search algorithm (Winston, 1993; Kumar, 1992) is guaranteed to find the optimal threading first, and thereafter to enumerate successive candidate threadings in score order. It provides a mathematically exact implementation for the ''gapped alignment'' threading methodology indicated in Figure 1 (e.g. see Jones *et al*., 1992; Bryant & Lawrence, 1993; White *et al*., 1994). Here, (1) specific pairwise amino acid interactions are confined to the structural model; (2) loops are scored by an arbitrary sequence-specific function; and (3) alignment gaps are prohibited within modeled secondary structure segments.

As much as possible, the formulation below deliberately isolates the search method from any particular theory of protein structure, from the way structural environments are defined, and from the score function employed. Consequently the search method applies to a wide variety of score functions that utilize pairwise amino acid interactions. For example, an early search prototype demonstrated the use of a non-statistical rule-based score function (Lathrop & Smith, 1994). Also, we are able to exploit pair score functions originally developed without variable-length gaps (e.g. see Miyazawa & Jernigan, 1985; Maiorov & Crippen, 1992; Sippl, 1990, 1993) by using (1) their exact definitions for the structural models, and (2) an auxiliary loop score function for the loop regions. Below, we consider five pair score functions from the literature: Bryant & Lawrence (1993), Maiorov & Crippen (1992), Miyazawa & Jernigan (1985), Sippl (1990, 1993) and White *et al*. (1994).

It is important to understand the results below from a rigorously formal perspective. In any threading trial, the input sequence, structural model and score function exactly define an abstract mathematical space. Each point in this search space corresponds one-to-one with a distinct alignment between the sequence and the structure. The score function assigns a scalar value (a score or pseudo-energy) to each point. The global minimum score on the resulting pseudo-energy landscape is the lowest score achieved by any point in the space. The global optimum alignment(s) is exactly the point(s) that achieves the global minimum score. These are well-defined objects of independent mathematical interest. They are fixed, in an exact mathematical sense, once the input sequence, structural model and score function are known. The particular values of the global minimum and the best alignment, therefore, are a function only of the input; while our ability to identify them is a function of the search algorithm.

Results

This section is organized into two parts. The first part presents the branch-and-bound search algorithm's computational behavior and current limits. The second presents biological examples selected to illustrate characteristic performance strengths and weaknesses in current threading score functions and structural models. These are exposed here more clearly than in previous studies because the global optimum eliminates search approximation error. In both parts we have chosen a single score function with which to work detailed examples, and have used all five score functions to illustrate general trends. Every example described has been run under all five score functions employed, and yields the same qualitative behavior (often with substantial variation in detail).

Two of the five score functions shown below (Bryant & Lawrence, 1993; White *et al*., 1994) directly provide loop (or loop reference state) score terms as part of their score function. The other three (Miyazawa & Jernigan, 1985; Maiorov & Crippen, 1992; Sippl, 1990, 1993) here require an auxiliary loop score function. This was set to zero for our timing analysis, which therefore depends only on previously published values or theories. For the biological examples we set it proportional to a negative log odds ratio, $-\log(P(a|\text{loop})/P(a))$, summed over all amino acid residues a in the loop. Here $P(a)$ is the prior probability of a and $P(a|\text{loop})$ is the probability of observing a in a loop region.

Computational resources

This section shows that the search can succeed in many practical cases, and illustrates the relationship between problem size and computational resources required. Detailed computational analyses are based on the score function of Bryant & Lawrence (1993), because it has the highest convergence rate found (99.8%) and thus gives a picture of performance spanning 30 orders of magnitude in search space size ($<10^1$ to $>10^{31}$).

Structural model library across computational trials

We developed a library of core structural models taken from 58 non-homologous, monomeric, single-domain, soluble, globular proteins representing diverse structure types (described in Table 1). We believe this to be one of the simplest interesting test cases: statistical artifacts arising from much smaller test sets are avoided, and the proteins require no arbitrary decisions about hydrophobic face packing on domain or multimer boundaries. In order to avoid any subjective bias in core definition, core segments were exactly the main-chain plus β-carbon atoms (inferred for glycine) of α-helices and β-strands taken from the Brookhaven Protein Data Bank feature tables (Bernstein *et al*., 1977), or computed from atomic coordinates using DSSP (Kabsch & Sander, 1983; smoothed as described by Stultz *et al*., 1995) if not present. All side-chains were replaced by alanine, in order to assign structural model environments independent of the original amino acid identities. Loops were then discarded. The resulting structural models were equivalent to a backbone trace plus β-carbon atoms of the core secondary structure, annotated as required by the score function. We sought to reduce residual traces of the structure's original primary sequence (sequence memory) and loop lengths (gap memory), as otherwise threading alignment accuracy on distant structural homologs may suffer (see discussions by Ouzonis *et al*., 1993; Russell & Barton, 1994; Rost & Sander, 1994).

We exhaustively threaded every library sequence through every library structural model. This created 3364 sequence-model pairs, each consisting of a single fixed sequence and model. Model loops assigned length zero or 1 by the crystallographer were treated as fixed-length because they usually reflect constrained ''kinks'' in the secondary structure. In all other cases we considered all physically realizable loop lengths that maintained core segment topological order. Any loop length that could be proven to break the main chain or violate excluded atomic volumes was discarded as illegal. Consequently, 833 sequence-model pairs were discarded *a priori* because the sequence was too short to occupy the model under any legal loop assignment. With the remaining 2531 admissible pairs we searched for the global optimum threading under all five score functions considered. This resulted in a total of 12,655 legal trials, where each trial corresponded to a search for the global optimum threading given a fixed sequence, structural model, and score function. Trials were run on a desktop workstation DEC Alpha 3000-M8000, using public-domain CMU Common Lisp (MacLachlan, 1992), and were terminated at our computational limit of two hours. For each trial, we computed the size of the search space of legal threadings and recorded the elapsed time required to find its global optimum threading.

Problem size and computation time

In a total of 12,109 trials (96%) the search converged within two hours; in 488 trials (4%) time was exhausted first; and in 58 trials (0.5%) space was exhausted first. Figure 2 shows the time required to find the global optimum in every convergent trial under all five score functions, as a function of search space size. On a DEC Alpha 3000-M8000 desktop workstation running LISP (Steele, 1990), we have identified the global optimum threading in NP-hard search spaces as large as 9.6×10^{31} at rates ranging as high as 6.8×10^{28} equivalent threadings per second, most of which were pruned before they were ever explicitly examined.

Table 1 shows detailed timing results for self-threading each sequence onto its own core

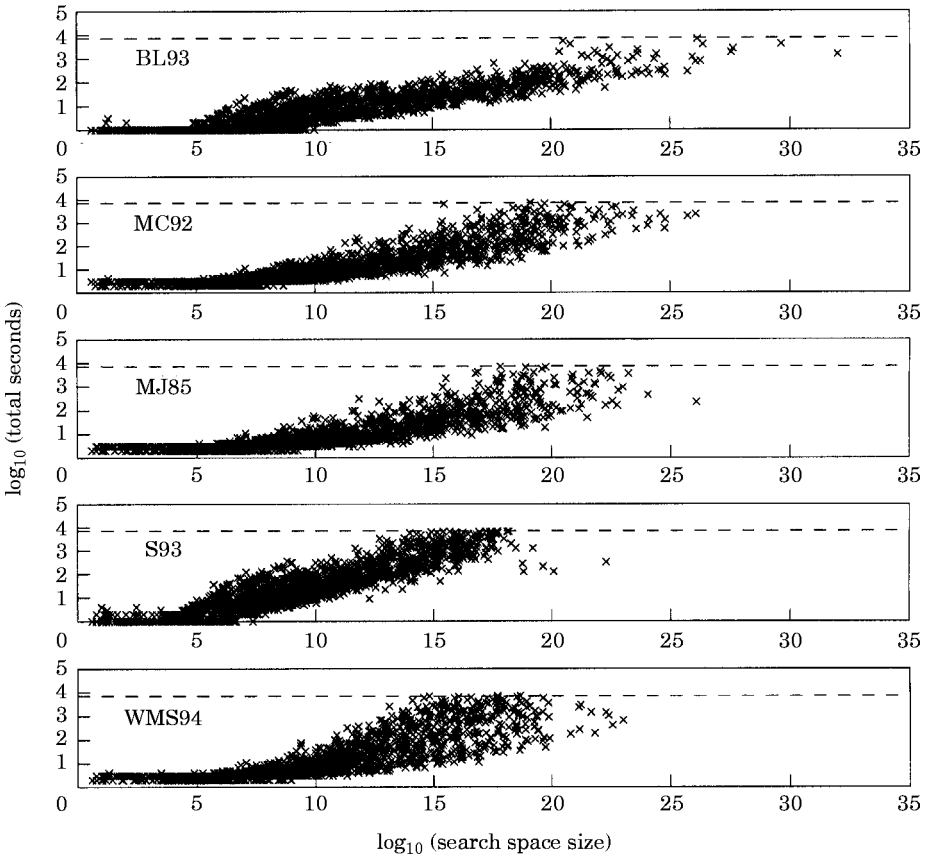

Figure 2. The time required to find the global minimum is shown on log-log axes as a function of search space size. All sequences and all structural models in our library were threaded through each other under every score function considered. The graph for each score function shows every trial that converged under that score function. PDB codes are shown in Table 1. Score functions: BL93, Bryant & Lawrence (1993); MC92, Maiorov & Crippen (1992); MJ85, Miyazawa & Jernigan (1985); S93, Sippl (1990, 1993); WMS94, White *et al*. (1994). Timing resolution is one second. The broken line corresponds to our computational limit of two hours. All physically realizable loop lengths were admitted, but gaps provably breaking the chain or violating excluded atomic volumes were prohibited. Occasionally the crystallographer assigns a loop length of zero or 1; this usually reflects a constrained ''kink'' in secondary structure, not a true loop, and was left unchanged. Trials were performed using CMU Common Lisp (MacLachlan, 1992) running on a DEC Alpha 3000-M8000 desktop workstation.

structural model. Protein size is stated in terms of sequence length and number of core segments; search space growth is exponential in number of core segments, but in practice proteins are roughly one-half secondary structure and so the two measures are roughly proportional. Total elapsed time is resolved into initialization and search components, showing that the fast search does not require a prohibitively long initialization. Table 1 may be cross-indexed to Figure 2, BL93, by $x = \log_{10}$(Search Space Size) and $y = \log_{10}$(total seconds).

Table 2 shows the fraction of trials that converged in each case, the total and per-trial time required, and the log-log regression slopes and intercepts, across all five score functions used. It compares native and non-native threadings for each graph in Figure 2, and gives the pooled results of all trials. Table 2 summarizes Table 1 in the row labeled BL93 Native.

Figure 3 shows histograms of number of trials and total time expended finding optimal threadings according to Bryant & Lawrence (1993), grouped by search space size. In 81% of all trials, the search space contained fewer than 10^{15} legal threadings. However, the searches in those same trials expended only 11% of the total time. Conversely, only 4% of all trials involved a search space that contained more than 10^{20} threadings, but their searches expended 71% of the total time. Figure 3 corresponds to Figure 2, BL93.

Biological examples

The examples here illustrate the promise and remaining challenges of protein threading. They span the full range of shared evolutionary history: from self-threading, through ancient homologs, to unrelated structural analogs. Self-threading illustrates effects due to structural environment similarity and propagated pairwise interactions. Homologous extension illustrates the effects of multimeric interfaces, model length and active sites. All five score functions both succeed and fail when

Table 1. Timing details for self-threading (Bryant & Lawrence (1993), on DEC Alpha in LISP)

Protein number	PDB code	Protein length	Number of core segments	Search Space Size	Number of search iterations	Total (search-only) seconds	Equivalent threadings per iteration	Equivalent threadings per second
1	256b	106	5	6.19e + 3	6	1 (1)	1.03e + 3	6.19e + 3
2	1end	137	3	4.79e + 4	6	1 (1)	7.98e + 3	4.79e + 4
3	1rcb	129	4	5.89e + 4	7	1 (1)	8.41e + 3	5.89e + 4
4	2mhr	118	4	9.14e + 4	7	1 (1)	1.31e + 4	9.14e + 4
5	351c	82	4	1.12e + 5	5	1 (1)	2.24e + 4	1.12e + 5
6	1bgc	174	4	1.63e + 5	6	1 (1)	2.72e + 4	1.63e + 5
7	1ubq	76	5	1.70e + 5	6	1 (1)	2.83e + 4	1.70e + 5
8	1mbd	153	8	1.77e + 5	10	1 (1)	1.77e + 4	1.77e + 5
9	1lis	136	5	5.02e + 5	7	1 (1)	7.17e + 4	5.02e + 5
10	1aep	161	5	5.76e + 5	13	1 (1)	4.43e + 4	5.78e + 5
11	1hoe	74	6	7.36e + 5	8	1 (1)	9.20e + 4	7.36e + 5
12	2hpr	87	6	1.34e + 6	8	1 (1)	1.68e + 5	1.34e + 6
13	5cyt	103	5	1.37e + 6	8	1 (1)	1.71e + 5	1.37e + 6
14	1bp2	123	5	1.53e + 6	8	1 (1)	1.92e + 5	1.53e + 6
15	1aba	87	7	1.95e + 6	13	1 (1)	1.50e + 5	1.95e + 6
16	1cew	108	6	2.32e + 6	8	1 (1)	2.91e + 5	2.32e + 6
17	5cpv	108	5	2.80e + 6	6	1 (1)	4.33e + 5	2.60e + 6
18	2mcm	112	10	1.31e + 7	15	1 (1)	8.75e + 5	1.31e + 7
19	5fd1	106	5	2.25e + 7	12	1 (1)	1.88e + 6	2.25e + 7
20	1plc	99	6	3.63e + 7	10	1 (1)	3.63e + 6	3.63e + 7
21	1alc	123	6	1.70e + 8	10	2 (1)	1.70e + 7	8.51e + 7
22	1yat	113	7	2.03e + 8	8	1 (1)	2.54e + 7	2.03e + 8
23	7rsa	124	10	2.54e + 8	12	1 (1)	2.12e + 7	2.54e + 8
24	3fxn	138	9	7.09e + 8	12	2 (1)	5.91e + 7	3.54e + 8
25	9rnt	104	8	7.53e + 8	21	2 (1)	3.58e + 7	3.76e + 8
26	2sns	149	8	2.19e + 9	14	4 (1)	1.58e + 8	5.47e + 8
27	1ifc	132	12	2.31e + 9	87	2 (1)	2.66e + 7	1.16e + 9
28	2lzm	164	12	3.16e + 9	37	2 (1)	8.54e + 7	1.58e + 9
29	3chy	128	10	4.08e + 9	45	1 (1)	9.06e + 7	4.08e + 9
30	1pkp	150	9	5.32e + 9	20	3 (1)	2.66e + 8	1.77e + 9
31	1aak	152	8	2.34e + 10	10	3 (1)	2.34e + 9	7.82e + 9
32	8dfr	189	10	1.45e + 11	25	7 (1)	5.78e + 9	2.06e + 10
33	1cde	212	13	1.51e + 11	38	5 (1)	3.99e + 9	3.03e + 10
34	2cp1	165	10	1.82e + 11	17	5 (1)	1.07e + 10	3.65e + 10
35	3adk	194	13	1.89e + 12	66	3 (1)	2.86e + 10	6.30e + 11
36	1rec	201	10	3.54e + 12	30	4 (1)	1.18e + 11	8.85e + 11
37	2cyp	294	10	3.55e + 12	181	20 (4)	1.96e + 10	1.78e + 11
38	1f3g	161	16	5.17e + 12	45	6 (1)	1.15e + 11	8.61e + 11
39	4fgf	146	12	1.06e + 13	48	4 (1)	2.22e + 11	2.66e + 12
40	1baa	243	9	1.53e + 13	64	10 (2)	2.39e + 11	1.53e + 12
41	2act	220	11	1.12e + 14	34	7 (1)	3.30e + 12	1.60e + 13
42	1dhr	241	14	4.56e + 14	51	5 (1)	8.94e + 12	9.12e + 13
43	1mat	264	11	5.25e + 14	100	15 (2)	5.25e + 12	3.50e + 13
44	1tie	172	12	1.19e + 15	394	20 (9)	3.03e + 12	5.96e + 13
45	3est	240	13	1.92e + 15	1946	47 (36)	9.85e + 11	4.08e + 13
46	2ca2	259	10	4.51e + 15	100	20 (2)	4.51e + 13	2.25e + 14
47	1byh	214	14	1.07e + 16	95	12 (4)	1.12e + 14	8.90e + 14
48	1apa	266	14	3.56e + 17	141	18 (6)	2.52e + 15	1.98e + 16
49	4tgl	269	14	5.86e + 18	361	22 (7)	1.62e + 16	2.66e + 17
50	5tmn	316	14	6.51e + 18	164	28 (7)	3.97e + 16	2.32e + 17
51	1lec	242	15	7.01e + 18	320	26 (12)	2.19e + 16	2.70e + 17
52	1nar	290	17	2.33e + 19	3984	208 (183)	5.85e + 15	1.12e + 17
53	1s0l	275	15	4.36e + 19	541	32 (13)	8.05e + 16	1.36e + 18
54	5cpa	307	16	1.22e + 20	1089	72 (50)	1.12e + 17	1.69e + 18
55	9api	384	17	1.95e + 22	290	57 (25)	6.71e + 19	3.41e + 20
56	2had	310	19	2.57e + 22	4027	201 (179)	6.39e + 18	1.28e + 20
57	2cpp	414	20	6.37e + 24	3068	205 (164)	2.08e + 21	3.11e + 22
58	6taa	478	23	9.63e + 31	4917	1409 (1267)	1.96e + 28	6.83e + 28

