RAPTOR: OPTIMAL PROTEIN THREADING BY LINEAR PROGRAMMING

JINBO XU, MING LI
Department of Computer Science, University of Waterloo,
Waterloo, Ont. N2L 3G1, Canada, {*j3xu,mli*}*@math.uwaterloo.ca*

DONGSUP KIM, YING XU
Protein Informatics Group, Life Science Division and Computer Sciences and Mathematics Division,
Oak Ridge National Lab, Oak Ridge, TN 37831, USA, {*afg,xyn*}*@ornl.gov*

This paper presents a novel linear programming approach to do protein 3-dimensional (3D) structure prediction via threading. Based on the contact map graph of the protein 3D structure template, the protein threading problem is formulated as a large scale integer programming (IP) problem. The IP formulation is then relaxed to a linear programming (LP) problem, and then solved by the canonical branch-and-bound method. The final solution is globally optimal with respect to energy functions. In particular, our energy function includes pairwise interaction preferences and allowing variable gaps which are two key factors in making the protein threading problem NP-hard. A surprising result is that, most of time, the relaxed linear programs generate integral solutions directly. Our algorithm has been implemented as a software package RAPTOR – RApid Protein Threading by Operation Research technique. Large scale benchmark test for fold recognition shows that RAPTOR significantly outperforms other programs at the fold similarity level. The CAFASP3 evaluation, a blind and public test by the protein structure prediction community, ranks RAPTOR as top 1, among individual prediction servers, in terms of the recognition capability and alignment accuracy for Fold Recognition (FR) family targets. RAPTOR also performs very well in recognizing the hard Homology Modeling (HM) targets. RAPTOR was implemented at the University of Waterloo and it can be accessed at http://www.cs.uwaterloo.ca/~j3xu/RAPTOR_form.htm.

1. Introduction

The Human Genome Project has led to the identification of over 30 thousand genes in the human genome, which might encode, by some estimation, about 100,000 proteins as a result of alternative splicing. To fully understand the biological functions and functional mechanisms of these proteins, the knowledge of their three-dimensional structures is essential. The ambitious *Structural Genomics Initiatives*, launched by NIH in 1999, intends to solve these protein structures within the next ten years, through the development and application of significantly improved experimental and computational technologies.

A protein structure is typically solved using *x-ray crystallography* or *nuclear magnetic resonance spectroscopy (NMR)*, which are costly and very time-consuming (ranging from months to years per structure) and is quite difficult for high-throughput production. The overall strategy of the NIH Structural Genomics Initiatives is to solve protein structures using experimental techniques like x-ray crystallography or NMR only for a small fraction of all proteins and to employ computational techniques to model the structures for the rest of the proteins. The basic premise used here is that though there could be millions of proteins in nature, the number of unique structural folds is most likely 2-3 (or even more) orders of magnitude smaller. Hence by strategically selecting the proteins with unique structural folds for experimental solutions, we can put the vast majority of other proteins "within the modeling distance" of these proteins. Model-based structure prediction techniques could play a significant role in helping to achieve the goal of the Structural Genomics Initiatives. *Protein threading* represents one of the most promising such techniques.

Protein structure prediction through threading approach mainly involves the following four steps:

- Construction of structure template database: Select protein structures from protein structure databases as structural templates. This generally involves selecting proteins, from databases like FSSP[1], SCOP[2], or CATH[3,4], removing proteins with high sequence similarities, in order to save computational time.
- Threading energy function: Develop a statistics-based energy function which can be used to assess the quality of a predicted structure.
- Threading alignment: Align a query sequence with each of the structure templates which optimizes a specified scoring (energy) function.
- Threading prediction: Select the threading alignment that is statistically most probable, as the threading prediction; and make a structure prediction by placing the backbone atoms of the query protein to their aligned backbone positions of the selected structural template.

Protein threading can be used for both structure prediction and protein fold recognition, i.e., detection of homologous proteins. Numerous computer algorithms have been proposed for protein structure prediction, based on the threading approach. Based on the energy function models and computational methods, they can be grouped into three classes:

(1) The energy function does not include the pairwise interaction preferences explicitly. For this kind of models, a simple dynamic programming algorithm can be employed to optimize the energy function by aligning the template sequence (profile or Hidden Markov Model) to the target sequence (profile). Currently most threading servers belong to this category, including FUGUE[5], 3DPSSM[6], and GenTHREADER[7]. These methods are usually fast and achieve reasonable accuracy for HM targets, but they are generally not adequate for FR targets.

(2) The energy function includes the pairwise interaction preferences. It has been proved that this problem is NP-hard when variable gaps and pairwise interactions are considered simultaneously[8]. Various types of approximation algorithms have been used to optimize the energy function. These methods include double dynamic programming[9], interaction-frozen approximation[10], and Monte Carlo sampling algorithm[11]. Unfortunately, T. Akutsu and S. Miyano have proved that this problem is MAX-SNP-hard[12], which implies that no polynomial time algorithm can achieve arbitrarily close approximation unless NP=P. We must point out that all NP-hardness proofs are based on the unrealistic assumptions that all residue-residue contacts should be considered. Therefore, these intractability proofs may have nothing to do with the complexity of real life protein threading

problem afterall, as partially demonstrated in this paper.

(3) The energy function includes the pairwise interaction preferences and an exact algorithm is designed to optimize the energy function. Xu *et al.* have proposed a divide-and-conquer method employed by PROSPECT-I[13]. PROSPECT-I runs fast on simple protein template (interaction) topology but could take a long time or run out of memory (in a 32 bit platform) for protein templates with dense residue-residue interactions and long target sequences. Lathrop and Smith have proposed a practical branch-and-bound algorithm for protein threading. But for some data set, their algorithms couldn't converge within 2 hours for about 10% threading pairs[?].

The main focus of this paper is to develop a globally optimal and practically efficient threading algorithm by formulating the protein threading problem as a large scale integer/linear programming problem. The formulation is based on the alignment model treating the pairwise interaction preferences strictly and allowing variable gaps. The integer program solver could automatically select the optimization pathway by exploiting the relationships of various kinds of scores involved in the energy function. It also allows us to utilize the existing powerful linear programming packages to rapidly reach the optimal alignment. To our knowledge, this is the first time that the integer/linear programming approach is applied to protein threading.

This paper is organized as follows. Section 2 presents some assumptions and the alignment model employed by our algorithm. Section 3 gives three different integer program formulations of the protein threading problem and briefly analyzes their strengths and relationship. Section 4 presents some implementation details of RAPTOR. These include scoring systems, weight training, and Support Vector Machine fold recognition approach. Section 5 compares RAPTOR's performance with some popular servers using several standard large scale experimental benchmark data and the recently released CAFASP3 evaluation. In Section 5, we also present several predicted structures generated by RAPTOR. Section 6 analyzes the computational efficiency issues such as CPU time and memory consumption. Finally Section 7 discusses future extensions to RAPTOR.

2. Alignment Model

We represent the amino acid sequence, of length m, of a protein template by $t_1 t_2 \dots t_m$ and a query sequence, of length n, by $s_1 s_2 \dots s_n$.