Detailed data for all self-threading cases in Figure 2, BL93. Experimental conditions are as described for Figure 2. PDB code is the locus name in the Brookhaven Protein Data Bank (Bernstein *et al*., 1977). Search space size is the size of the search space within which the algorithm finds the optimal threading. Note that the algorithm does not actually calculate all of these threadings, which is one of its critical strengths. Total (search-only) seconds is total (in parentheses, only the search component) real elapsed clock time. Total time includes reading the protein sequence and core structure from files, all datastructure initialization, and the search itself, but not reading the score function parameter files (in a predictive setting these would be memory-resident). Number of search iterations is the number of times the loop labeled Iteration in Methods was executed. Equivalent threadings per second and Equivalent threadings per iteration are the ratio of search space size to the respective quantities. The ratios represent, respectively, the speed required for exhaustive search to achieve the same time, and the number of threadings that exhaustive search would examine per iteration of pruned search.

Table 2. Convergence rates, total hours, slopes and intercepts

Potentials	Native or non-native	Searches (%)	Converged	Total hours	Avg. seconds per search	Regression	
						Slope	Interc.
BL93	Native	100	58/58	0.7	43.3	0.12	–0.74
	Non-native	99.8	2467/2473	42.3	61.5	0.13	–0.47
MC92	Native	98	57/58	3.2	199.1	0.13	–0.50
	Non-native	98	2426/2473	167.5	243.8	0.14	–0.44
MJ85	Native	98	57/58	3.3	204.5	0.12	–0.42
	Non-native	99	2446/2473	101.6	148.0	0.13	–0.35
S93	Native	90	52/58	13.7	853.1	0.16	–0.43
	Non-native	89	2189/2473	749.7	1091.4	0.24	–0.81
WMS94	Native	88	51/58	18.4	1143.7	0.18	–0.74
	Non-native	93	2306/2473	455.6	663.2	0.18	–0.62
Pooled	Pooled	96	12,109/12,655	1556.0	442.6	0.15	–0.40

Experimental conditions are as described for Figure 2. Native refers to threading sequences onto their native cores, Non-native onto non-native cores. Total hours is the total time expended by all searches. Searches that did not converge expended two hours before being terminated. Slope and Interc. are respectively the slope and intercept (a and b in $y = ax + b$) of the best fitting regression line to the graphs in Figure 2, discarding points at $y = 0$. Pooled accumulates the results of all trials.

threading unrelated structural analogs. These examples show strengths and weaknesses of current protein threading. In general, the errors seen are those one would expect based on current understanding of protein structure.

Our detailed biological analyses use the score function of White *et al*. (1994; parameters given by Stultz *et al*., 1995), based on Markov Random Field theory, because (1) it rests on well-understood mathematics, and (2) we were able to generate completely cross-validated score function parameters for all protein structures studied (i.e. excluding the protein tested and any homologs from the set of proteins used to develop parameters; for the other score functions we used literature values, and so they are only partially cross-validated for our test set).

Characteristic self-threading behaviors

Self-threading a sequence through its own structural model is an exercise in which alignment

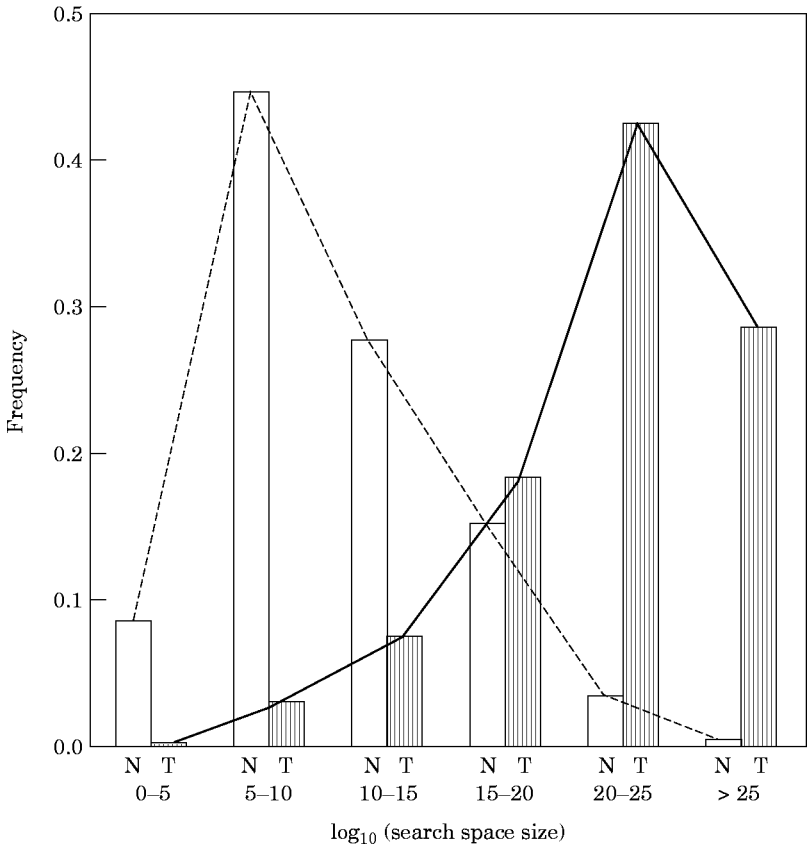

Figure 3. Histograms of number of trials (N, white bars, dotted line) and total time expended (T, striped bars, continuous line), grouped by search space size. Trials and search space sizes reflect the 2531 legal sequence-model pairs that result from threading every sequence through every structural model in our library (PDB codes are given in Table 1). Time expended reflects trials shown in Figure 2, BL93; each non-convergent trial expended two hours. The score function used is that described by Bryant & Lawrence (1993). The histograms group trials according to $\log_{10}$ (search space size). For example, 0–5 indicates the trials such that the search space size is between 10^0 and 10^5, N indicates the fraction of total trials having search space sizes in that range, and T indicates the fraction of the total time that was expended on them.

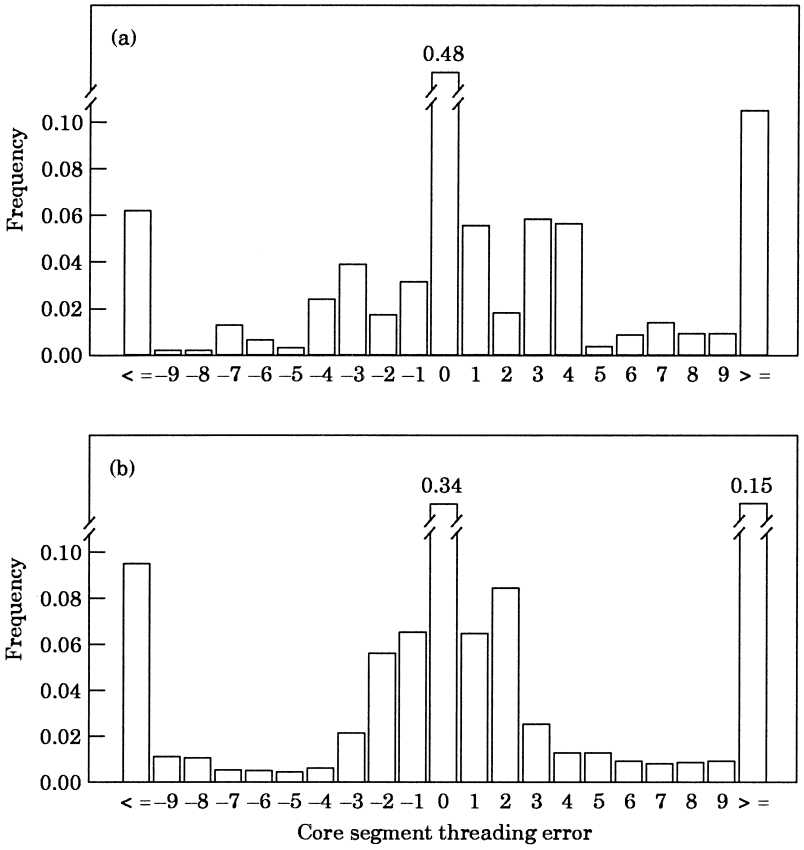

Figure 4. Alignment errors between the optimal and the native threadings in Figure 2 for each core segment, across all five score functions and all convergent self-threading trials. Error is computed as the optimal threading sequence index minus the native sequence index. (a) Histogram of errors from 1137 α-helix core segment threadings. (b) Histogram of errors from 1488 β-strand core segment threadings.

errors are unambiguous and certain behaviors are exposed very clearly. Accurate self-threading across a library of diverse structure types is a challenging task for any current score function when a predictive setting is emulated: (1) side-chains are removed to reduce sequence memory in the structural model (contrast using the original amino acid residues to define structural environments, neighbors, or score function parameters); (2) the score function does not favor loop lengths from the model's original sequence (contrast a dynamic programming gap penalty that penalizes other loop lengths); and (3) the protein and all homologs are excluded from the data set used to estimate score function parameters (contrast uncross-validated tests, which partially ''memorize'' the training set).

Short amphipathic shifts arise when an individual core segment may be aligned to any of several similar nearby local environments. They often maintain local hydrophobic registration. We measured the displacement between the optimal and the native threadings for each core segment shown in Figure 2, across all five score functions and all self-threading trials that converged. Error distributions from 1137 α-helix and 1488 β-strand core segment threadings are shown in Figure 4. The distributions reflect the amphipathic periodicity of secondary structure.

The amino acid residues native to one α-helix or β-strand often are threaded into the structural positions of an adjacent α-helix or β-strand of similar length and amphipathic character. This may involve the sequential offset of many segments. Figure 5(a1) shows an all-β protein (Eriksson *et al*., 1993; PDB code 4FGF) in which segments 1 through 7 in the native alignment were shifted to segments 2 through 8 in the optimal alignment. This may have occurred because segment 1 was misaligned by the score function (for whatever reason), and the adjacent β-strands occupied adjacent locations in the structure that were, perhaps, the next most favorable given the misalignment of segment 1. This was seen by repeating the threading with segment 1 constrained to its native alignment. The score function was modified to assign $+\infty$ (effectively, ''infinitely bad'') to all other locations of segment 1. This immediately pruned all threadings that would place segment 1 at a non-native sequence location, while leaving the rest of the search space unaffected. The result is shown in Figure 5(a2). When segment 1 was correctly aligned, all other long-range displacements were eliminated. Only short amphipathic shifts remained.

It is much harder to identify errors due to pairwise amino acid score terms rather than average local structural environment, because pairwise score contributions are distributed across many non-local interactions. Figure 5(b) shows an all-α protein (Finzel *et al*., 1984; PDB code 2CYP) with one α-helix (segment 4) displaced by two residues,

which would reverse its amphipathic nature. However, this core segment interacts with segments 1, 2 and 6, which were displaced by amphipathic shifts as in Figure 4. It is likely that these three segments propagated their errors to segment 4 across pairwise interactions. This was seen by comparing the components of the score for segment 4 between its native and optimal threadings, while holding all other segments fixed at the optimal threading. In seeking a minimum score, the score components attributable to local structural environment (core element plus loop scores) slightly favored the native placement (167.7 native to 167.8 optimal). The pairwise interactions within segment 4 also slightly favored the native (–0.4 to 0.2), as did the pairwise interactions it made with the correctly placed segments 3 and 10 (0.1 to 0.4). However, its pairwise interactions with the misplaced segments 1, 2 and 6, strongly favored the optimal over the native (3.1 native to –2.2 optimal). The final contributions to the placement of segment 4 in the sequence (170.5 native to 166.2 optimal) were dominated by its pairwise interactions with other misplaced segments.

Aligning ancient homologs

Several interesting problems arise when threading is applied to homologous extension modeling in cases where very little primary sequence similarity

(a) Fibroblast Growth Factor
(Eriksson et al, 1993; PDB code 4FGF)

Search Space Size 1.06e+13

Seg-Num	1	2	3	4	5	6	7	8	9	10	11	12
Type	E	E	E	E	E	E	E	E	E	E	E	E
Length	4	6	5	7	7	6	7	7	7	6	3	5

(a1) No constraints.

Native	22	30	39	52	62	72	80	93	103	114	123	138
Optimal	3	22	30	39	53	61	68	81	90	116	135	142
Error	-19	-8	-9	-13	-9	-11	-12	-12	-13	2	12	4

(a2) Segment one (only) constrained to native alignment.

Native	22	30	39	52	62	72	80	93	103	114	123	138
Optimal	22	30	39	50	60	68	80	95	103	114	125	138
Error	0	0	0	-2	-2	-4	0	2	0	0	2	0

(b) Cytochrome C Peroxidase
(Finzel et al, 1984; PDB code 2CYP)

Search Space Size 3.55e+12

Seg-Num	1	2	3	4	5	6	7	8	9	10
Type	H	H	H	H	H	H	H	H	H	H
Length	19	13	16	17	9	14	9	9	13	18

Native	15	42	84	103	150	164	201	232	241	255
Optimal	18	39	84	105	150	165	202	232	241	255
Error	3	-3	0	2	0	1	1	0	0	0

Interaction Matrix

1)	98	1	10	12	0	0	0	0	0	0
2)		62	13	18	0	5	0	0	0	1
3)			80	4	0	0	0	0	0	0
4)				86	0	1	0	0	0	3
5)					38	3	0	6	2	0
6)						68	4	4	10	13
7)							38	3	7	0
8)								38	2	0
9)									62	1
10)										92

Figure 5. Two example threadings illustrating characteristic errors are shown. (a1) The sequence of fibroblast growth factor (Eriksson *et al*., 1993; PDB code 4FGF) threaded onto its own structural model. Segments 1 through 7 were displaced in the optimal threading, and occupy the approximate native locations of segments 2 through 8. (a2) The threading of (a1) repeated with segment 1 constrained to its native location. (b) Shows the sequence of cytochrome *c* peroxidase (Finzel *et al*., 1984; PDB code 2CYP) threaded onto its own structural model. Segment 4 is displaced by two residues, which would reverse its amphipathic nature. Seg-Num indexes successive core segments. Type is H for α-helix and E for β-strand (extended). Length is in amino acid residues. Native is the native threading. Optimal is the global optimum threading. Error is optimal threading coordinate minus native. Interaction Matrix is the number of pairwise interactions between each pair of core segments; because the matrix is symmetric, only the top half is shown. For example, reading down then over, segment 4 interacts with segments 1 (12 interactions), segment 2 (18), segment 3 (4), internally within itself (86), segment 6 (1) and segment 10 (3). The score function used is that described by White *et al*. (1994; parameters as given by Stultz *et al*., 1995). Structural environments were defined by (1) two secondary structure types (α-helix or β-strand); (2) two solvent exposure levels (above or below a 30% cutoff based on Eisenberg & McLachlan (1986) exposure values, with alanine substituted for all side-chains, an expanded radius of 2.1 Å for β-carbon atoms, and an expanded water molecule radius of 2.4 Å); and (3) for pairwise environments, whether or not the pair came from the same or different secondary structures (core segments). Pairwise neighbors were defined by (1) the distance between β-carbon atoms (less than 7.2 Å), and (2) the dot product of the two α-carbon to β-carbon vectors (positive if within the same α-helix or β-sheet, negative if within different secondary structures, and the equivalent for the special case of the beta-barrel).

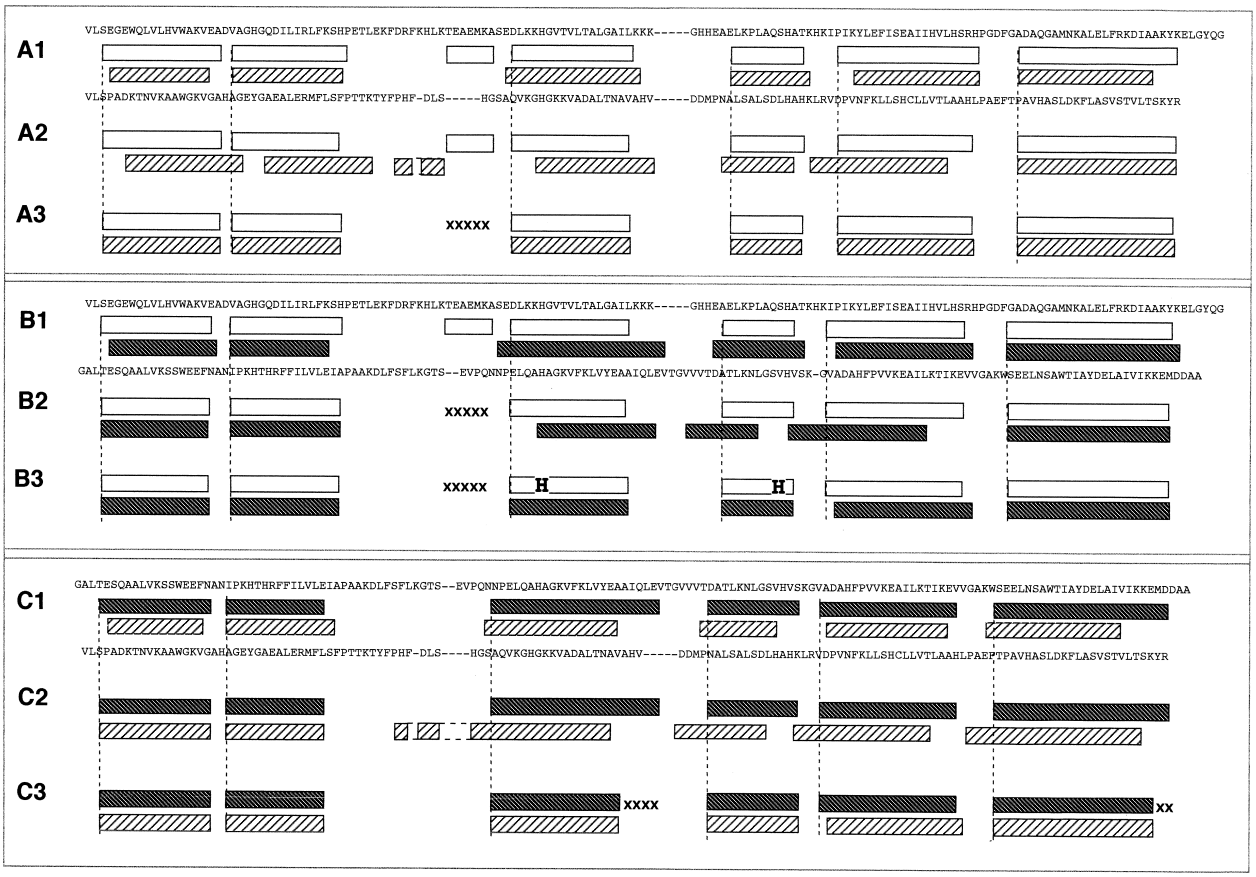

Figure 6. Threading ancient homologs, hemoglobin α-chain, myoglobin and leghemoglobin. Example optimal threadings of sperm whale myoglobin (Watson, 1969; PDB code 1MBN), yellow lupin root nodule leghemoglobin (Vainshtein *et al*., 1975; PDB code 2LH7), and human hemoglobin α-subunit (Kavanaugh *et al*., 1992; PDB code 1DXT, chain A). (a) α-Hemoglobin sequence threaded through the myoglobin structural model. (a1) Reference alignment. (a2) Short ''D'' helix present in model. (a3) Short ''D'' helix removed from model. (b) Leghemoglobin sequence threaded through the myoglobin structural model. (b1) Reference alignment. (b2) Short ''D'' helix removed, no active site constraints. (b3) Short ''D'' helix removed, requiring model elements at positions E7 and F8 to be occupied by histidine, H, from the sequence. (c) α-Hemoglobin sequence threaded through the leghemoglobin structural model. (c1) Reference alignment. (c2) PDB-DSSP-defined structural model. (c3) Removing from the model the elements (indicated by X) that correspond to a gap in the sequence described by Bashford *et al*. (1987). Each part subfigure shows a reference sequence alignment from Bashford *et al*. (1987), model core secondary structure as consistent with DSSP (Kabsch & Sander, 1983), and the core segment placements of the optimal threading. Vertical broken lines indicate the reference sequence-structure alignment taken from Bashford *et al*. (1987). White boxes indicate myoglobin, light boxes indicate α-hemoglobin, dark boxes indicate leghemoglobin. The score function used is from White *et al*. (1994; parameters as given by Stultz *et al*., 1995), as described for Figure 5.

remains. These include hydrophobic mismatch at multimeric interfaces, the presence of active-site residues in unusual structural environments, and secondary structure length mismatch.

The globins are a well studied case (among other threading studies, compare Bowie *et al*., 1991; Bryant & Lawrence, 1993; Godzik *et al*., 1992; Hendlich *et al*., 1990; Jones *et al*., 1992; Sippl & Weitckus, 1992) in which a common structure has been conserved while the amino acid sequence has diverged to the point of unrecognizability between some family members. Evolutionary conservation is seen in the common structure, in the common heme co-factor, and in the amino acid residues that coordinate the contained iron atom. In myoglobin, hemoglobin and leghemoglobin, the traditional sequence alignment step of modeling by homologous extension fails (Sander & Schneider, 1991) due to the minimal sequence similarity. However, the common heme co-factor and its conserved interacting residues supply sufficient alignment constraints.

We created core structural models for sperm whale myoglobin (Watson, 1969; PDB code 1MBN), yellow lupin root nodule leghemoglobin (Vainshtein *et al*., 1975; PDB code 2LH7), and human hemoglobin α-subunit (Kavanaugh *et al*., 1992; PDB code 1DXT, chain A). Structural models consisted of the crystallographer's secondary structure assignments, excluding the C and D helices. The ends of helices were adjusted to include exactly those residues shown by DSSP (Kabsch & Sander, 1983) to make proper hydrogen bonds. We threaded

all three sequences through all three structural models. Example threadings between hemoglobin and myoglobin, leghemoglobin and myoglobin and hemoglobin and leghemoglobin, are shown in Figure 6.

Hemoglobin is a tetramer while myoglobin and leghemoglobin are monomers. Hence, positions at the subunit interface, which normally are buried and occupied by hydrophobic residues in the hemoglobin tetramer, would be mismodeled as exposed to solvent in the myoglobin core. Figure 6(a1) shows the result of threading the hemoglobin α-subunit sequence through the myoglobin structural model when the D helix is present. All helices but one were misaligned.

In the globin case, at least in part, the multimeric interface problem can be dealt with by excluding the helices involved in the hemoglobin α- and β-subunit tetramer interactions. These are the two short C and D helices. This is readily justified because the crystallographers do not identify these helices in all three globins. Myoglobin lacks a C helix and leghemoglobin lacks a D helix in the crystal structures. They are not truly part of a shared common core structure. Even when identified, they appear not as true α-helices but as short 3_{10} or irregular helices. Figure 6(a2) shows the threading that results after removing the D helix from the myoglobin structural model. The threaded hemoglobin α-subunit amino acid residues were perfectly aligned with their analogous myoglobin structural positions. For all other trials in this section, C and D helices were always discarded from the structural models.

The globins bind a heme ring at the active site near the center of their structure. This produces substantially buried positions that contain polar residues, a statistically unlikely event. Figure 6(b1) shows the result of threading the leghemoglobin sequence through the myoglobin structural model. Half of the helices were misaligned, by four or five residues. In virtually all globins, the heme group is held in place by two histidine residues. These occur at positions E7 in the E helix and F8 in the F helix (Bashford *et al*., 1987). The score function was modified to return $+\infty$ whenever E7 and F8 were not occupied by histidine. This immediately pruned all threadings that would not place the desired amino acid type into those positions, while leaving the rest of the search space unaffected. Figure 6(b2) shows the result. With the additional requirement that positions E7 and F8 must be occupied by histidine, all the helices were aligned correctly except the G helix, which was misaligned by an amphipathic shift of one residue.

The leghemoglobin model E and H helices are substantially longer than the hemoglobin model E and H helices (by five and six residues, respectively). Bashford *et al*. (1987) inserted alignment gaps at the C-terminal ends of both leghemoglobin model helices (four and two gaps, respectively); note that this violates modeling assumption (3) in the section Finding the global optimum (above). When correctly aligned to the leghemoglobin model, the hemoglobin sequence is unable to span the gap between the leghemoglobin E and F helices without breaking the chain, and is too short to fill the leghemoglobin model H helix. Figure 6(c1) shows the result of threading the hemoglobin α-subunit sequence through the leghemoglobin structural model. Four of the six helices were misaligned. We created a reduced leghemoglobin structural model by removing from the E and H helices those core elements that correspond to gaps in Bashford *et al*. (1987). Figure 6(c2) shows the result of threading the hemoglobin sequence through the reduced model. All helices were aligned correctly.

The other threadings not shown exhibit qualitatively similar behavior. Using the reduced leghemoglobin model and requiring positions E7 and F8 to be occupied by histidine, 34 of the 36 unshown helix alignments were correct: the E helix in the myoglobin self-threading found the wrong histidine and was badly misaligned, and the H helix in the α-subunit hemoglobin self-threading was misaligned by an amphipathic shift of four residues. Without the reduced leghemoglobin model and histidine requirement, only 23 of the same 36 were correct. Each of the four other score functions we considered also placed more core segments correctly when using the reduced leghemoglobin model and histidine requirement than when not using them.

Aligning structural analogs

The next generalization beyond homologous extension modeling is to model one protein's structure by a structural analog that has no recognizable shared evolutionary history, but is assumed to share a very similar backbone core structure. Two such proteins are the mouse interleukin 1-β (Treharne *et al*., 1990; PDB code 8I1B) and the *Erythrina caffra* trypsin inhibitor (Onesti *et al*., 1991; PDB code 1TIE). Both have a 12-strand β-barrel structure (a β-trefoil) of very similar size and topology. The two sequences have no recognizable sequence similarity (<20% identity). Their functions, an interleukin and a trypsin inhibitor, do not suggest an evolutionary relationship. This example is one of the nine protein domain superfolds described by Orengo *et al*. (1994) and one of the non-homologous structural families described by Holm & Sander (1993).

Figure 7 shows the optimal threadings of each sequence through a structural model constructed from the other's determined structure. Here again there were two very different results. First, the optimal threadings of the interleukin through the trypsin inhibitor were quite accurate across most of the five studied score functions. However, in the converse case, the optimal threadings of the trypsin inhibitor through the interleukin structural model were quite poor, also across most score functions.

Discussion

It is important to remember that the search algorithm requires as input the sequence, structural model and score function. This input completely determines the entire pseudo-energy landscape, including the global minimum score and the optimal threading(s). Although thereby determined, the landscape features generally are unknown. The sole task performed by the search algorithm is to report the value of the global minimum score, and to identify the global optimum threading(s) that instantiate(s) it. The branch-and-bound search algorithm is unusual in that, in any given case, it either finds the mathematically exact answer or it first exhausts time or space resources. Importantly, it never returns an approximate or inexact result. Of course, even for the same sequence and structural model, different input score functions will produce different landscapes and different global minima. Consequently, any agreement, or lack of it, between optimal and native alignments is a property only of the input, and not of the search algorithm.