Definition 1. An alignment between the template and the sequence is a set of pairs $(\hat{t_1}, \hat{s_1}), (\hat{t_2}, \hat{s_2}), \dots, (\hat{t_L}, \hat{s_L})$, where $L \leq m + n$, $\hat{t_1}\hat{t_2} \dots \hat{t_L}$ is an expansion of $t_1 t_2 \dots t_m$ by inserting some gaps and $\hat{s_1}\hat{s_2} \dots \hat{s_L}$ is an expansion of $s_1 s_2 \dots s_n$ by inserting some gaps and for any pair $(\hat{t_i}, \hat{s_i})$, at most one of $\hat{t_i}$ and $\hat{s_i}$ is a gap, for $1 \leq i \leq L$.

Definition is a very general definition of alignment, which could lead to a huge search space of feasible alignments. Some biological observations could be employed to give a more specific definition. In formulating the protein threading problem, we follow a few basic assumptions widely adopted by the protein threading community [13]. We assume that:

(1) Each template sequence is parsed as a linear series of cores with the connecting

loops between the adjacent cores. Each core is a conserved segment of an α-helix or β-sheet secondary structure. Although the secondary structure is often conserved, insertion or deletion may occur at the two ends of a secondary structure. So we only keep the most conserved part. Let $c_i = core(head_i, tail_i)$ denote all cores of one template, where $i = 1, 2, \dots, M$, M is the number of the cores, and $1 \leq head_1 \leq tail_1 < head_2 \leq tail_2 < \dots < head_M \leq tail_M \leq m$. The region between $tail_i$ and $head_{i+1}$ is a loop, for each i. The length of c_i is $len_i = tail_i - head_i + 1$. Let loc_i denote the sum of the length of all cores before c_i, i.e., $loc_i = \sum_{j=1}^{i-1} len_j$.

(2) When aligning a query protein sequence to a template, alignment gaps are confined to loops, that is, the regions between cores or the two ends of the template. The biological justification is that cores are conserved so that the chance of insertions or deletions within them is very little.

(3) We consider only interactions between residues in the cores. It is generally believed that interactions involving loop residues can be ignored as their contribution to fold recognition is relatively insignificant. We say that an interaction exists between two residues if the spatial distance between their C_β atoms is within $7A$ and they are at least 4 positions apart in the template sequence. We say that an interaction exists between two cores if there exists at least one residue-residue interaction between the two cores.

Our threading energy function consists of environment fitness score E_s, mutation score E_m, secondary structure compatibility score E_{ss}, gap penalty E_g and pairwise interaction score E_p. The overall energy function E has the following form:

$$E = W_m E_m + W_s E_s + W_p E_p + W_g E_g + W_{ss} E_{ss},$$

where $W_m, W_s, W_p, W_g, W_{ss}$ are weight factors to be determined by training.

Global alignment and global-local alignment methods are employed to align one template to one sequence. For the detailed description, we refer the reader to the paper by Fischer *et al*[14]. In the case that the template size is smaller than the sequence size, the whole template structure is aligned to the sequence. Head gap penalty and tail gap penalty are harnessed. In the case that the template size is larger than the query sequence size, it is possible that some cores at the two ends of the template cannot be aligned to the sequence. But we can always extend the sequence by adding some "artificial" amino acids to the two ends of the sequence to make all cores align to the (extended) sequence. All scores involving those extended positions are set to be 0.

3. Formulation

Definition 2. We use an undirected graph $CMG = (\mathcal{V}, \mathcal{E})$ to denote the (simplified) contact map graph of a protein template structure. Here, $\mathcal{V} = \{c_1, c_2, ..., c_M\}$ where c_i represents i^{th} core in the protein template, and $\mathcal{E} = \{(c_i, c_j) | \text{there is an interaction between } c_i \text{ and } c_j, \text{ or } |i - j| = 1\}$.

See Figure 1 for an example of contact map graph. For simplicity, when we say core c_i is aligned to position s_j, we always mean that core $c_i = (head_i, tail_i)$ is aligned to segment (s_j,s_{j+len_i-1}).

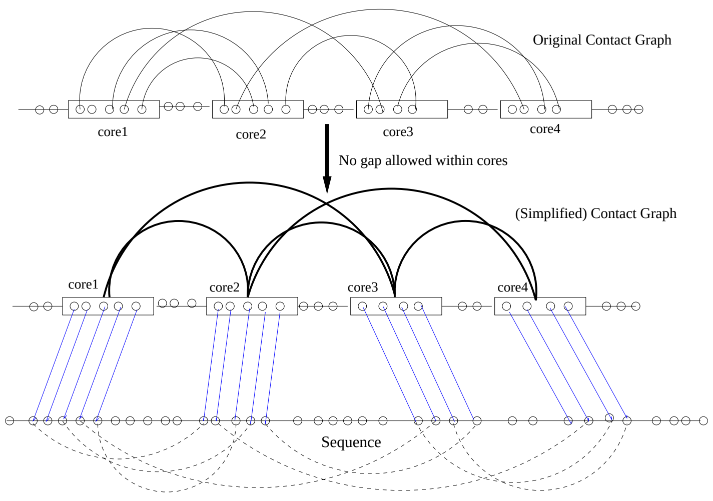

Fig. 1. Template contact graph and an example of alignment between a template and a sequence. A small circle represents a residue. A solid arc in the original contact graph represents an interaction between the two connected residues. A dashed arc means that if two query sequence residues are aligned to two interacting template residues, then the interaction score of these two query sequence residues must be counted in the energy function. The interaction score between two segments of the query sequence is the sum of the interaction scores of two query sequence residues which are aligned by two interacted template residues.

Definition 3. Let B denote the alignment bipartite graph of one threading pair. Each core of the template corresponds to one vertex in B, labeled as $c_i, i = 1, 2, \ldots, M$, each residue in the query sequence corresponds to one vertex in B, labeled as s_j, $j = 1, 2, \ldots, n$. The edges of B consist of all valid alignments between each core and each sequence position. The edges of B are also called the alignment edges.

Intuitively, if the alignment edges do not cross, we say they are not in conflict. Formally, this fact is stated in the next definition.

Definition 4. For any two different edges $e_1 = (c_{i_1}, s_{j_1})$ and $e_2 = (c_{i_2}, s_{j_2})$ in an alignment bipartite graph B, if $(loc_{i_1} - loc_{i_2}) \times (s_{j_1} + loc_{i_2} - loc_{i_1} - s_{j_2}) \leq 0$, then we say e_1 and e_2 are in *conflict*.

The following three lemmas give some properties of *conflict* and *non-conflict*. They are useful in understanding the correctness of our LP/IP formulation. Lemma describes the transitivity of *conflict* while Lemma says that *non-conflict* is also transitive.