Computational resources

It is clear from Table 1 that the algorithm is successful at drastically reducing the portion of search space that actually need be examined. This allows the algorithm to find the optimal threading in vastly larger search spaces than heretofore possible. Figure 2 shows that this behavior is characteristic of a wide variety of score functions. Larger search spaces do require more time, as expected, but in most cases examined an exact search could be accomplished within reasonable limits.

Because the algorithm search behavior depends on the input score function, sequence and structural model, it is difficult to give an analytic statement of its average-case time complexity. Figure 2 shows that the log-log relationships remain approximately linear over nearly 25 orders of magnitude. For the range of data considered, the regression slopes in Table 2 are between 0.12 and 0.24, and the intercepts are between –1.16 and –0.39. Changes in raw speed due to different computer languages or models should affect the intercept but leave the slope unchanged, producing a constant vertical offset in Figure 2. Because $\log y = a \log x + b$ implies $y = e^b x^a$, the search time in seconds was approximately proportional to between the eighth and fourth root of the search space size and the proportionality constant was between 0.3 and 0.7. Differences in the underlying search space landscapes probably give rise to the considerable scatter about the central tendency and the variation in timing behavior between score functions. We expect that further speed increases can be achieved by parallelizing the algorithm. It has efficient implementation strategies for both SIMD and MIMD parallel computers. Tighter lower bound formulae (see equation (5)) would decrease the slope in Figure 2, which would lead to greater leverage in larger search spaces.

Due to the exponential nature of the search, most of the time is expended during the few trials that search the very largest search spaces. However,

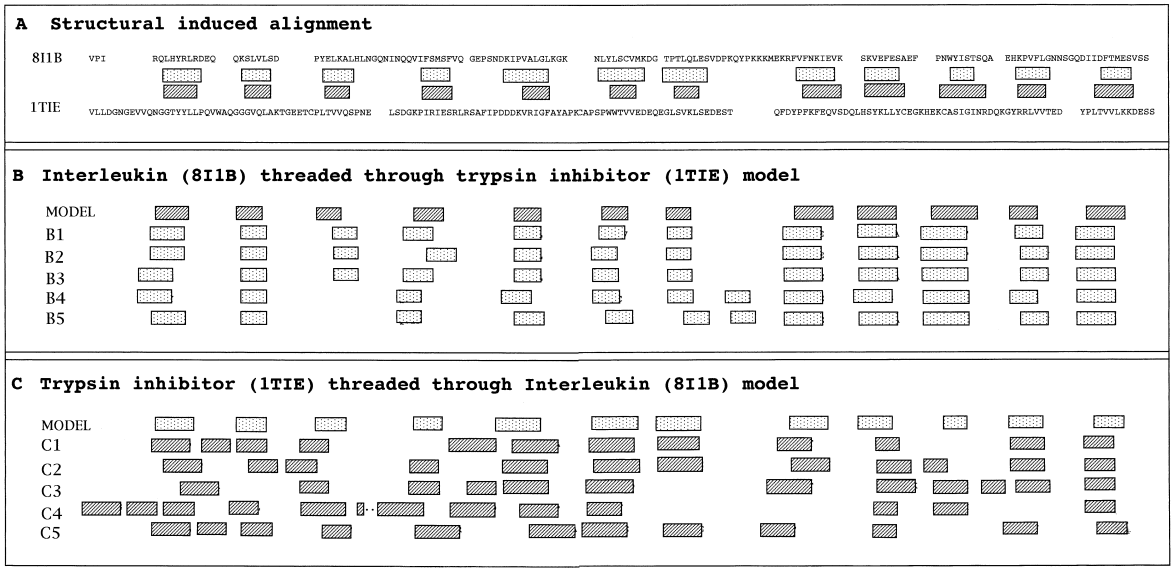

Figure 7. Threading structural analogs, 1tie vs. 8i1b. The sequences of 1TIE and 8I1B cross-threaded through each other's structural core models. (a) FSSP-defined structural alignment (Holm & Sander, 1994); (b) 8I1B sequence threaded through the 1TIE structural model; (c) 1TIE sequence threaded through the 8I1B structural model. Labeling is as for Figure 6, except that the reference alignment is taken from the FSSP structural database (Holm & Sander, 1994), light boxes indicate 8I1B, and dark boxes indicate 1TIE. The optimal threadings of each sequence through the other's structural model are given for all five score functions: (1) Bryant & Lawrence (1993); (2) Maiorov & Crippen (1992); (3) Miyazawa & Jernigan (1985); (4) Sippl (1990, 1993); (5) White *et al*. (1994).

most trials involve search spaces that are substantially smaller, hence more quickly searched. As Figure 3 shows, most results can be obtained for relatively little computational effort. All five score functions converged quickly on almost all trials that involved a small to medium-sized search space. An important implication is that structural models at the level of super-secondary structure or domain motifs should thread very quickly. This result should greatly encourage efforts to recognize protein structure building blocks and assemble them hierarchically.

The detailed discussion in Appendix II describes a technical point about the lower bound mechanism that is potentially important. It corresponds to the use of a singleton approximation to a pairwise effect (there, the use of J^* to approximate H^*). The singleton approximation is then relaxed to the full pairwise computation only at specific points where it actually matters. It is likely that this same principle could be used to accommodate triplet and other higher-order interaction terms, by successively relaxing the pairwise computation to a triplet computation, then triplet to quadruplet, and so forth, only at specific points where it actually matters. In this way, it may be possible to achieve arbitrarily higher-order interactions describing the fully specified environment of every amino acid residue.

Biological examples

It is clear from Figure 2 that nearly half of all core secondary structure segments are correctly aligned with their native subsequences. However, correctly predicting the alignment of an entire core from the optimal threading is still very difficult, and most optimal self-threadings contain several errors. Thus, the current score functions demonstrate the ability to ''predict'' alignment in only two senses: the native threading nearly always scores much better than the vast majority of all possible threadings; and the optimal threading often places residues at or near their proper analogous positions. Of course, the accuracy of the studied score functions as presented elsewhere may be different for many reasons, among which are: they may have been improved by their authors since the referenced publication; they may have used a non-global threading algorithm, or one that prohibits variable-length loops; they may have used information from loop or side-chain spatial coordinates; they may have been tested on a much smaller set of proteins; they may have used a different set of proteins to estimate score function parameters.

Consistent with current understanding of protein structure, local structural environments and hydrophobic effects dominate the optimal threadings. β-Strand displacements of one or two are common, while α-helix displacements of one, three or four are more common than two. Because β-strands have a periodicity of 2 while helices have a periodicity of 3.6, these errors are expected from any score function that reflects the hydrophobic compatibility of an amino acid residue with its environment. For buried (respectively surface) β-strands, threading errors of one (respectively two) have minimal hydrophobic implications. For surface helices, threading displacements of one, three and four usually preserve amphipathic registration. All five of the threading score functions investigated placed considerable weight on the hydrophobic compatibility of an amino acid residue with its environment. This does not necessarily imply an explicit encoding of hydrophobicity or exposure, however. Implicit pairwise encodings of buried or exposed include, for example, whether a particular type of amino acid residue prefers neighbors that are hydrophobic (buried) or hydrophilic (exposed); or whether it prefers to have many neighbors (buried) or few (exposed).

Active sites typically are composed of conserved functionally important residues in structural environments they might not otherwise occupy. Functional constraints are not modeled by structural environments, nor by score functions based on statistical occurrence frequencies. Thus, active-site residues are more likely to be mis-threaded than are functionally neutral residues. Also, general structural models for non-homologous proteins with no common function should not include function-related effects, while models for homologous extension modeling should. Finally, data used to estimate score function parameters probably should exclude amino acid residues taken from active sites or co-factor binding positions.

There appears to be no obvious biological reason for the asymmetry in threading performance depending on whether interleukin 1-β or trypsin inhibitor was used as the structural model. Both structures have a set of adjacent β-strands that are considerably longer than the rest, offset by four β-strand positions. Both models contain one element of one core segment that protrudes beyond the FSSP structural alignment (Holm & Sander, 1994); removing this element from the model shifts the threadings but does not improve the result. The example's performance is consistent with recent blind prediction tests (Moult *et al*., 1995; Shortle, 1995).

Our search method supports any loop score function that depends only on how the sequence is threaded through pairs of core segments. The threading of any two adjacent core segments fixes the subsequence in the intervening loop region, and so the loop score function may assign any arbitrary score to each different possibility. Such a sequence-specific score function for the loops may be sufficient; across distant homologs, loops are highly variable in composition, length, and spatial position. Consequently, the coordinated mutations required for specific pair interactions are less likely than in the conserved core. Some other approaches exhibit large pair interaction scores for the loop regions, but implicitly encoded hydrophobic effects rather than specific pairwise interactions may be the dominant contribution.

In contrast, methods that do use amino acid pair interaction terms to compute the loop score function require detailed information about loop placement in 3D space. This information is not generally available to any structural model that allows loop insertions or deletions of arbitrary length; our core models do allow the insertion or deletion of upto an entire folding domain (see Starzyk *et al*., 1987), a common occurrence in multi-domain proteins. Malorov & Crippen (1994) propose an elegant approach that could include the needed 3D information for pairwise terms, but there is currently no practical algorithm for their scheme. Using our current search method, loop super-secondary structures such as beta-hairpins or tight turns could be included as additional types of core segments, and so fully evaluated with a pairwise score function, but this would require identifying turns and loops that were structurally invariant.

Due in part to the limited size of the current database, some amino acid pairs are sparsely populated in particular structural environments. In such cases the threading score functions are sensitive to fluctuations in the observed number of occurrences. For example, methods that use the logarithms of probabilities or derived odds ratios are sensitive to small frequencies because the logarithm becomes numerically unstable as its argument approaches zero. Consequently, score functions may tend to ''memorize'' the particular proteins in the data set used to derive score function parameters, and cross-validation (jack-knife, hold-one-out, etc.) tests are critical (see the discussion by Ouzonis *et al*., 1993). In our experience, failure to cross-validate can double the estimated accuracy, and over-estimates of 10 to 50% are common.

Conclusions

We have discussed several technical problems: (1) the problem of multiple nearby regions of hydrophobic compatibility, leading to amphipathic alignment shifts (Figure 4); (2) the problem of cascaded registration shifts (Figure 5(a)) due to ''look-alike'' regions of secondary structure; (3) the problem of pairwise interactions propagating local errors, leading to non-local alignment errors (Figure 5(b)); (4) the problem of monomer *versus* multimer mismatch due to incompatible hydrophobic faces (Figure 6(a)); (5) the problem of functionally important, but structurally inconsistent, residues (Figure 6(b)); and (6) the problem of length mismatch between model core and modeled sequence (Figure 6(c)).

Additionally, we discussed several methodological problems that previously have led to overly optimistic estimates of predictive power: (1) failure to cross-validate the score function; (2) ''sequence memory'' in the structural model; and (3) ''gap length memory'' in the sequence-structure alignment. These problems arise whenever the original training proteins are encoded in the score function parameters, whenever the original side-chains are encoded in any structural environments derived using them, and whenever the original protein loop lengths are encoded in the search for the best threading. The problems may be reduced by withholding the protein being tested (and its sequence homologs) when generating score function parameters, by removing the original side-chains before computing structural environments, and by withholding the original loop lengths from the search procedure.

In order for the threading methodology to progress, several other problems must be solved. First, a proper statistic must be found to compare optimal threadings of the same sequence through models of very different complexity (Bryant & Altschul, 1995). Second, the current score functions must continue to evolve. For example, they must compensate for low numbers of amino acid pairs in many structural environments, and properly treat residues in ''special sites.'' Third, the structural models must generalize the X-ray determined structures: backbone coordinates should come from idealized geometric models, not native proteins, symmetry often allows a simple description of alternate but related models, β-barrels may have superimposable backbones but different N-terminal entries into the structure, and superimposable β-sheets may have different topological connectivity. For example, Altman & Gerstein (1994) describe probabilistic 3D core elements; Finkelstein & Reva (1991) describe variable-length core segments.

Without certainty as to the optimal threading, validation of new score functions or structural models will be inconclusive. It will be impossible to ascertain whether inferior performance is due to deficiencies in a newly proposed score function, or whether the score function is in fact accurate and the poor performance is due to approximation errors introduced by the search method. For example, Jones *et al*. (1992) report cases in which their search algorithm misaligned the proteins because it failed to find a reasonable optimum. The converse applies also; approximate search methods that use the native threading to initialize the search necessarily bias the search to favor the native threading, and consequently may yield a performance estimate that is unduly optimistic. Our search algorithm, and its ability to identify the optimal threading, provides the tool needed to validate modeling and score function improvements as they are proposed.

Although here we discarded cases for which the sequence is too short to completely occupy the structural model, there is great practical interest in threading a short sequence fragment through a structural model to find the portion of the model preferred by the fragment. This might arise, for example, in attempts to identify the structure corresponding to cDNA fragments. Our search method could accomplish this. The fragment could be embedded in a long string of null characters, and

the score function modified to assign zero to all nulls; threading the modified string through the model would optimally align the fragment. Of course, problems of estimating the contributions due to missing interactions and hydrophobic effects would arise.

Alignment constraints from functionally conserved residues, residue properties and primary sequence patterns may be incorporated directly into our branch-and-bound search algorithm. Combined structural and functional pattern recognition is likely to prove more powerful than either alone.

We have described the first practical method of finding the global optimum threading when both variable-length gaps and pair score functions are included. The method achieved speeds that were 10^{25} times faster than the best previously published speed for an exact solution, while using the same score function (compare Bryant & Lawrence (1993) with line 58 of Table 1). Consequently, a great many more protein sequences and models now can be threaded quickly and exactly.

The program and core library described here are available electronically from: MIT Technology Licensing Office, 28 Carleton Street, E32-300, Cambridge, MA 02142 USA (send email to Heather Mapstone: mapstone@mit.edu). An email server is available by sending the word ''help'' as the subject to needle-request@darwin.bu.edu. A WWW home page is available at http://bmerc-www.bu.edu/needle/ or at http://www.bmerc.bu.edu/needle/.

Methods

This section describes our branch-and-bound search algorithm, which finds the mathematically exact global optimum gapped sequence-structure alignment. It begins with preliminary definitions and the general form of the score function. Then it shows how the branch-and-bound search algorithm works: it repeatedly subdivides the search space into smaller subsets, and chooses the most promising subset to split next. The Appendices provide mathematical details, and also efficient methods for computing search space size, uniform random sampling, segment placement probabilities, mean, standard deviation and the partition function.

Definitions and preliminaries

The notation used here was formalized by White *et al*. (1994), Lathrop (1994) and Stultz *et al*. (1995). A protein sequence, **a**, consists of n amino acid residues, a_i. A core structural model, C, consists of m core segments, C_i, each of length c_i. The jth position in C_i is the core element $C_{i,j}$. It is created by erasing one residue identity (side-chain) from the modeled structure, and will be occupied by a single amino acid residue from the threaded sequence. Core segments are connected by a set of loop regions, λ, with loop λ_i connecting C_i to C_{i+1}, N-terminal leader λ_0 preceding C_1, and C-terminal trailer λ_m following C_m. Knowing the endpoints of λ_i is equivalent to knowing the threadings of C_i and C_{i+1}. The maximum (respectively minimum) length of loop λ_i is $l_i^{\max}$ (respectively $l_i^{\min}$). Unless stated otherwise, $l_i^{\max} = +\infty$ and $l_i^{\min}$ = the minimum geometric spanning loop length. Other values can reflect knowledge of additional constraints. For example, loops assigned length zero or 1 by the crystallographer usually reflect constrained ''kinks'' in the secondary structure that should be retained ($l_i^{\max} = l_i^{\min} = l_i^{\text{native}}$) or restricted ($l_i^{\max} = 1$ and $l_i^{\min} = 0$). As another example, Bryant & Lawrence (1993) set $l_i^{\max}$ and $l_i^{\min}$ based on the maximum and minimum loop lengths in an aligned homologous family.

An interaction graph (also called interaction matrix, adjacency graph, neighbor matrix, contact map, etc.) describes core element positions that are ''neighbors.'' Positions are defined to be neighbors if they interact in the assumed score function. The interaction graph consists of a set ν of vertices and a set ϵ of edges. Graph vertices $v \in \nu$ correspond one-to-one with core elements $C_{i,j}$ (the vertices are simply a second indexing scheme). Each graph edge $\{u, v\} \in \epsilon$ indicates that vertices u and v are neighbors in the folded structure. For example, an interaction graph or matrix summed across core segments is shown in Figure 5(b). Each vertex and edge is assigned an environment by an environmental state function s. Each vertex v is assigned an environment $s(v)$ that may describe exposure to solvent, secondary structure, etc. Each edge $\{u, v\}$ is assigned an environment $s(\{u, v\})$, which may additionally describe the distance between vertices u and v, etc. Being unaligned regions, loops do not participate directly in the interaction graph of pairwise relations. Rather, a loop score function separately scores each loop region and every possible spanning subsequence.

Any specific threading, $\mathbf{t}^a$, associates one sequence index from **a** with the first core element of each core segment, as shown in Figure 1(C). Thus, the threading may be represented compactly by a vector of length m as $\mathbf{t}^a = (t_1^a, t_2^a, \ldots, t_m^a)$. The vector space axes correspond to core segments, and the coordinates t_i^a specify their sequence indices. Due to the spacing constraints, $1 + \Sigma_{j<i}(c_j + l_j^{\min}) \leqslant t_i^a \leqslant n + 1 - \Sigma_{j \geqslant i}(c_j + l_j^{\min})$. Currently we maintain core segment topological order. This implies ordering constraints, $t_i^a + c_i + l_i^{\min} \leqslant t_{i+1}^a \leqslant t_i^a + c_i + l_i^{\max}$.

It results in simpler notation if the absolute sequence coordinates $\mathbf{t}^a$ are converted to relative coordinates $\mathbf{t}$ defined by $t_i = t_i^a - \Sigma_{j<i}(c_j + l_j^{\min})$. Let $\tilde{n} = n + 1 - \Sigma_i(c_i + l_i^{\min})$ and $\tilde{l}_i = l_i^{\max} - l_i^{\min}$. The quantity $\tilde{n}$ is the number of distinct placements in the sequence that are accessible to any given core segment, hence the ''effective length'' of the sequence relative to the model, and $\tilde{l}_i$ is the ith loop length variability. Then for all segments C_i, $t_i = 1$ corresponds to the lowest legal value of t_i^a and $t_i = \tilde{n}$ to the highest, the spacing constraints simplify to $1 \leqslant t_i \leqslant \tilde{n}$, and the ordering constraints simplify to $t_i \leqslant t_{i+1} \leqslant t_i + \tilde{l}_i$. Below, the absence of the superscript a will indicate relative coordinates.

The general score function

Given a specific protein sequence and a structural model of m core segments, the fully general form of the score function is:

$$f(\mathbf{t}) = \sum_i g_1(i, t_i) + \sum_i \sum_{j>i} g_2(i, j, t_i, t_j) + \cdots + \sum_i \sum_{j>i} \cdots \sum_{l>k} g_m(i, j \ldots k, l, t_i, t_j \ldots t_k, t_l) \quad (1)$$

where $i, j \ldots k, l$ index core segments and $t_i, t_j \ldots t_k, t_l$ give their relative positions in the sequence. The final sum is repeated over m indices representing all m core

segments, and reflects amino acid interactions among all m core segments simultaneously.

In practice there are insufficient data to specify all the parameters such a function would imply. Approaches differ in where they terminate the expansion. Profile-based methods employ only the g_1 term. Dynamic programming methods that do not allow pairwise amino acid interactions employ the g_1 term plus an affine gap penalty g_2 term of the form $g_2(i, i+1, t_i, t_{i+1}) = a + b|l_i - l_i^{\text{native}}|$. Methods that permit pairwise interactions, as here, employ a full g_2 term. The triplet interactions used by Godzik *et al*. (1992) require a g_3 term. No current threading proposal has yet suggested a score function requiring g_4 or higher terms, though they would arise in detailed treatment of steric packing among multiple core segments or linked constraint equations on structural environments. Grossman *et al*. (1995) and Bagley *et al*. (1995) describe full models of local structural neighborhood (g_4 or higher), but have not yet applied them to threading.

We implement the general pairwise score function

Here, the score of a candidate threading is defined to be a function only of the sum of (1) terms in g_1 that depend only on the threading of a single core segment, plus (2) terms in g_2 that depend only on the threading of a pair of core segments:

$$f(\mathbf{t}) = \sum_i g_1(i, t_i) + \sum_i \sum_{j>i} g_2(i, j, t_i, t_j) \qquad (2)$$

Our search method provides a direct, mathematically exact implementation for any score function that can be expressed in this form.

In principle, every core element and every possible pair of elements could be assigned a unique structural environment encoding a different score table, and each loop region could assign a different score to every possible spanning subsequence. Consequently, equation (2) is fully general for pair interactions. In most threading schemes, the score of a candidate threading is built up from the scores of the vertices and edges in the interaction graph, and the sequence composition of the loop regions. Appendix I shows how g_1 and g_2 might be derived for a typical score function. Score functions that depend on separation in the sequence, proximity to the N or C-terminal end of the sequence, or specialized identities of particular segments (e.g. including a regular expression pattern match based on known enzymatic function) are accommodated easily because the segment numbers (i, j) and segment indices (t_i, t_j) appear explicitly in the g_1 and g_2 argument lists. Other score components may be included provided they depend only on singleton or pairwise core segment threadings as shown.

The functions g_1 and g_2 are the essential point of contact between the search algorithm and any particular choice of scoring function, neighbor relationships, or structural environments. The search algorithm is driven only by g_1 and g_2, regardless of how the score function assigns values to them. For example, the segment placement constraint of Figure 5(a2), and the histidine constraint of Figure 6(b2), both were implemented very simply by setting $g_1(i, t_i) = +\infty$ wherever $\langle i, t_i \rangle$ violated a constraint.

Branch-and-bound search

We use the well-known branch-and-bound search algorithm (Winston, 1993; Kumar, 1992) to efficiently search the space of legal threadings for the global optimum. The search process is illustrated by Figure 8 and the algorithm is given below. Branch-and-bound search has been applied to molecular conformations (Havel *et al*., 1983) and protein functional patterns (Lathrop *et al*., 1987).

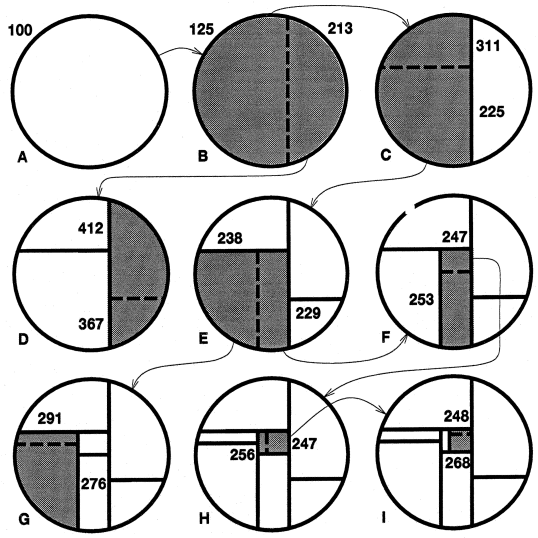

Figure 8. Splitting the search space. The first A through I steps in a hypothetical search. Step A shows the entire search space schematically as a circle. Steps B through I show successive hypothetical splits dividing the search space into finer subsets. Subsets resulting from each split are associated with a hypothetical lower bound (always the closest number to an active subset). At each step the subset having the current lowest lower bound is selected (the next selected subset is shown shaded) and split (the next split is shown as a broken line) into three subsets (the two regions on either side, and the line itself). The tail of each arrow is on an active subset, and the head points to the step where it is split. Although we cannot graphically show subsets corresponding to the line, they represent the mechanism by which dimensionality is reduced. At all times, the union of the subsets covers the entire search space. Some subsets (e.g. perhaps the upper left corner in step C) may have very high lower bounds and consequently may not be selected again until after the global optimum has been found; such subsets are implicitly pruned. The sorted list of lower bounds pending at each step, with new additions underlined: (a) 100; (b) 125, 213; (c) 213, 225, 311; (d) 225, 311, 367, 412; (e) 229, 238, 311, 367, 412. (f) 238, 247, 253, 311, 367, 412; (g) 247, 253, 276, 291, 311, 367, 412; (h) 247, 253, 256, 276, 291, 311, 367, 412; (i) 248, 253, 256, 268, 276, 291, 311, 367, 412.

In order to succeed, our branch-and-bound search requires the ability to: (1) represent the entire search space as a set of possibilities; (2) split any set into subsets; and (3) compute a lower bound on the best score achievable within any subset. Any correct implementation of these three requirements would result in a correct search, but search speed may vary dramatically. The keys to an efficient search are a powerful lower bound and good branch-points when splitting sets.

The search begins with a single set containing all legal threadings. At each step, the algorithm chooses the set with the currently lowest lower bound and splits it into several subsets. The entire search space always is

represented explicitly as the union of the sets created so far. After some finite number of steps, the chosen set will contain only one threading. Its score equals its lower bound. Every other set had an equal or greater lower bound, and so every other threading must have an equal or greater score. Consequently, this is the desired global optimum threading.

A set is pruned whenever its lower bound is above the global minimum score, because the global optimum threading will be discovered before that set is ever considered again. The global minimum score is unknown until the search terminates, and so pruning is implicit. If the search space may be pruned rapidly, then the search may be relatively short. Cases of multiple threadings with the same optimal score occur very rarely, and are detected automatically by continuing the search.

Algorithm

Initialization. (1) Compute a lower bound for the set of all threadings. (2) Initialize a sorted list to contain one entry, the set of all threadings with its lower bound.

Iteration. (1) Remove from the list the set having the lowest lower bound. (2) If the set contains only one threading, stop and announce success. This is a global optimum threading. The procedure later may be continued from this point to enumerate successive candidate threadings in score order. (3) Otherwise, split the set into smaller subsets. (4) Compute a lower bound for each new subset. (5) Merge the new subsets into the list, sorted by lower bound. The sorted list is implemented as a priority queue, or heap (Aho *et al*., 1982), for rapid access to the currently lowest lower bound.

Sets of threadings

A set $\mathcal{T}$ of threadings may be specified as $\mathcal{T} = \{\mathbf{t} | b_i \leqslant t_i \leqslant d_i\}$. This is indicated schematically by the dark arrows in Figure 1(D). The integers b_i and d_i define an interval, $[b_i, d_i]$, made up of the allowed sequence coordinates for core segment C_i. These m intervals may be represented compactly by two m-length vectors, **b** and **d** (mnemonic for ''Begin'' and ''enD''). This allows us to represent all sets $\mathcal{T}[\mathbf{b}, \mathbf{d}]$ that have the particularly simple form of an m-dimensional axis-parallel hyper-rectangle whose two opposite corners are the vectors **b** and **d**. A list of several hyper-rectangles corresponds to the union of the sets they represent.

Each hyper-rectangle also contains a large number of illegal threadings that violate spacing or ordering constraints. Illegal threadings are ignored, always. By convention, if $\mathbf{t}^{\text{ill}}$ is an illegal threading then $f(\mathbf{t}^{\text{ill}}) = +\infty$. Whenever we speak of a set of threadings we mean only the legal ones. Whenever search space sizes are computed, only legal threadings are counted.

The set of all legal threadings is represented by the hyper-rectangle:

$$\mathcal{T}[\mathbf{1}, \tilde{\mathbf{n}}] = \{\mathbf{t} | 1 \leqslant t_i \leqslant \tilde{n}\} \tag{3}$$

This includes all threadings that satisfy the spacing and ordering constraints described above.

The ability to represent and manipulate the search space directly allows for controlling the search. If a particular list of hyper-rectangles is used to initialize the search (Algorithm, above), the subsequent search will examine only the corresponding threadings. For example, the placement constraint on segment 1 in Figure 5(a2) could have been implemented simply by setting $b_1 = d_1 = t_1^{\text{native}}$ during search initialization. The histidine constraint of Figure 6(b2) corresponds to initialization using a list of hyper-rectangles that occupy E7 and F8 by histidine in all possible ways.

Lower bound on scores in threading sets

The branch-and-bound search we employ exploits a lower bound on the score $f(\mathbf{t})$ attainable by any threading **t** in the set $\mathcal{T}$. Any correct lower bound would result in correct search behavior, but the stronger the lower bound, the more rapidly the search prunes unwanted sets of threadings and converges to the optimum. Evaluation of the lower bound occurs in the inner loop of the search algorithm and consumes virtually all of its computation time. Consequently, it is crucial that it be computable efficiently.

For example, one lower bound that is easy to derive and easy to compute can be obtained from equation (2) by summing lower bounds on each term separately:

$$\begin{aligned}\min_{\mathbf{t}\in\mathcal{T}} f(\mathbf{t}) &= \min_{\mathbf{t}\in\mathcal{T}} \sum_i \left[g_1(i, t_i) + \sum_{j>i} g_2(i, j, t_i, t_j) \right] \\ &\geqslant \sum_i \left[\min_{b_i \leqslant x \leqslant d_i} g_1(i, x) + \sum_{j>i} \min_{\substack{b_i \leqslant y \leqslant d_i \\ b_j \leqslant z \leqslant d_j}} g_2(i, j, y, z) \right]\end{aligned} \tag{4}$$

The indicated min operations are computable efficiently using binary trees over sub-intervals of $g_1(i, x)$, and quad-trees or 2D-trees (Sedgewick, 1990) over sub-intervals of $g_2(i, j, y, z)$. This simple formula works well for small cases, and consequently would be useful for threading small super-secondary structure motifs or for testing a prototype branch-and-bound search implementation. It is insufficiently powerful to provide effective pruning in search spaces larger than about 10^9 or 10^{12}.