Lemma 1. *For any three different edges $e_r = (c_{i_r}, s_{j_r})$, $r = 1, 2, 3$ and $loc_{i_1} < loc_{i_2} < loc_{i_3}$, if e_1 conflicts with e_2 and e_2 conflicts with e_3, then e_1 conflicts with e_3.*

Proof. For simplicity of notation, we will use l_i for loc_i.

$$\begin{aligned} & (l_{i_1} - l_{i_3})(s_{j_1} + l_{i_3} - l_{i_1} - s_{j_3}) \\ = & (l_{i_1} - l_{i_2} + l_{i_2} - l_{i_3})(s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2} + s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3}) \end{aligned}$$

$$\begin{aligned}
&= (l_{i_1} - l_{i_2})(s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2}) + (l_{i_2} - l_{i_3})(s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3}) + \\
&\quad (l_{i_1} - l_{i_2})(s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3}) + (l_{i_2} - l_{i_3})(s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2}) \\
&\le 0 + 0 + (l_{i_1} - l_{i_2})(s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3}) + (l_{i_2} - l_{i_3})(s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2}) \\
&= (l_{i_1} - l_{i_2})(l_{i_2} - l_{i_3})((s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3})/(l_{i_2} - l_{i_3}) \\
&\quad + (s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2})/(l_{i_1} - l_{i_2})) \\
&\le 0 \qquad \square
\end{aligned}$$

Lemma 2. *For any three different edges $e_r = (c_{i_r}, s_{j_r})$, $r = 1, 2, 3$ and $loc_{i_1} < loc_{i_2} < loc_{i_3}$, if e_1 does not conflict with e_2 and e_2 does not conflict with e_3, then e_1 does not conflict with e_3.*

Proof. As before, we will use l_i for loc_i.

$$\begin{aligned}
&\quad (l_{i_1} - l_{i_3})(s_{j_1} + l_{i_3} - l_{i_1} - s_{j_3}) \\
&= (l_{i_1} - l_{i_2} + l_{i_2} - l_{i_3})(s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2} + s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3}) \\
&= (l_{i_1} - l_{i_2})(s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2}) + (l_{i_2} - l_{i_3})(s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3}) + \\
&\quad (l_{i_1} - l_{i_2})(s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3}) + (l_{i_2} - l_{i_3})(s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2}) \\
&\ge 0 + 0 + (l_{i_1} - l_{i_2})(s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3}) + (l_{i_2} - l_{i_3})(s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2}) \\
&= (l_{i_1} - l_{i_2})(l_{i_2} - l_{i_3})((s_{j_2} + l_{i_3} - l_{i_2} - s_{j_3})/(l_{i_2} - l_{i_3}) + \\
&\quad (s_{j_1} + l_{i_2} - l_{i_1} - s_{j_2})/(l_{i_1} - l_{i_2})) \\
&> 0 \qquad \square
\end{aligned}$$

Lemma 3. *For any three different edges $e_r = (c_{i_r}, s_{j_r})$, $r = 1, 2, 3$ and $loc_{i_1} < loc_{i_2} < loc_{i_3}$, if e_1 conflicts with e_3, then e_2 conflicts with e_3 or e_2 conflicts with e_1.*

Proof. This lemma follows from Lemma directly. $\square$

Lemma says that if one alignment edge e_2 is sandwiched by two alignment edges e_1, e_3 which are in *conflict*, then the sandwiched edge e_2 must be in conflict with at least one of the two conflicted edges e_1, e_3.

Let $D[i]$ denote all valid query sequence positions that c_i could be aligned to. Let $R[i, j, l]$ denote all valid alignment positions of c_j given c_i is aligned to s_l. $\forall k \in R[i, j, l]$, the two edges (c_i, s_l) and (c_j, s_k) do not conflict. See Figure 2 for an example of $D[i]$ and $R[i, j, l]$.

Definition 5. An alignment between the template and the sequence is valid if: (1) it satisfies Definition ; (2) each core of the template is aligned to some position of the (extended) sequence[a]; and (3) For any two different cores c_{i_1} and c_{i_2}, their two alignment edges do not conflict in the alignment graph, i.e., if c_{i_j} is aligned to s_{l_j}, $j = 1, 2$, then $s_{l_1} \in R[i_2, i_1, s_{l_2}]$ and $s_{l_2} \in R[i_1, i_2, s_{l_1}]$.

[a]As mentioned before, global and global-local alignment are employed.

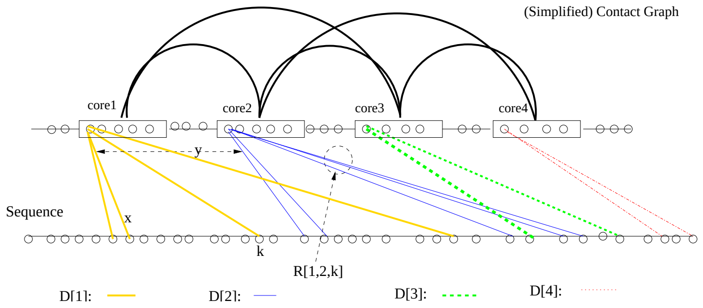

Fig. 2. Example of $D[i]$ and $R[i,j,l]$.

Figure 1 contains an example of an alignment.

Let $x_{i,l}$ be a boolean variable such that $x_{i,l} = 1$ if and only if core c_i is aligned to position s_l. Similarly, for any $(c_{i_1}, c_{i_2}) \in \mathcal{E}(CMG)$, let $y_{(i_1,l_1),(i_2,l_2)}$ indicate the pairwise interactions between x_{i_1,l_1} and x_{i_2,l_2} if the two edges $(c_{i_1}, s_{l_1}), (c_{i_2}, s_{l_2})$ do not conflict. $y_{(i_1,l_1),(i_2,l_2)} = 1$ if and only if $x_{i_1,l_1} = 1$ and $x_{i_2,l_2} = 1$. We say $y_{(i_1,l_1),(i_2,l_2)}$ is generated by x_{i_1,l_1} and x_{i_2,l_2}. The x variables are called the alignment variables and y variables are called the interaction variables.

Now the objective function of the protein threading problem can be formulated as an integer program as follows.

$$\min W_m E_m + W_s E_s + W_p E_p + W_g E_g + W_{ss} E_{ss}, \tag{1}$$

$$E_m = \sum_{i=1}^{M} \sum_{l \in D[i]} [x_{i,l} \times \sum_{r=0}^{len_i - 1} Mutation(head_i + r, l + r)], \tag{2}$$

$$E_s = \sum_{i=1}^{M} \sum_{l \in D[i]} [x_{i,l} \times \sum_{r=0}^{len_i - 1} Fitness(head_i + r, j + r)], \tag{3}$$

$$E_{ss} = \sum_{i=1}^{M} \sum_{l \in D[i]} [x_{i,l} \times \sum_{r=0}^{len_i - 1} SS(head_i + r, j + r)], \tag{4}$$

$$E_p = \sum_{1 \le i < j \le M, (c_i, c_j) \in E(CMG)} \sum_{l \in D[i]} \sum_{k \in R[i,j,l]} y_{(i,l),(j,k)} P(i,j,l,k), \tag{5}$$

$$P(i,j,l,k) = \sum_{u=0}^{len_i - 1} \sum_{v=0}^{len_j - 1} \delta(head_i + u, head_j + v) Pair(l + u, k + v) \tag{6}$$

$$E_g = \sum_{i=1}^{M} \sum_{l \in D[i]} \sum_{k \in R[i,i+1,l]} y_{(i,l),(i+1,k)} G(i,l,k), \tag{7}$$

where $\delta(u,v) = 1$ if there is an interaction between residues at positions u and v in the template, otherwise 0. $G(i,l,k)$ is the gap penalty between c_i and c_{i+1} when they are aligned to query sequence positions l and k respectively. $G(i,l,k)$ could be computed by dynamic programming in advance given i,l,k.

The constraint set is as follows:

$$\sum_{j\in D[i]} x_{i,j} = 1, \quad i = 1,2,\ldots,M; \tag{8}$$

$$\sum_{l\geq l_0, l\in D[i]} x_{i,l} + \sum_{k\in D[i+1]-R[i,i+1,l_0]} x_{i+1,k} \leq 1, \quad l_0 \in D[i],\ i = 1,2,\ldots,M-1; \tag{9}$$

$$\sum_{k\in R[i,j,l]} y_{(i,l),(j,k)} \leq x_{i,l}, \quad \forall l \in D[i],\ i,j = 1,2,\ldots,M; \tag{10}$$

$$\sum_{l\in R[j,i,k]} y_{(i,l),(j,k)} \leq x_{j,k}, \quad \forall k \in D[j],\ i,j = 1,2,\ldots,M; \tag{11}$$

$$\sum_{k\in R[i,j,l]} y_{(i,l),(j,k)} \geq x_{i,l} + \sum_{k\in R[i,j,l]} x_{j,k} - 1, \quad l \in D[i],\ i,j = 1,2,\ldots,M; \tag{12}$$

$$\sum_{l\in R[j,i,k]} y_{(i,l),(j,k)} \geq x_{j,k} + \sum_{l\in R[j,i,k]} x_{i,l} - 1, \quad k \in D[j],\ i,j = 1,2,\ldots,M; \tag{13}$$

$$x_{i,j} \in \{0,1\}, \quad j \in D[i],\ i = 1,2,\ldots,M; \tag{14}$$

$$y_{(i,l)(j,k)} \in \{0,1\}, \quad \forall l \in D[i], k \in D[j],\ i,j = 1,2,\ldots,M. \tag{15}$$

Constraint 8 says that one core can be aligned to a unique sequence position, i.e., given core i, only one of the $x_{i,j}$'s is 1, for $j \in D[i]$. Constraint 9 forbids the conflicts between the adjacent two cores. Based on the transitivity of *non-conflict* (see Lemma), this constraint guarantees that there are no conflicts between any two cores if variable x and y are integral. Therefore, it guarantees that the integral solution corresponds to a valid alignment. Constraint 10 and 11 say that at most one interaction variable can be 1 between any two cores that have interactions between them. Constraint 12 and 13 enforce that if two cores have their alignments to the sequence respectively and also have interactions between them, then at least one interaction variable should be 1. Constraint 14 and 15 guarantee x and y variables to be either 0 or 1. The integer program as formulated above does not have a computationally feasible solver. Hence we relax the integral Constraint 14 and 15 to the following real value constraints.

$$x_{i,j} \geq 0, \quad j \in D[i],\ i = 1,2,\ldots,M; \tag{16}$$

$$y_{(i,l)(j,k)} \geq 0, \quad \forall l \in D[i], k \in D[j],\ i,j = 1,2,\ldots,M. \tag{17}$$

Constraint 8, 16 and 17 guarantee x and y to be between 0 and 1. They will be used to replace Constraint 14 and 15.

There is another set of more obvious constraints which can replace Constraint 9-13. They are:

$$x_{i,l} + x_{i+1,k} \leq 1, \quad \forall k \in D[i+1] - R[i,i+1,l]; \tag{18}$$

$$y_{(i,l)(j,k)} \le x_{i,l}, \quad k \in R[i,j,l],\ (c_i, c_j) \in \mathcal{E}(CMG); \tag{19}$$

$$y_{(i,l)(j,k)} \le x_{j,k}, l \in R[j,i,k], \quad (c_i, c_j) \in \mathcal{E}(CMG); \tag{20}$$

$$y_{(i,l)(j,k)} \ge x_{i,l} + x_{j,k} - 1, \quad (c_i, c_j) \in \mathcal{E}(CMG); \tag{21}$$

Constraint 18 forbids the conflict between the alignments of two neighboring cores. It guarantees that there is no conflict between the alignments of any two cores based on Lemma . Constraint 19-21 guarantee that one interaction variable is 1 if and only if its two generating x variables are 1 simultaneously. Constraint 18-21 can be inferred from Constrain 9-13. Conversely, the above inference relationship is not true. Therefore, Constraint 18-21 are weaker than Constraint 9-13.

In order to improve running time, we found yet another set of Constraint 22 and 23 from which both 9-13 and 18-21 can be inferred.

$$\sum_{k \in R[i,j,l]} y_{(i,l)(j,k)} = x_{i,l}, \quad (c_i, c_j) \in \mathcal{E}(CMG); \tag{22}$$

$$\sum_{l \in R[j,i,k]} y_{(i,l)(j,k)} = x_{j,k}, \quad (c_i, c_j) \in \mathcal{E}(CMG); \tag{23}$$

Constraint 22 and 23 imply that one x variable is 1 is equivalent to that one of the y variables generated by it is 1. It means that if two interacted cores i and j are aligned to two sequence positions l and k respectively, then the interaction score between these two sequence segments (starting from l and k respectively) should be counted in the energy function value. These two are the strongest constraints. Experimental results show that our algorithm with Constraint 22 and 23 (combining with Constraint 8, 16 and 17) runs significantly faster. The correctness of this formulation is not so obvious, we will prove it at the end of this section (see Lemma).

Let $CS1$ denote the constraint set formed by Constraint 8-13, 16 and 17; $CS2$ the constraint set formed by Constraint 8,16,17 and 18-21; $CS3$ the constraint set formed by Constraint 8, 16, 17 and 22-23. Then we have the following relationships.

Lemma 4. *Constraint set $CS1$ is implied by $CS3$.*

Proof. From the fact that $\forall i, R[i,j,l] \subseteq D[i]$, it is easily seen that Constraint 10-13 are implied by Constraint 22 and 23. According to the following equation, Constraint 9 is also implied by $CS3$.

$$\begin{aligned}
& \sum_{l \ge l_0, l \in D[i]} x_{i,l} + \sum_{k \in D[i+1] - R[i,i+1,l_0]} x_{i+1,k} \\
= & \sum_{l \ge l_0, l \in D[i]} x_{i,l} + \sum_{k \in D[i+1] - R[i,i+1,l_0]} \Big(\sum_{l \in R[i+1,i,k]} y_{(i,l)(i+1,k)} \Big) \\
\le & \sum_{l \ge l_0, l \in D[i]} x_{i,l} + \sum_{l \in D[i], l < l_0} \Big(\sum_{k \in R[i,i+1,l]} y_{(i,l)(i+1,k)} \Big) \\
\le & \sum_{l \ge l_0, l \in D[i]} x_{i,l} + \sum_{l \in D[i], l < l_0} x_{i,l}
\end{aligned}$$

$$= \sum_{l \in D[i]} x_{i,l}$$
$$= 1 \qquad \square$$

Lemma 5. *Constraint set $CS2$ is implied by $CS1$.*

Proof. Constraint 18 is implied by Constraint 9 and 16. Constraint 19-20 are implied by Constraint 10-11 and 17. Constraint 21 is implied by Constraint 12-13 and 19-20. □

We also have the following lemma about the correctness of our IP formulation.

Lemma 6. *Each integral solution of $CS1$, $CS2$, $CS3$ corresponds to a valid alignment between a template and a sequence.*

Proof. First, we prove that any integral solution of $CS2$ corresponds to a valid alignment. According to Constraint 18, for any two adjacent cores i, $i + 1$, if they are aligned to sequence positions l, k respectively, then these two alignment edges (c_i, s_l) and (c_{i+1}, s_k) do not conflict. Based on the transitivity of nonconflict (see Lemma), the two alignment edges of any two cores wouldn't conflict, i.e, any integral solution of $CS2$ corresponds to a valid alignment.