We have explored several alternative forms of the lower bound (Lathrop & Smith, 1994). Our current version, denoted $\text{lb}(\mathcal{T})$, is:

$$\begin{aligned}\min_{\mathbf{t}\in\mathcal{T}} f(\mathbf{t}) \;&\geqslant\; \text{lb}(\mathcal{T}) \\ &= \min_{\mathbf{t}\in\mathcal{T}} \sum_i \left[g_1(i, t_i) + g_2(i-1, i, t_{i-1}, t_i) \right. \\ &\qquad \left. + \min_{\substack{\mathbf{u}\in\mathcal{T} \\ l_j^{\max} = +\infty}} \sum_{|j-i|>1} \tfrac{1}{2} g_2(i, j, t_i, u_j) \right]\end{aligned} \tag{5}$$

The enclosing $\min_{\mathbf{t}\in\mathcal{T}}$ ensures that the lower bound will be instantiated on a specific legal threading $\mathbf{t}^{\text{lb}} \in \mathcal{T}$. This will be used in splitting $\mathcal{T}$, below. The equation further ensures that the singleton term, in $g_1(i, t_i)$, remains consistent both with the terms that reflect loop scores, in $g_2(i-1, i, t_{i-1}, t_i)$, and with the other (non-loop) pairwise terms, in $g_2(i, j, t_i, u_j)$. The inner $\min_{\mathbf{u}\in\mathcal{T}}$ allows a different vector **u** for each i, but requires **u** to be a legal threading. The assumption $l_j^{\max} = +\infty$ supports an efficient implementation. Equation (5) would be a tight lower bound (i.e. actually achieved in $\mathcal{T}$) if we further required that $\mathbf{u} = \mathbf{t}$; but then evaluating the bound would be equivalent to solving the search problem. It is easy to see that if $\mathcal{T}$ is a singleton set, $\{\mathbf{t}\}$, then $\text{lb}(\{\mathbf{t}\}) = f(\mathbf{t})$.

Appendix II provides mathematical formulae and an efficient implementation for the lower bound.

Splitting threading sets

The second key element of our branch-and-bound search is the ability to successively subdivide sets of threadings, as illustrated in Figure 1(D). A set is split by choosing a single core segment C_i and split-point t_i^{split}. The interval $[b_i, d_i]$ is divided into three sub-intervals: the points (1) less than the split-point, $[b_i, t_i^{\text{split}} - 1]$; (2) equal to the split-point, $[t_i^{\text{split}}, t_i^{\text{split}}]$; and (3) greater than the split-point, $[t_i^{\text{split}} + 1, d_i]$. This results in three mutually disjoint and exhaustive subsets of the original set. There are many possible ways to choose C_i and t_i^{split} (Lathrop & Smith, 1994). The choice affects search speed, but not so much as does the choice of lower bound.

Currently, we choose t_i^{split} based on $\mathbf{t}^{\text{lb}} \in \mathscr{T}$, the specific threading that instantiates the lower bound in equation (5), by choosing $t_i^{\text{split}} = t_i^{\text{lb}}$.

It is less obvious how to select the core segment C_i at which to split. One simple method, easy to implement and appropriate for threading small super-secondary structure motifs, is to split at the segment having the widest interval, i.e. at the i that maximizes the value of $d_i - b_i$. The method we currently use chooses the segment i that has the most negative expected score contribution if its interval were to be split at t_i^{lb}. Specifically, we split at $\langle i, t_i^{\text{lb}} \rangle$ where i yields the most negative value of the expression:

$$P_1(i, t_i^{\text{lb}}) \times \left[g_1(i, t_i^{\text{lb}}) - \mu_i + \sum_{j \neq i} (1 - \alpha(j)/2)(g_2(i, j, t_i^{\text{lb}}, t_j^{\text{lb}}) - \mu_{i,j}) \right] \quad (6)$$

Assuming a uniform probability distribution over all legal threadings, $P_1(i, t_i^{\text{lb}})$ is the probability that a randomly drawn threading will place C_i at t_i^{lb}; μ_i is the expected value of $g_1(i, *)$; $\mu_{i,j}$ is the expected value of $g_2(i, j, *, *)$; and $\alpha(i)$ indicates whether segment i is active (variable) or inactive (fixed) in the set:

$$\alpha(i) = \begin{cases} 1, & \text{if } b_i < d_i; \\ 0, & \text{if } b_i = d_i. \end{cases} \quad (7)$$

In equation (6), the factor $P_1(i, t_i^{\text{lb}})$ biases the choice to prefer combinations of i and t_i^{lb} that are *a priori* more likely. The terms $g_1(i, t_i^{\text{lb}}) - \mu_i$ and $g_2(i, j, t_i^{\text{lb}}, t_j^{\text{lb}}) - \mu_{i,j}$ bias the choice to prefer scores that are lower than expected. The factor $(1 - \alpha(j)/2)$ assigns the entire pairwise term $g_2(i, j, t_i^{\text{lb}}, t_j^{\text{lb}}) - \mu_{i,j}$ to core segment C_i if C_j is inactive, and shares it evenly between them if both are active.

Appendix III gives a number of formulae that are important for characterizing the threading search space and interpreting the raw score that the optimal alignment provides. Fast approximate formulae and implementations for μ_i, $\mu_{i,j}$, and $P_1(i, t_i)$ are described in Appendix III, as are methods for computing exact formulae, including search space size, segment placement probabilities, uniform random sampling, score distribution mean and standard deviation, and the partition function.

Acknowledgements

We thank Raman Nambudripad, Ljubomir Buturović, Chris Gaitatzes, Melissa Cline, Lisa Tucker-Kellogg, Loredana Lo Conte and Srikar Rao for their work on core modeling and score functions; Bob Rogers for assisting in the code implementation; Jim White, Ilya Muchnik, Gene Myers and Jim Knight for discussions of the mathematical formalism; Barbara Bryant, Tomás Lozano-Perez and Patrick Winston for discussions of computational protein folding; Janice Glasgow, David Haussler and Alan Lapedes for applications and extensions; Steve Bryant, Gordon Crippen, Chip Lawrence, Vladimir Maiorov and Manfred Sippl for discussions of their score functions; and R. Mark Adams, Jean-Michel Claverie, Larry Cosenza, Dror Rosenbach, Collin Stultz, Teresa Webster and the anonymous referees for comments that improved the paper. Special thanks to all crystallographers who deposited their coordinates in the international scientific databases. This paper describes research performed at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, in consortium with the BioMolecular Engineering Research Center of Boston University, sponsored by the National Science Foundation under grant DIR-9121548. Support for the Artificial Intelligence Laboratory's research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-91-J-4038. Support for the BioMolecular Engineering Research Center's research is provided in part by the National Institutes of Health under grant RR02275-05. We also thank the Aspen Center for Physics at which part of this research was performed.

References

Abagyan, R., Frishman, D. & Argos, P. (1994). Recognition of distantly related proteins through energy calculations. *Proteins: Struct. Func. Genet.* **19**, 132–140.

Aho, A. V., Hopcroft, J. E. & Ullman, J. D. (1982). *Data Structures and Algorithms*, Addison-Wesley, Reading, MA.

Altman, R. & Gerstein, M. (1994). Finding an average core structure: application to the globins. In *Proceedings of the 2nd International Conference on Intelligent Systems for Molecular Biology* (Altman, R., Brutlag, D., Karp, P., Lathrop, R. H. & Searls, D., eds), pp. 19–27, AAAI Press, Menlo Park, CA.

Bagley, S. C., Wei, L., Cheng, C. & Altman, R. B. (1995). Characterizing oriented protein structural sites using biochemical properties. In *Proceedings of the 3rd International Conference on Intelligent Systems for Molecular Biology* (Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. & Wodak, S., eds), pp. 12–20, AAAI Press, Menlo Park, CA.

Bashford, D., Chothia, C. & Lesk, A. M. (1987). Determinants of a protein fold: unique features of the globin amino acid sequences. *J. Mol. Biol.* **196**, 199–216.

Bauer, A. & Beyer, A. (1994). An improved pair potential to recognize native protein folds. *Proteins: Struct. Funct. Genet.* **18**, 254–261.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). The protein data bank: a computer-based archival file for macromolecular structures. *J. Mol. Biol.* **112**, 535–542.

Bowie, F. U., Lüthy, R. & Eisenberg, D. (1991). A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, **253**, 164–170.

Bowie, J. & Eisenberg, D. (1993). Inverted protein structure prediction. *Curr. Opin. Struct. Biol.* **3**, 437–444.

Bryant, S. H. & Altschul, S. F. (1995). Statistics of sequence-structure threading. *Curr. Opin. Struct. Biol.* **5**, 236–244.

Bryant, S. H. & Lawrence, C. E. (1993). An empirical energy function for threading protein sequence through the folding motif. *Proteins: Struct. Funct. Genet.* **16**, 92–112.

Chothia, C. (1992). One thousand families for the molecular biologist. *Nature*, **357**, 543–544.

Crippen, G. M. (1991). Prediction of protein folding from amino acid sequence over discrete conformational spaces. *Biochemistry*, **30**, 4232–4237.

Eisenberg, D. & McLachlan, A. D. (1986). Solvation energy in protein folding and binding. *Nature*, **319**, 199–203.

Eriksson, A. E., Cousens, L. S. & Matthews, B. W. (1993). Refinement of the structure of human basic fibroblast growth factor at 1.6 Å resolution and analysis of presumed heparin binding sites by selenate substitution. *Protein Sci.* **2**, 1274–1284.

Fetrow, J. S. & Bryant, S. H. (1993). New programs for protein tertiary structure prediction. *Bio/Technology*, **11**, 479–484.

Finkelstein, A. V. & Reva, B. (1991). A search for the most stable folds of protein chains. *Nature*, **351**, 497–499.

Finzel, B. C., Poulos, T. L. & Kraut, J. (1984). Crystal structure of yeast cytochrome c peroxidase refined at 1.7 Å resolution. *J. Biol. Chem.* **259**, 13027–13036.

Godzik, A., Kolinski, A. & Skolnick, J. (1992). Topology fingerprint approach to the inverse folding problem. *J. Mol. Biol.* **227**, 227–238.

Goldstein, R. A., Luthey-Schulten, Z. A. & Wolynes, P. G. (1992). Tertiary structure recognition using optimized Hamiltonians with local interactions. *Proc. Natl Acad. Sci. USA*, **89**, 9029–9033.

Greer, J. (1990). Comparative modeling methods: application to the family of the mammalian serine proteases. *Proteins: Struct. Func. Genet.* **7**, 317–333.

Grossman, T., Farber, R. & Lapedes, A. (1995). Neural net representations of empirical protein potentials. In *Proceedings of the 3rd International Conference on Intelligent Systems for Molecular Biology* (Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T. & Wodak, S., eds), pp. 154–161, AAAI Press, Menlo Park, CA.

Havel, T. F., Kuntz, I. D. & Crippen, G. M. (1983). The combinatorial distance geometry method for the calculation of molecular conformation. *J. Theor. Biol.* **104**, 359–381.

Hendlich, M., Lackner, P., Weitckus, S., Floeckner, H., Froschauer, R., Gottsbacher, K., Casari, G. & Sippl, M. J. (1990). Identification of native protein folds amongst a large number of incorrect models: the calculation of low energy conformations from potentials of mean force. *J. Mol. Biol.* **216**, 167–180.

Holm, L. & Sander, C. (1993). Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.* **233**, 123–138.

Holm, L. & Sander, C. (1994). The FSSP database of structurally aligned protein fold families. *Nucl. Acids Res.* **22**, 3600–3609.

Johnson, M. S., Overington, J. P. & Blundell, T. L. (1993). Alignment and searching for common protein folds using a data bank of structural templates. *J. Mol. Biol.* **231**, 735–752.

Jones, D. T. & Thornton, J. M. (1993). Protein fold recognition. *J. Computer-Aided Mol. Design*, **7**, 439–456.

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

Kabsch, W. & Sander, C. (1983). Dictionary of protein secondary structure. *Biopolymers*, **22**, 2577–2637.

Karlin, S., Zuker, M. & Brocchieri, L. (1994). Measuring residue associations in protein structures: possible implications for protein folding. *J. Mol. Biol.* **239**, 227–248.

Kavanaugh, J. S., Rogers, P. H. & Arnone, A. (1992). High-resolution X-ray study of deoxy recombinant human hemoglobins synthesized from beta-globins having mutated amino termini. *Biochemistry*, **31**, 8640–8647.

Kumar, V. (1992). Search, branch-and-bound. In *Encyclopedia of Artificial Intelligence* (Shapiro, S. C., ed.), vol. 2, pp. 1468–1472, John Wiley & Sons, New York.

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

Lathrop, R. H. & Smith, T. F. (1994). A branch and bound algorithm for optimal protein threading with pairwise (contact potential) interaction preferences. In *Proceedings of the 27th Hawaii International Conference on System Sciences* (Hunter, L. & Shriver, B., eds), pp. 365–374, IEEE Computer Society Press, Los Alamitos, CA.

Lathrop, R. H., Webster, T. A. & Smith, T. F. (1987). ARIADNE: pattern-directed inference and hierarchical abstraction in protein structure recognition. *Commun. ACM*, **30**, 909–921.

Lüthy, R., Bowie, J. U. & Eisenberg, D. (1992). Assessment of protein models with three-dimensional profiles. *Nature*, **356**, 83–85.

MacLachlan, R. A. (1992). Editor of *CMU Common Lisp User's Manual*. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. CMU Common Lisp source code and executables are freely available *via* anonymous FTP from lisp-rt1.slisp.cs.cmu.edu (128.2.217.9) and lisp-rt2.slisp.cs.cmu.edu (128.2.217.10).

Maiorov, V. N. & Crippen, G. M. (1992). Contact potential that recognizes the correct folding of globular proteins. *J. Mol. Biol.* **227**, 876–888.

Maiorov, V. N. & Crippen, G. M. (1994). Learning about protein folding *via* potential functions. *Proteins: Struct. Funct. Genet.* **20**, 167–173.

Matsuo, Y. & Nishikawa, K. (1994). Protein structural similarities predicted by a sequence-structure compatibility method. *Protein Sci.* **3**, 2055–2063.

Miyazawa, S. & Jernigan, R. L. (1985). Estimation of effective interresidue contact energies from protein crystal structures: quasi-chemical approximation. *Macromolecules*, **18**, 534–552.

Moult, J., Pedersen, J. T., Judson, R. & Fidelis, K. (1995). A large-scale experiment to assess protein structure prediction methods. *Proteins: Struct. Funct. Genet.* **23**, 2–4.

Novotný, J., Rashin, A. A. & Bruccoleri, R. E. (1988). Criteria that discriminate between native proteins and incorrectly folded models. *Proteins: Struct. Funct. Genet.* **4**, 19–30.