Based on Lemma and , Constraint 18 is implied by $CS1$ and $CS3$, therefore, each integral solution of $CS2$ and $CS3$ corresponds to a valid alignment. □

So far we have presented three kinds of linear (integer) program formulations (the objective function combining with each of $CS1$, $CS2$ and $CS3$) to formulate the target sequence-template alignment problem. We have also proved that constraint set $CS3$ is the strongest when x and y variables are relaxed to be real between 0 and 1. Another advantage of constraint set $CS3$ is that the number of non-zero elements in the constraint matrix employed by the linear program solver is also the smallest. RAPTOR uses $CS3$ by default.

4. RAPTOR – Implementation

4.1. *Scoring System*

We calculated the averaged energy over a set of homologous sequences, as used in PROSPECT-II[15]. Given a query sequence of length n, an $n \times 20$ frequency matrix $PSFM$ is calculated by using PSI-BLAST[16] with maximum iteration number being set to 5. Each column of this matrix describes the occurring frequency of 20 amino acids at this position. Assume a template position i is aligned to the sequence position j. Then the mutation score and fitness score are calculated as follows.

$$Mutation(i, j) = \sum_a p_{j,a} M(t_i, a),$$

$$Fitness(i,j) = \sum_a p_{j,a} F(env_i, a),$$

where $p_{j,a}$ represents the occurring frequency of amino acid a at sequence position j, $M(a,b)$ represents the mutation potential as in PAM250 matrix[17] between two amino acids a and b, and $F(env, a)$ denotes the fitness potential when amino acid a is placed into environment env.

Two groups of structural features have been selected to describe the local environment env of a position in the template: (a) Secondary structure. Three classes were defined: α-helix, β-strand and irregular structure (coil). (b) Solvent accessibility (sa). Three levels were defined: buried (inaccessible), intermediate, and accessible. The boundary between different solvent accessibility levels were determined by Equal-Frequency discretization method, i.e, residues in the database are equally distributed within each level. The boundaries between three solvent accessibility levels are at 7% and 37%. Secondary structure and solvent accessibility assignments are generated by DSSP package[18]. The combination of these two features gives 9 local structural environments in total.

The gap penalty function is assumed to be an affine function, i.e., a gap open penalty plus a length-dependent gap extension penalty. Gap open penalty is set at 10.6 and gap extension penalty is 0.8 per single gap[19].

$SS(i,j)$ is defined to be the difference between the template secondary structure at position i and the predicted sequence secondary structure at position j. Any of the good secondary structures prediction programs, for example PSIPRED [20], can be used to predict the secondary structure of the query sequence.

If the two ends of an interaction are aligned to j_1^{th} and j_2^{th} positions of the query sequence respectively, then the pair score for this interaction is given by:

$$Pair(j_1, j_2) = \sum_a p_{j_1,a} \sum_b p_{j_2,b} P(a,b),$$

where $P(a,b)$ denotes the pairwise interaction potential between two amino acids a and b. F and P are taken from PROSPECT-II[15].

4.2. *Branch-and-Bound Method*

We use a branch-and-bound algorithm to solve the integer programming problem, defined in Section 3. In Section 3, we have relaxed the original integer program Constraint 14 and 15 to the linear program Constraint 16 and 17, allowing x and y to be real values between 0 and 1. We first solve the resulting linear program, using for example the powerful IBM OSL (Optimization and Solution Library) package. If the solution (x^*, y^*) of the linear program is integral, then that is the optimal solution for the corresponding integer program problem. Otherwise, we select one non-integral variable and generate two subproblems by setting the variable to 0 and 1, respectively. These two subproblems are solved recursively.

A general introduction to solving integer programming problems can be found in Wolsey's book[21].

4.3. *Weight Training*

The weight factors $W_m, W_s, W_{ss}, W_g, W_p$ are chosen by optimizing the overall alignment accuracy on our training set. The optimal alignment accuracy does not necessarily imply the best fold recognition capability though. In the following subsection, an SVM (Support Vector Machine) method is used to rank structural folds for fold recognition. A set of 95 structurally-aligned protein pairs are chosen from Holm and Sander's test set[22] as the training samples, each of which has only fold-level similarity. The alignments generated by RAPTOR is compared with the structural alignments generated by SARF[23]. An alignment for a residue is regarded as correct if it is within 4 residue shift away from the structure-structure alignment by SARF. The overall alignment accuracy is defined as the ratio between the number of the correctly-aligned positions of all threading pairs and the number of the maximum alignable positions. Our objective is to maximize the overall alignment accuracy. A genetic algorithm plus a local pattern search method implemented in DAKOTA[24] is used to search for the optimal weight factors. We attained 56% alignment accuracy over this set of training pairs. A set of 1100 protein pairs which are in the fold-level similarity is also generated from Holm and Sander's test set[22] to test the weight factors and 50% alignment accuracy is attained. We have also selected 95 structurally-aligned protein pairs from Holm and Sander's test set, each of which is in superfamily-level or family-level similarity, 80% alignment accuracy is achieved when the same set of weight factors is used.

4.4. *z-score and Fold Recognition*

After threading a sequence onto a template, a z-score is calculated according to the method proposed in paper by Bryant and Altschul[25] to cancel out the composition bias. Let z_{raw} denote this kind of z-score. Since the accurate z_{raw} is expensive to compute, we just approximate it by (i) fixing the optimal sequence-template alignment generated by our IP approach; (ii) shuffling the amino acids in the query sequence randomly; (iii) calculating the shuffled alignment scores based on the given alignment positions for 1000 times rather than doing optimal alignments again and again; (iv) calculating the mean $score_{mean}$ and standard deviation $score_{sd}$ of the scores generated in (iii); (v) z_{raw} is defined to be $\frac{score_{mean} - score_{opt}}{score_{sd}}$ where $score_{opt}$ is the optimal alignment score generated by our IP approach. All templates are sorted according to the z_{raw} in descending order.

An SVM with RBF kernel is employed to adjust the approximate z-score. Vapnik's book[26] contains a comprehensive tutorial of SVM. There are several free SVM software such as SVM light[27]. A set of 60000 training pairs formed by all-against-all threading between 300 templates (randomly chosen from the FSSP database) and 200 sequences (randomly chosen from Holm and Sander's test set [22]) is used as the training samples of our SVM model. The relationship between two proteins is judged based on SCOP database[2]. If one pair is in at least fold-level similarity, then it is treated as a positive example, otherwise a negative example. Each of the training samples consists of the following features: (1)z_{raw}; (2) the sequence size; (3) the template size; (4) the number of cores in the template; (5) the sum of the core sizes in the template; (6) the number of aligned cores; (7)

the number of aligned positions; (8) the number of identical residues; (9) the number of contacts with both ends on the aligned cores; (10) the number of cut contacts with one end on the aligned cores and the other on the unaligned cores; (11) the total score; (12) mutation score; (13) singleton fitness score; (14) gap score; (15) secondary score; and (16) pair score. Given one threading result, SVM outputs a real value. The value greater than 0 means this threading pair is in at least fold-level similarity. We do not use this directly due the abundance of the false negatives. We calculate the final z-score for each query sequence. For all threading pairs of one given sequence, let $o_1, o_2, ..., o_q$ denote the outputs from SVM model. The final z-score is calculate by $\frac{o_i - u(o)}{std(o)}$, where $u(o)$ is the mean value of o_i and $std(o)$ is the standard deviation of o_i. Daniel Fischer *et al* benchmark[14] is used to fix the parameters of the SVM model. Again, the template with the biggest z-score is chosen as the best-fit one for the target sequence. Experimental results show that after using SVM approach the fold recognition capability is improved by about 5%.

5. Experimental Results

In this section, we present some experimental results on RAPTOR in terms of fold recognition and alignment accuracy. Two classes of results are presented here. One is the large scale benchmark test by the authors, and the other one is a blind test organized by the structure prediction community, called CAFASP. We also present the experimental structures and the prediction structures of several specific targets from CAFASP3 and LiveBench-6[28].

5.1. *Benchmark Tests*

Fischer *et al*'s benchmark consists of 68 target sequences and 301 templates. RAPTOR ranks 56 pairs out of 68 pairs as top 1, achieving about 82% prediction rate.

The fold recognition performance of RAPTOR was further tested on Lindahl's benchmark set consisting of 976 protein sequences [29]. By threading them all against all, there are totally 976×975 threading pairs. We measured RAPTOR's performance in three different similarity levels: fold, superfamily and family. The results are shown in Table 5.1. The results of other methods are taken from Shi *et al*'s paper[5]. The correctness of the prediction is assessed based on the SCOP classification.

As shown in Table 5.1, the performance of RAPTOR at all similarity levels is better than the others, and much better at the fold level. At the family level, RAPTOR's recognition performance is comparable to that of FUGUE [5], the best method for family and superfamily level except RAPTOR. Thus, we may conclude that a strict treatment of the pairwise interactions is necessary for fold level and superfamily level recognition. For family level recognition, sequence (or profile) alignment could attain satisfactory results.

5.2. *CAFASP3 Evaluation*

We now present RAPTOR's performance in CAFASP3[30] (The Third Critical Assessment of Fully Automated Structure Prediction). The goal of CAFASP is to evaluate the state-

Table 1. The performance of RAPTOR at three different similarity levels

	Family		Superfamily		Fold	
method	Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
RAPTOR	**84.8**	**87.1**	**47.0**	**60.0**	**31.3**	**54.2**
FUGUE	82.2	85.8	41.9	53.2	12.5	26.8
PSI-BLAST	71.2	72.3	27.4	27.9	4.0	4.7
HMMER-PSIBLAST	67.7	73.5	20.7	31.3	4.4	14.6
SAMT98-PSIBLAST	70.1	75.4	28.3	38.9	3.4	18.7
BLASTLINK	74.6	78.9	29.3	40.6	6.9	16.5
SSEARCH	68.6	75.7	20.7	32.5	5.6	15.6
THREADER	49.2	58.9	10.8	24.7	14.6	37.7

of-the-art of protein structure prediction servers available to the community. Since its inception in 1998, CAFASP has been held for three times, jointly with CASP(Critical Assessment of Structure Prediction) conferences every other year. CAFASP3 was jointly held with CASP5[31]. The difference of CAFASP and CASP is that CAFAST allows only authomatic prediction and while CASP allows manual improvements and also with all the CAFASP predictions available from all servers. CAFASP evaluation is a blind test, administrated by the CAFASP committee. CAFASP is also the largest scale blind test of structure prediction servers in the community. The experimental structures of the test sequences were unknown to the predictors before the end of the test. CAFASP3 makes use of MaxSub[32] to evaluate the alignment accuracy. The recognition is considered correct only if the alignment accuracy is above some level. For detailed evaluation rules, see CAFASP3 website. Table 2 shows the ranking of all servers in term of fold recognition sensitivity of FR targets. Table 3 shows the ranking of all servers in terms of hard HM target recognition sensitivity. In these two tables, those servers whose names are printed in italic are meta-servers[b], as classified by the CAFASP3 organizers. As shown Table 2, RAPTOR ranks top 1 among all individual servers in terms of both alignment accuracy and fold recognition capability for the FR targets. RAPTOR also ranks top in hard homology modeling target recognition. FR targets and hard homology modeling targets are the main focuses of the research community because the easy family homology targets are relatively easily recognized by almost all programs.

In addition to the sensitivity of servers, the specificity is also very important for high-throughput automatic structure prediction programs. Table 4 shows the specificity of all CAFASP3 servers on 33 targets. These 33 targets consist of all FR targets and Hard HM targets, but a multidomain FR target is treated as one target here while it is regarded as multiple targets in sensitivity evaluation. From this table, we can see the specificity of RAPTOR is good but not very good. Actually, RAPTOR's specificity is underestimated because the scale of the reliability score of RAPTOR changed dramatically during the

[b]Meta-servers refer to those servers which do consensus predictions based on the outputs of other servers.

Table 2. CAFASP3 Evaluation: RAPTOR performance on 30 Fold Recognition Targets.

Rank	Servers	Sum MaxSub Score	Number of Correct Recognitions
1	*3ds5 robetta*	5.17-5.25	15-17
3	*pmod 3ds3 pmode3*	4.21-4.36	13-14
4	**raptor**	**3.98**	**13**
5	shgu	3.93	13
6	*3dsn* orfeus	3.64-3.90	12-13
7	*pcons3*	3.75	12
8	fugu3 orf_c	3.39-3.67	11-12
10	fugsa orf_b	3.44-3.63	10-12
...	...	...	...
48	pdbblast	0.00	0
...	...	...	...
54	blast	0.00	0

Table 3. CAFASP3 Evaluation: RAPTOR performance on 12 Hard HM Targets.

Rank	Servers	Score
1	*3ds5*	5.13
2	*3ds3* shgu	4.93-5.02
4	*pmod pmod3*	4.60-4.68
6	orfeus orfb 3dpsm **raptor** fugu3 *pco3 robetta*	4.33-4.43
8	samt02	4.18
...	...	...
55	cmap	0

competition. In the first month of CAFASP3, the reliability score generated by RAPTOR is the original output of SVM model, but in the following months, the reliability score is the standardized output of SVM model.

5.3. *Specific Examples*

Here, we present some beautiful structure prediction examples generated by RAPTOR in CAFASP3 evaluation and LiveBench-6 test[28]. Most of experimental structures of the CAFASP3 targets are not allowed to be published so far, therefore we choose some targets from LiveBench.

Figure 3 presents the superimposition between the experimental structure (grey color) and the RAPTOR's predicted structure (black color) of T0136_1. This figure is taken from CAFASP3 website. It is generated by RasMol and MaxSub. According to the evaluation of MaxSub, 17 of 54 servers generated correct fold recognitions for this target. RAPTOR produced the best alignment among all. MaxSub could superimpose a segment of 118

Table 4. CAFASP3 Evaluation: RAPTOR's specificity on 33 Targets consisting of FR targets and Hard HM targets. But a multidomain FR target is counted only once.