Onesti, S., Brick, P. & Blow, D. M. (1991). Crystal structure of a Kunitz-type trypsin inhibitor from *Erythrina caffra* seeds. *J. Mol. Biol.* **217**, 153–176.

Orengo, C. A. & Taylor, W. R. (1990). A rapid method of protein structure alignment. *J. Theor. Biol.* **147**, 517–551.

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

Ouzounis, C., Sander, C., Scharf, M. & Schneider, R. (1993). Prediction of protein structure by evaluation of sequence-structure fitness. *J. Mol. Biol.* **232**, 805–825.
Richardson, J. S. (1981). The anatomy and taxonomy of protein structures. *Advan. Protein Chem.* **34**, 167–339.
Rost, B. & Sander, C. (1994). Conservation and prediction of solvent accessibility in protein families. *Proteins: Struct. Funct. Genet.* **20**, 216–226.
Russell, R. B. & Barton, G. J. (1994). Structural features can be unconserved in proteins with similar folds. *J. Mol. Biol.* **244**, 332–350.
Sander, C. & Schneider, R. (1991). Database of homolog-derived protein structures and the structural meaning of sequence alignment. *Proteins: Struct. Funct. Genet.* **9**, 56–68.
Sankof, D. & Kruskal, J. B. (1983). Editors of *Time Warps, String Edits and Macromolecules*, Addison-Wesley, Reading, MA.
Sedgewick, R. (1990). *Algorithms in C*, Addison-Wesley, Reading, MA.
Shortle, D. (1995). Protein fold recognition. *Nature Struct. Biol.* **2**, 91–93.
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. (1993). Boltzmann's principle, knowledge-based mean fields and protein folding. *J. Computer-Aided Mol. Design*, **7**, 473–501.
Sippl, M. J. (1995). Knowledge-based potentials for proteins. *Curr. Opin. Struct. Biol.* **5**, 229–235.
Sippl, M. J. & Weitckus, S. (1992). Detection of native-like models for amino acid sequences of unknown three-dimensional structure in a data base of known protein conformations. *Proteins: Struct. Funct. Genet.* **13**, 258–271.
Starzyk, R., Webster, T. & Schimmel, P. (1987). Evidence for disposable sequences inserted into a nucleotide fold. *Science*, **237**, 1614–1618.
Steele, G. L. (1990). *Common Lisp: the Language*. Digital Press, Bedford, MA.
Stultz, C. M., Nambudripad, R., Lathrop, R. H. & White, J. V. (1995). Predicting protein structure with probabilistic models. In *Protein Folding and Stability* (Allewell, N. & Woodward, C., eds), JAI Press, Greenwich. In the press.
Taylor, W. R. & Orengo, C. A. (1989). Protein structure alignment. *J. Mol. Biol.* **208**, 1–22.
Treharne, A. C., Ohlendorf, D. H., Weber, P. C., Wendoloski, J. J. & Salemme, F. R. (1990). X-ray structural studies of the cytokine interleukin 1-beta. *Prog. Clin. Biol. Res.* **349**, 309–319.
Vainshtein, B. K., Harutyunyan, E. H., Kuranova, I. P., Borisov, V. V., Sosfenov, N. I., Pavlovsky, A. G., Grebenko, A. I. & Konareva, N. V. (1975). Structure of leghaemoglobin from lupin root nodules at 5 Angstroms resolution. *Nature*, **254**, 163–164.
Watson, H. C. (1969). The stereochemistry of the protein myoglobin. In *Progress in Stereochemistry* (Aylett, B. J. & Harris, M. M., eds), vol. 4, pp. 299–333, Butterworths, London.
White, J., Muchnik, I. & Smith, T. F. (1994). Modeling protein cores with Markov random fields. *Math. Biosci.* **124**, 149–179.
Wilmanns, M. & Eisenberg, D. (1993). Three-dimensional profiles from residue-pair preferences: identification of sequences with β/α-barrel fold. *Proc. Natl Acad. Sci. USA*, **90**, 1379–1383.
Winston, P. H. (1993). *Artificial Intelligence*, 3rd edit., pp. 82–90, Addison-Wesley, Reading, MA.
Wodak, S. J. & Rooman, M. J. (1993). Generating and testing protein folds. *Curr. Opin. Struct. Biol.* **3**, 247–259.

Appendix I: A Typical Pairwise Score Function

Here we give an example of one way that a score function might be constructed. Details will vary with the particular score function and environment definitions chosen. Notation is explained in the Methods section of the main text.

For any threading $\mathbf{t}$, let $f_v(v, \mathbf{t})$ be the score assigned to core element or vertex v, $f_e(\{u, v\}, \mathbf{t})$ the score assigned to interaction or edge $\{u, v\}$, and $f_l(\lambda_i, \mathbf{t})$ the score assigned to loop region λ_i. Then the total score of the threading is:

$$f(\mathbf{t}) = \sum_{v \in \nu} f_v(v, \mathbf{t}) + \sum_{\{u,v\} \in \epsilon} f_e(\{u, v\}, \mathbf{t}) + \sum_{\lambda_i \in \lambda} f_l(\lambda_i, \mathbf{t}) \quad (8)$$

We can rewrite this as a function of threadings of pairs of core segments as follows:

$$\begin{aligned} f(\mathbf{t}) &= \sum_i \sum_{v \in C_i} f_v(v, \mathbf{t}) + \sum_i f_l(\lambda_i, \mathbf{t}) \\ &\quad + \sum_i \sum_j \sum_{\substack{\{u,v\} \in \epsilon \\ u \in C_i \\ v \in C_j}} f_e(\{u, v\}, \mathbf{t}) \qquad (9) \\ &= \sum_i \left[\sum_{v \in C_i} f_v(v, \mathbf{t}) + \sum_{\substack{\{u,v\} \in \epsilon \\ u,v \in C_i}} f_e(\{u, v\}, \mathbf{t}) \right] \\ &\quad + f_l(\lambda_0, \mathbf{t}) + f_l(\lambda_m, \mathbf{t}) \\ &\quad + \sum_i \left[f_l(\lambda_i, \mathbf{t}; 0 < i < m) \right. \\ &\quad \left. + \sum_{j \neq i} \sum_{\substack{\{u,v\} \in \epsilon \\ u \in C_i \\ v \in C_j}} f_e(\{u, v\}, \mathbf{t}) \right] \qquad (10) \\ &= \sum_i g_1(i, t_i) + \sum_i \sum_{j > i} g_2(i, j, t_i, t_j) \qquad (11) \end{aligned}$$

The singleton terms, in g_1, include contributions from sources such as (in the order in equation (10)) individual core elements assigned to particular structural environments, pairwise interactions within a single core segment, and loop scores of the N- and C-terminal loop regions. The pairwise terms, in g_2, include contributions from sources such as (in the order in equation (10)) interior loop scores, and pairwise interactions between different core segments.

Avoiding most of the computation

Pre-computing g_1 and g_2 and storing them in arrays permits rapid evaluation of individual

threadings as in equation (11), compared with their time-consuming *ab initio* evaluation as implied by equation (8). Storing g_1 requires $\mathcal{O}(m)$ arrays of size $\tilde{n}$, and storing g_2 requires $\mathcal{O}(m(m-1)/2)$ arrays of size $\tilde{n}(\tilde{n}+1)/2$, though in practice less storage is required because some core segment pairs do not interact.

Appendix II: Lower Bound

Efficient calculation of a strong lower bound is the essence of our branch-and-bound algorithm. The first part of this section describes an efficient implementation strategy. The second part describes a practical caching scheme that avoids much of the computation.

Implementation and recursive formula

This section describes an implementation of the lower bound $\mathrm{lb}(\mathcal{T})$ on the possible scores achieved by threadings within a set $\mathcal{T}$. The notation is explained in the Methods section of the main text.

As in equation (7) in the main text, we say that a search space axis i (i.e. the placement of core segment C_i in the sequence) is active in $\mathcal{T}$ if $b_i < d_i$ (i.e. the placement of core segment C_i in the sequence may vary within $\mathcal{T}$), and inactive if $b_i = d_i$ (C_i is fixed in $\mathcal{T}$). Note that this does not refer to pairwise or singleton contributions; both active and inactive segments may have contributions from both pairwise and singleton sources.

We separate the lower bound $\mathrm{lb}(\mathcal{T})$ into an inactive part $q(\mathcal{T})$ and an active part $r(\mathcal{T})$. These satisfy $\mathrm{lb}(\mathcal{T}) = q(\mathcal{T}) + r(\mathcal{T})$. The inactive part $q(\mathcal{T})$ sums the contributions that can be determined by knowing the exact placement of the inactive axes. These are the singleton contributions from each inactive axis, plus the pairwise contributions from each pair of inactive axes. For each subset created during the search, $q(\mathcal{T})$ is stored with the m-vectors **b** and **d** and updated at each split. The active part $r(\mathcal{T})$ estimates a lower bound on the contribution from the active axes plus their interactions with the inactive axes. It is recomputed each time the lower bound computation is done.

We use $\alpha(i)$ to indicate whether axis i is active (equation (7)), and $\beta(i, j)$ to indicate whether either of axes i or j are active. Let:

$$\beta(i,j) = \begin{cases} 1, & \text{if either axis } i \text{ or } j \text{ is active} \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

β is related to α by:

$$\beta(i,j) = \alpha(i) + \alpha(j) - \alpha(i)\alpha(j) \quad (13)$$

$$= \alpha(i)(1 - \alpha(j)/2) + \alpha(j)(1 - \alpha(i)/2) \quad (14)$$

Then, define:

$$q(\mathcal{T}) = \sum_i (1 - \alpha(i)) \times \left[g_1(i, b_i) + \sum_{j>i} (1 - \beta(i,j)) g_2(i, j, b_i, b_j) \right] \quad (15)$$

$$r(\mathcal{T}) = \min_{\mathbf{t} \in \mathcal{T}} \sum_i \left[\alpha(i) g_1(i, t_i) + \beta(i-1, i) g_2(i-1, i, t_{i-1}, t_i) + \alpha(i) \min_{\substack{\mathbf{u} \in \mathcal{T} \\ l_j^{\max} = +\infty}} \sum_{|j-i|>1} (1 - \alpha(j)/2) g_2(i, j, t_i, u_j) \right] \quad (16)$$

Note that in the inner $\min_{\mathbf{u} \in \mathcal{T}}$, the ordering constraints imply that $j < i \Rightarrow u_j \leqslant t_i$ and $j > i \Rightarrow u_j \geqslant t_i$, as otherwise $g_2(i, j, t_i, u_j) = +\infty$. By convention, $g_2(j, i, t_j, t_i) = g_2(i, j, t_i, t_j)$.

Recall that $\mathrm{lb}(\mathcal{T}) = q(\mathcal{T}) + r(\mathcal{T})$. The terms in equations (15) and (16) have the same meanings as in equation (5). The inactive part $q(\mathcal{T})$ is easy to update after each split simply by accounting for newly inactive axes. The remainder of this section describes a recursive formulation of $r(\mathcal{T})$ that leads to an efficient implementation.

Define H as:

$$H(i, t_i) = \alpha(i)[g_1(i, t_i) + H^*(i, m, t_i, d_m)] + \min_{\substack{x \leqslant \min(d_{i-1}, t_i) \\ x \geqslant \max(b_{i-1}, t_i - \tilde{l}_{i-1})}} (H(i-1, x) + \beta(i-1, i) g_2(i-1, i, x, t_i)) \quad (17)$$

where $g_1(i, t_i)$ accounts for singleton terms, $g_2(i-1, i, x, t_i)$ forces pairwise terms containing loop scores to be consistent between $i-1$ and i, and $H^*(i, m, t_i, d_m)$ bounds the contribution from non-loop pairwise terms at $\langle i, t_i \rangle$. H^* is defined as:

$$H^*(i, k, t_i, x) = \begin{cases} \min([H^*(i, k-1, t_i, x) + (1 - \alpha(k)/2) g_2(i, k, t_i, x)], \\ \quad H^*(i, k, t_i, x-1)), & \text{if } k < i-1 \text{ or } k > i+1 \\ H^*(i, k-1, t_i, x), & \text{if } i-1 \leqslant k \leqslant i+1 \\ +\infty, & \text{if } x < b_k,\ x > d_k, \\ & \quad k < i \text{ and } x > t_i \\ & \quad \text{or } k > i \text{ and } x < t_i \\ 0, & \text{otherwise} \end{cases} \quad (18)$$

Equation (18) treats i and t_i as parameters, and uses the assumption that $l_j^{\max} = +\infty$. From equation (18) it follows that:

$$H^*(i, m, t_i, d_m) = \min_{\substack{\mathbf{u} \in \mathcal{T} \\ l_j^{\max} = +\infty}} \sum_{|j-i|>1} (1 - \alpha(j)/2) g_2(i, j, t_i, u_j) \tag{19}$$

and consequently:

$$r(\mathcal{T}) = \min_x H(m, x) \tag{20}$$

as desired.

One important aspect of this lower bound computation is that the lower bound actually is instantiated on a specific threading $\mathbf{t}^{\text{lb}}$ in the outermost $\min_{\mathbf{t} \in \mathcal{T}}$ of equation (16). By keeping track of the indices x at which the minimum was actually achieved in equation (17), it is possible to follow the backtrace from the x minimizing equation (20) in order to produce $\mathbf{t}^{\text{lb}}$. This plays an important role in choosing the next split point, and in avoiding computation.

A reasonably efficient implementation results from holding H and H^* in arrays and iteratively computing the array values using dynamic programming techniques. The formal computational complexity of the lower bound computation is $\mathcal{O}(m^2\tilde{n}^2)$, but this can be reduced as described next. An open problem is to devise a clever tree-structured embedding that avoids brute-force iteration, much as binary trees avoid brute-force iteration when finding the minimum value on an interval (Sedgewick, 1990). A second open problem is to strengthen the current lower bound. A third is to generalize it to higher-order core segment interactions.

Avoiding most of the computation

Most of the time is expended while computing H^* for use in computing H. However, most values of H are so bad that we actually do not need the strong bound given by H^*. In most cases, we can substitute:

$$J^*(i, t_i) = \sum_{|j-i|>1} (1 - \alpha(j)/2) \min_{1 \leqslant x \leqslant \tilde{n}} g_2(i, j, t_i, x) \tag{21}$$

The fact that $J^*(i, t_i) \leqslant H^*(i, m, t_i, d_m)$ guarantees that the result is a valid lower bound. Computing $J^*(i, t_i)$ is very fast because $\min_x g_2(i, j, t_i, x)$ can be precomputed and stored for each (i, j, t_i), and the computation then reduces to sums of a few array references.