Servers	specificity
3ds5	24.8
pmodel 3ds3 3dsn pmode3	22.0-22.6
pcons shgu	21.4-21.6
inbgu fugu3	19.0-19.8
ffas03 fugsa orfeus	18.2-18.4
raptor 3dpsm orf_c	17.4-17.8
...	...
pdbblast	13.0
...	...
blast	4

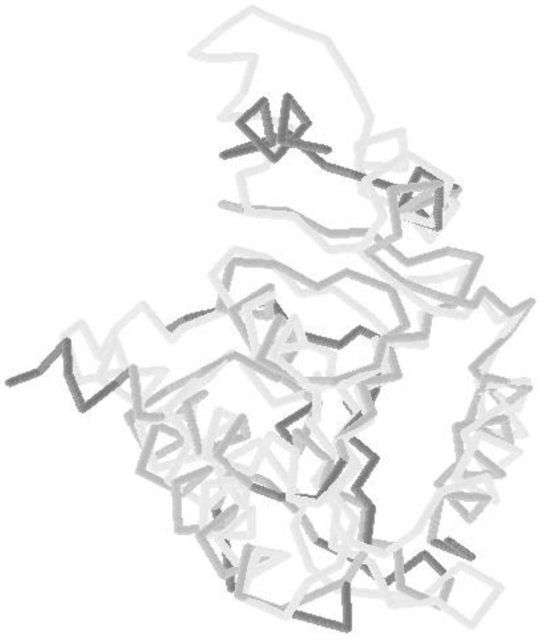

Fig. 3. The superimposition of experimental structure (grey color) and prediction structure (black color) of CAFASP3 target T0136_1.

residues (the sequence size is 144) of the predicted structure to the experimental structure and the RMSD is just 1.9A.

The following two figures are generated by RasMol based on the evaluation results of LiveBench-6. Figure 4 shows almost a perfect structure prediction for target 1ll8A. The alignment accuracy score measured by MaxSub is more than 9 (scale 10). Figure 5 presents a good structure prediction for target 1j53A. The alignment accuracy score measured by MaxSub is more than 6. Considering the length of the target sequence, this prediction is also quite successful.

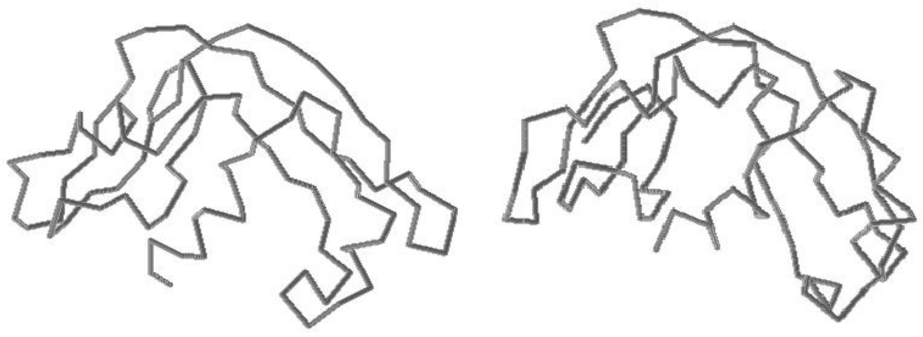

Fig. 4. The experimental structure (left) and the predicted structure (right) of 1kvzA.

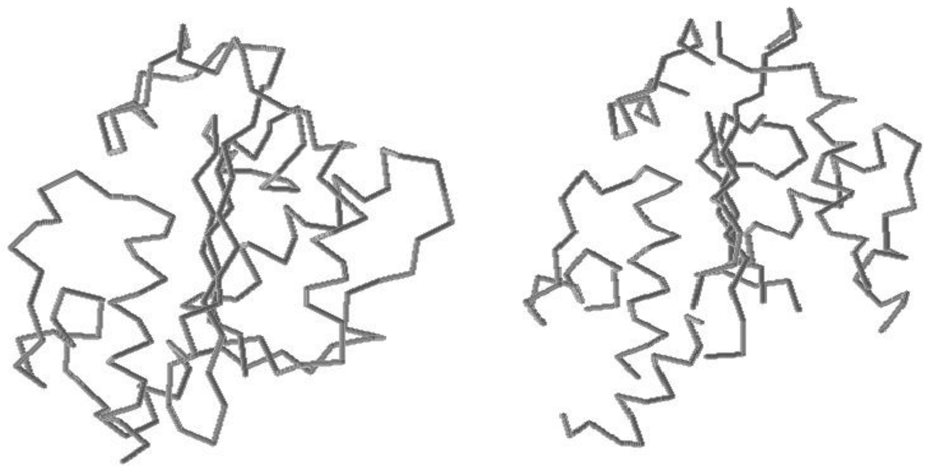

Fig. 5. The experimental structure (left) and the predicted structure (right) of 1j53A.

6. Computing Efficiency Issues

A key advantage of our algorithm is that the memory requirement is just about $O(|\mathcal{E}|n^2)$ (there are $O(|\mathcal{E}|n^2)$ active elements in the sparse matrix of the linear program), where $\mathcal{E}$ is the edge set of the contact graph of a protein template structure and n query sequence length. The observed memory usage of RAPTOR is $100 \sim 200M$ for most of threading pairs. Furthermore, in practice, the computing time does not increase exponentially with respect to the target sequence size. Figure 6 shows the CPU time of threading 100 sequences (chosen randomly from Lindahl's benchmark) with size ranging from 25 to 572 to a typical template 119l of length 162 (and 12 cores)[c]. It shows that the computational time of our algorithm increases very slowly with respect to the sequence size. In fact, we found out that for the real protein data, our relaxed linear programs directly outputted the integral solutions 99% of the times and generated only a few branch nodes when the solution was fractional[33].

Figure 7 shows the CPU time used for the prediction of each CAFASP3/CASP5 target sequence. There are totally 62 targets and totally 3236 protein templates in our template database. As shown in this figure, the CPU time increases very slowly with respect to the sequence size except that it takes about 45 hours for one target(t0174). After carefully examining the running time of threading t0174 to each template, we find out that there are 30 templates, for which it takes about 15 hours to thread t0174 to them. We plan to investigate these templates further.

7. Conclusions

In this paper, we have presented a novel integer programming approach to treat pairwise interactions rigorously in protein threading and some implementation details of our software package RAPTOR. Experimental results show that RAPTOR performs very well in terms of both alignment accuracy and fold recognition for FR targets. After carefully examining the performance of RAPTOR, we found out that if the best of the top 5 or top 10 predictions is evaluated, RAPTOR could recognize many more fold-level targets. A big challenge would be to design a better energy function or a better machine learning algorithm to boost the correct templates from top 5 or 10 to top 1. In terms of computational resource consumption, RAPTOR is also much better than those algorithms which treat the pairwise potentials strictly when dealing with the templates with complex interaction topology and long sequences. As mentioned before, the divide-and-conquer algorithm employed by PROSPECT-I could deal well with the templates with simple interaction topology. A future work would be to incorporate the divide-and-conquer idea into our IP approach to take advantage of the partial sparseness of the complex template interaction topology such that the computational time could be improved further.

[c]In this section, the CPU time is measured in a single CPU of a Silicon Graphics Origin 3800 system, which has 40 400 MHz MIPS R12000 CPUs and 20 GB of RAM.

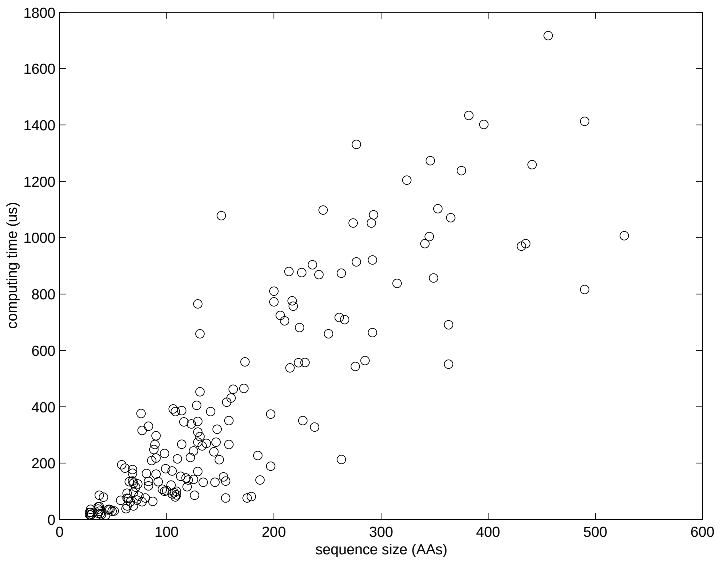

Fig. 6. CPU time of threading 100 sequences to template 119l (1us=0.01s).

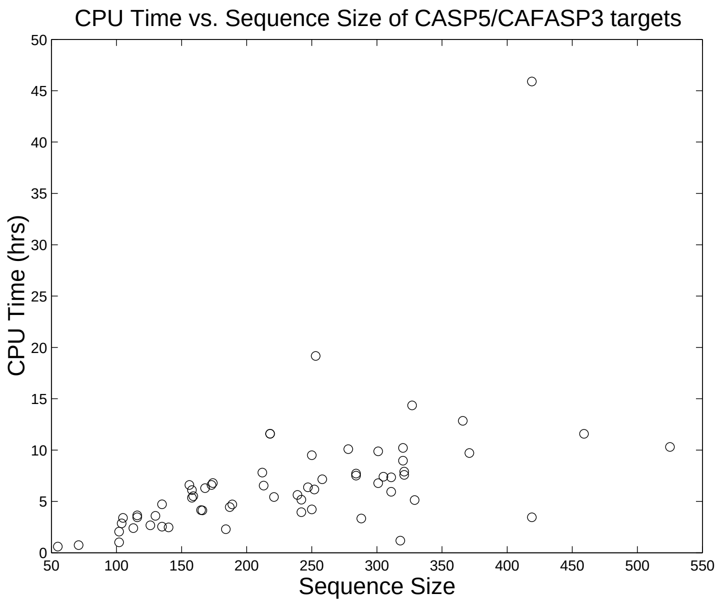

Fig. 7. CPU time of threading 62 CAFASP3 target sequences to 3236 templates.

8. Acknowledgments

The authors would like to thank Professor Guohui Lin and Dr. Dong Xu for their collaboration on related projects. Jinbo Xu was partially supported by NSERC grant OGP0046506. Ming Li was partially supported by NSERC grant OGP0046506, the Killam Fellowship, and the NSERC research chair program.

References

1. L. Holm and C. Sander. Mapping the protein universe. *Science*, 273:595–602, 1996.
2. A.G. Muzrin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP:a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.
3. C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. CATH-a hierarchic classification of protein domain structures. *Structure*, 5:1093–1108, 1997.
4. F.M.G. Pearl, D. Lee, J.E. Bray, I. Sillitoe, A.E. Todd, A.P. Harrison, J.M. Thornton, and C.A. Orengo. Assigning genomic sequences to CATH. *Nucleic Acids Research*, 28:277–282, 2000.
5. J. Shi, L. B. Tom, and M. Kenji. FUGUE: Sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. *J. Mol. Biol.*, 310:243–257, 2001.
6. L.A. Kelley, R.M. MacCallum, and M.J.E. Sternberg. Enhanced genome annotation using structural profiles in the program 3D-PSSM. *J. Mol. Biol.*, 299(2):499–520, 2000.
7. D.T. Jones. GenTHREADER: An efficient and reliable protein fold recognition method for genomic sequences. *J. Mol. Biol.*, 287:797–815, 1999.
8. R.H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Engineering*, 7:1059–1068, 1994.
9. D.T. Jones, W.R. Taylor, and J.M. Thornton. A new approach to protein fold recognition. *Nature*, 358:86–98, 1992.
10. A. Godzik, A. Kolinski, and J. Skolnick. A topology fingerprint approach to inverse protein folding problem. *J. Mol. Biol.*, 227:227–238, 1992.
11. S. Bryant. Evaluation of threading specitity and accuracy. *Proteins: Struct. Funct. Genet.*, 26:172–185, 1996.
12. T. Akutsu and S. Miyano. On the approximation of protein threading. *Theoretical Computer Science*, 210:261–275, 1999.
13. Y. Xu, D. Xu, and E.C. Uberbacher. An efficient computational method for globally optimal threadings. *Journal of Computational Biology*, 5(3):597–614, 1998.
14. D. Fischer, A. Elofsson, J.U. Bowie, and D. Eisenberg. Assessing the performance of fold recognition methods by means of a comprehensive benchmark. pages 300–318, Singapore, 1996. Biocomputing: Proceedings of the 1996 Pacific Symposium, World Scientific Publishing Co.
15. D. Kim, D. Xu, J. Guo, K. Ellrott, and Y. Xu. Prospect ii: Protein structure prediction method for genome-scale applications. 2002. submitted.
16. S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.
17. R.M. Schwartz and M.O. Dayhoff. *Matrices for detecting distant relationships*, pages 353–358. Natl. Biomed. Res. Found., 1978.
18. W. Kabsch and C. Sander. Dictionary of protein secondary structure: protein recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22:2577–2637, 1983.
19. G.H. Gonnet, M.A. Cohen, and S.A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

20. D.T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.*, 292:195–202, 1999.
21. L.A. Wolsey. *Integer Programming*. JOHN WILEY & SON Inc., 1998.
22. L. Holm and C. Sander. Decision support system for the evolutionary classification of protein structures. *ISMB*, 5:140–146, 1997.
23. N.N. Alexandrov. SARFing the PDB. *Protein Engineering*, 9:727–732, 1996.
24. M.S. Eldred, A.A. Giunta, B.G. van Bloemen Waanders, S.F. Wojtkiewicz, W.E. Hart, and M.P. Alleva. DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis. version 3.0 user manual. Technical Report SAND2001-3796, Sandia, 2002.
25. S.H. Bryant and S.F. Altschul. Statistics of sequence-structure threading. *Curr. Opin. Struct. Biol.*, 5:236–244, 1995.
26. V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
27. T. Joachims. *Making Large-scale SVM Learning Practical*. MIT Press, 1999.
28. L. Rychlewski. http://www.bioinfo.pl/livebench/6, 2002.
29. E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *J. Mol. Biol.*, 295:613–625, 2000.
30. D. Fischer. http://www.cs.bgu.ac.il/˜dfischer/CAFASP3/, December 2002.
31. CASP5. http://predictioncenter.llnl.gov/casp5/Casp5.html, December 2002.
32. N. Siew, A. Elofsson, L. Rychlewski, and D. Fischer. Maxsub: An automated measure for the assessment of protein structure prediction quality. *Bioinformatics*, 16(9):776–785, 2000.
33. J. Xu, M. Li, and Y. Xu. Protein threading by linear programming: Part 2, theoretical analysis and practical results. 2003. submitted.