In fact, it is sufficient if we ensure that $\mathbf{t}^{\text{lb}}$ and the value of its associated lower bound are computed using H^*; all other cases may use J^*. To do this, we record all indices $\langle i, t_i \rangle$ that have ever appeared in $\mathbf{t}^{\text{lb}}$ during any lower bound computation. Equation (17) is computed using H^* (equation (18)) for each such $\langle i, t_i \rangle$, and using J^* (equation (21)) otherwise. Specifically, let:

$$\gamma(i, t_i) = \begin{cases} 1, & \text{if } \langle i, t_i \rangle \text{ ever appeared in any } \mathbf{t}^{\text{lb}} \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

$$\begin{aligned} H^{\text{fast}}(i, t_i) = \alpha(i)[&g_1(i, t_i) + \gamma(i, t_i) H^*(i, m, t_i, d_m) \\ &+ (1 - \gamma(i, t_i)) J^*(i, t_i)] \\ &+ \min_{\substack{x \leqslant \min(d_{i-1}, t_i) \\ x \geqslant \max(b_{i-1}, t_i - l_{i-1})}} (H^{\text{fast}}(i-1, x) \\ &+ \beta(i-1, i) g_2(i-1, i, x, t_i)) \end{aligned} \tag{23}$$

Equation (23) is used in place of equation (17) in order to avoid most invocations of equation (18).

It remains to ensure that the current computation did not reach a new $\langle i, t_i \rangle$ appearing in the current $\mathbf{t}^{\text{lb}}$ for the first time, by checking $\gamma(i, t_i^{\text{lb}})$ for each $\langle i, t_i^{\text{lb}} \rangle$. If $\gamma(i, t_i^{\text{lb}}) = 0$ for any i, then that $\gamma(i, t_i^{\text{lb}})$ must be set to 1 and the lower bound computation repeated. In practice, only a few such $\langle i, t_i \rangle$ ever appear. Because most values of H are sufficiently bad, the difference between H^* and J^* does not matter in most cases. Cases where it does matter typically are identified early on, and subsequently very little repeat computation is done.

An efficient implementation might scale and round the input in order to use fast integer arithmetic; keep arrays as nested pointers in order to avoid multi-dimensional array references; lay out large arrays in small pieces in order to minimize disk paging; precompute or cache values where possible; and so on. A parallel MIMD implementation could distribute subsets among arbitrarily many processors. A parallel SIMD implementation could embed the array computations in a connected grid of processors.

Diagnostic invariants

Two useful invariants are (1) monotonically decreasing subsets have monotonically increasing lower bounds, and (2) for any $\mathbf{t}$, $\text{lb}(\{\mathbf{t}\})$ by equation (5), $f(\mathbf{t})$ by equation (8), and $f(\mathbf{t})$ by equation (11), all are equal. In the interest of correct computer code, an implementation should verify the first invariant whenever a subset is split, and the second whenever a global optimum threading is found.

Appendix III: Characterizing the Search Space

Formulae in this section are used when splitting sets of threadings, and also to obtain important statistical information.

Fast approximate formulae

For the heuristic choice of which core segment to split in equation (6) of the main text, speed is important and approximate formulae are accept-

able. Fast formulae result from the simplifying assumptions that the entire search space is included ($b_i = 1$ and $d_i = \tilde{n}$), and that loops can be arbitrarily long ($l_i^{\max} = +\infty$). Equations (24) through (26) are exact for this case, and are used to approximate all other cases in equation (6). Exact algorithms for all other cases are given below.

Under these simplifying assumptions the number of legal threadings, or search space size, S, is the factorial choose function:

$$S \approx \binom{\tilde{n} + m - 1}{m} \tag{24}$$

Simple formulae also hold for segment placement probabilities. Under a uniform probability distribution on threadings, let $P_1(i, t_i)$ be the probability that segment i occurs at index t_i in a randomly drawn threading, and let $P_2(i, j, t_i, t_j)$ be the probability that segment i occurs at index t_i and simultaneously segment j occurs at index t_j. Where $i < j$ and $t_i \leqslant t_j$:

$$P_1(i, t_i) \approx \binom{t_i + i - 2}{i - 1}\binom{\tilde{n} - t_i + m - i}{m - i} \Big/ \binom{\tilde{n} + m - 1}{m} \tag{25}$$

$$P_2(i, j, t_i, t_j) \approx \binom{t_i + i - 2}{i - 1}\binom{t_j - t_i + j - i - 1}{j - i - 1} \times \binom{\tilde{n} - t_j + m - j}{m - j} \Big/ \binom{\tilde{n} + m - 1}{m} \tag{26}$$

Each factor is always the binomial coefficient corresponding to 1 less than the number of available sequence indices plus the number of core segments to occupy them, choose the number of core segments to occupy them. The denominator is the approximate search space size from equation (24). Successive factors in the numerator correspond to the combinatorial number of arrangements between successive pairs of core segments fixed by the arguments. Similar formulae hold for $P_3(i, j, k, t_i, t_j, t_k)$, for $P_4(i, j, k, l, t_i, t_j, t_k, t_l)$, and so forth.

These relations permit us to estimate the expected singleton and pairwise score components attributable to each segment. The expected singleton contribution for segment i is μ_i, and the expected pairwise contribution for segments i and j is μ_{ij}. Where $i < j$ and $t_i \leqslant t_j$:

$$\mu_i = \sum_x P_1(i, x) g_1(i, x) \tag{27}$$

$$\mu_{ij} = \sum_x \sum_{y \geqslant x} P_2(i, j, x, y) g_2(i, j, x, y) \tag{28}$$

By convention, $P_2(j, i, t_j, t_i) = P_2(i, j, t_i, t_j)$ and $\mu_{ji} = \mu_{ij}$.

Avoiding most of the computation

In practice we compute the logarithm of equations (25) and (26), then exponentiate. When loading the system we precompute and store $\log n$ for $n < 1000$ and $\log\binom{n}{k}$ for $k < 50$ and $n < 1000$. Equations (25) and (26) then require only the sum of a few array references plus a transcendental function call. The approximations to $P_1(i, t_i)$, $P_2(i, j, t_i, t_j)$, μ_i and μ_{ij} all are constant for a given search space, and are precomputed and stored when each search is begun. The storage required is approximately the size of the g_1 and g_2 arrays. Consequently, equation (6) requires only sums and products of a few array references.

Exact search space size, probabilities and uniform sampling

In practice, external knowledge may constrain core segments to arbitrary intervals or specify maximum loop lengths. This section provides exact formulae for such cases.

Search space size

Let $\mathcal{T}[\mathbf{b}, \mathbf{d}] = \{\mathbf{t} | b_i \leqslant t_i \leqslant d_i\}$ be the set of threadings delimited by $\mathbf{b}$ and $\mathbf{d}$, let $S[\mathbf{b}, \mathbf{d}]$ be the number of legal threadings it contains, and let $B(i, x)$ be the number of legal threadings of segments i through m when segment i is placed at relative sequence index x or higher. B is given by the recursive formula:

$$B(i, x) = \begin{cases} d_m - x + 1, & \text{if } i = m \text{ and } b_m \leqslant x \leqslant d_m \\ B(i, b_i), & \text{if } 1 \leqslant i \leqslant m \text{ and } x < b_i \\ B(i, x + 1) + B(i + 1, x) - B(i + 1, x + l_i + 1), & \\ & \text{if } 1 \leqslant i < m \text{ and } b_i \leqslant x < d_i \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

The numbers involved in computing B become combinatorially large; arbitrary precision integer arithmetic is a language primitive in LISP, and usually available as a subroutine in other languages.

Consequently:

$$S[\mathbf{b}, \mathbf{d}] = B(1, b_1) \tag{30}$$

is exact for arbitrary $\mathbf{b}$, $\mathbf{d}$, $\mathbf{l}^{\min}$, and $\mathbf{l}^{\max}$. By applying equation (29) to $b_i = 1$ and $d_i = \tilde{n}$:

$$S = S[\mathbf{1}, \tilde{\mathbf{n}}] \tag{31}$$

gives the exact size of the entire legal search space. This is the exact formula corresponding to the approximate equation (24), and is used for all search space sizes reported in this work.

Exact segment placement probabilities

Exact formulae for segment placement probabilities are computable as the ratio of the search space

sizes corresponding to the constrained and the entire search spaces. The denominator in all cases is the entire search space size given by equation (31). The numerator corresponding to $P_1(i, t_i)$ arises from the set of threadings that fix C_i at t_i, denoted $\mathcal{T}\langle i, t_i \rangle$. Its search space size $S\langle i, t_i \rangle$ may be computed from equation (29) applied to $b_j = \{\text{if } j < i \text{ then 1 else } t_i\}$ and $d_j = \{\text{if } j \leqslant i \text{ then } t_i \text{ else } \tilde{n}\}$. Then:

$$P_1(i, t_i) = S\langle i, t_i \rangle / S[\mathbf{1}, \tilde{\mathbf{n}}] \tag{32}$$

is the exact formula corresponding to the approximate equation (25). Similar formulae hold for $P_2(i, j, t_i, t_j)$, $P_3(i, j, k, t_i, t_j, t_k)$, $P_4(i, j, k, l, t_i, t_j, t_k, t_l)$, and so forth.

Uniform random sampling

Equation (29) also allows us to randomly sample the threadings in any set $\mathcal{T}[\mathbf{b}, \mathbf{d}]$, assuming a uniform probability distribution (blind draw) on threadings. Let s be a random integer uniformly drawn between one and $S[\mathbf{b}, \mathbf{d}]$ inclusive; uniform random numbers are a language primitive in LISP, and usually available as a subroutine in other languages. Convert s to a unique threading as follows: for i from 1 to m do (1) find x such that $b_i \leqslant x \leqslant d_i$ and $B(i, x + 1) < s \leqslant B(i, x)$. (2) Set t_i to $b_i + x$. (3) Set s to $s - B(i, x + 1)$. It is necessary to compute $S[\mathbf{b}, \mathbf{d}]$ and B only once for each set $\mathcal{T}[\mathbf{b}, \mathbf{d}]$.

Exact analytic search space mean and standard deviation

Let $f(t) = \Sigma_i g_1(i, t_i) + \Sigma_i \Sigma_{i<j} g_2(i, j, t_i, t_j)$ be the threading score function chosen. Then the distribution mean μ is:

$$\mu = E(f(*)) = \sum_i E(g_1(i, *)) + \sum_i \sum_{i<j} E(g_2(i, j, *, *)) \tag{33}$$

where:

$$E(g_1(i, *)) = \sum_x P_1(i, x) g_1(i, x) \tag{34}$$

$$E(g_2(i, j, *, *)) = \sum_x \sum_y P_2(i, j, x, y) g_2(i, j, x, y) \tag{35}$$

The standard deviation σ is:

$$\sigma = E([\mu - f(*)]^2) = E(\mu^2 - 2\mu f(*) + f^2(*)) = E(f^2(*)) - \mu^2 \tag{36}$$

where:

$$E(f^2(*)) = E\left(\left[\sum_i g_1(i, *) + \sum_i \sum_{i<j} g_2(i, j, *, *)\right]^2\right) \tag{37}$$

$$= \sum_i E([g_1(i, *)]^2) + 2\sum_i \sum_{i<j} E(g_1(i, *) g_1(j, *)) + \sum_i \sum_{i<j} \sum_k E(g_1(k, *) g_2(i, j, *, *)) + \sum_i \sum_{i<j} E([g_2(i, j, *, *)]^2) + 2\sum_i \sum_{i<j} \sum_{j<k} \sum_{k<l} E(g_2(i, j, *, *)\, g_2(k, l, *, *)) \tag{38}$$

$$E([g_1(i, *)]^2) = \sum_x P_1(i, x) [g_1(i, x)]^2 \tag{39}$$

$$E(g_1(i, *) g_1(j, *)) = \sum_x \sum_y P_2(i, j, x, y) g_1(i, x) g_1(j, y) \tag{40}$$

$$E(g_1(k, *) g_2(i, j, *, *)) = \sum_x \sum_y \sum_z P_3(i, j, k, x, y, z) g_1(k, z) g_2(i, j, x, y) \tag{41}$$

$$E([g_2(i, j, *, *)]^2) = \sum_x \sum_y P_2(i, j, x, y) [g_2(i, j, x, y)]^2 \tag{42}$$

$$E(g_2(i, j, *, *) g_2(k, l, *, *)) = \sum_x \sum_y \sum_z \sum_z P_4(i, j, k, l, x, y, z, v) \times g_2(i, j, x, y) g_2(k, l, z, v) \tag{43}$$

The analytic formula for the mean has a computational complexity of $\mathcal{O}(m^2 \tilde{n}^2)$. The analytic formula for the standard deviation has a computational complexity of $\mathcal{O}(m^4 \tilde{n}^4)$.

Avoiding most of the computation

In practice, the fourth-power computational complexity of the analytic standard deviation formula is burdensome for most proteins. Consequently, we usually estimate the mean and standard deviation by sampling the search space. Drawing and scoring 10,000 uniformly distributed random threadings using the methods in Appendix I (Avoiding most of the computation) and Uniform random sampling (above) takes only a few seconds. This results in sufficiently accurate estimates for most ordinary purposes. In cases where an exact

value is important, the analytic formulae are available at additional computational cost.

The partition function

Suppose that the scores $f(\mathbf{t}_i)$ are interpretable as (or convertible into) a pseudo-energy or negative logarithm of a probability, as is true for many methods. Then the Boltzman probability of any given threading $\mathbf{t}$ is:

$$P(\mathbf{t}) = \exp(-f(\mathbf{t})/rT)/Z \tag{44}$$

where rT is an adjustable temperature (or scale) factor controlling the relative importance of low pseudo-energy (or low scoring) threadings, and:

$$Z = \sum_{\mathbf{u}} \exp(-f(\mathbf{u})/rT) \tag{45}$$

This expression gives an explicit probability for each threading, but is difficult to evaluate because the summation is over all possible threadings. Extreme values in the low-pseudo-energy tail make the largest contributions and so dominate the computation, but are not well modeled by a normal distribution.

The partition function may be estimated by enumerating the (non-normal) low-pseudo-energy tail of the distribution explicitly, and estimating the body of the distribution from the normal distribution. Suppose the s lowest-pseudo-energy scoring threadings are explicitly enumerated as $\mathbf{t}_1, \ldots, \mathbf{t}_s$, then:

$$Z \approx \exp(-f_0/rT)\left[\sum_{i=1}^{s} \exp((f_0 - f(\mathbf{t}_i))/rT) + S\int_{z_s}^{\infty} \exp((f_0 - (\mu + \sigma z))/rT)N(z)\,dz\right] \tag{46}$$

where f_0 is the global minimum score, z_s is the z-score of $\mathbf{t}_s$, $N(z)$ is the normal density, and S is the search space size from equation (31). If the estimated contribution from the body of the distribution is ignored, the error ΔZ in the estimate of Z is bounded by

$$\Delta Z = \Sigma_{i=s+1}^{S} \exp(-f(\mathbf{t}_i)/rT) \leqslant S \exp(-f(\mathbf{t}_s)/rT),$$

and so

$$f(\mathbf{t}_s) > rT(\log S - \log \delta) \text{ implies } \Delta Z < \delta.$$

Edited by F. Cohen

(Received 6 January 1995; accepted in revised form 20 October 1995)