# Assessment of RAPTOR's Linear Programming Approach in CAFASP3

Jinbo Xu, Ming Li
Department of Computer Science, University of Waterloo, Waterloo, Canada N2L 3G1
{j3xu, mli}@uwaterloo.ca

## ABSTRACT

We have developed a new algorithm based on the mathematical theory of linear programming (LP) and implemented it in our program RAPTOR. Our new approach provides an elegant formulation of the protein threading problem, overcomes the intractability problem of protein threading, in practice, and allows us to use existing powerful linear programming software to obtain optimal protein threading solutions. CASP5 and CAFASP3 gave us the first chance to test RAPTOR in an unbiased way. RAPTOR was ranked as the top individual (automatic) server for fold recognition by the CAFASP3 organizers. In this short paper, we describe RAPTOR's LP formulation, assess RAPTOR's performance in CAFASP3/CASP5, explain why it has superceded other existing automatic individual methods, and point out its strengths, limitations, extensions and prospects for improvement.

## Keywords

Protein Threading, Integer Programming, Support Vector Machines

## 1. INTRODUCTION

Protein three-dimensional (3D) structures may be determined experimentally or predicted computationally. Experimental methods such as x-ray crystallography or nuclear magnetic resonance spectroscopy (NMR) are accurate but costly and slow. While computational solutions are inexpensive, they are not yet completely accepted by protein scientists due to their low success rates. Despite such problems, software packages for protein 3D structure prediction have already been used to some extent in the pharmaceutical industry for distant homology search and preliminary protein structure modeling. This practice is spreading as new and better methodologies are developed for protein structure prediction and more solved protein structures are stored in the databases. We believe that eventually the computational approach will become the dominant method for protein structure determination.

Currently, protein threading is one of the most practical computational approaches for protein structure prediction. Essentially, given a protein sequence whose structure we want to determine, the threading method tries to match the given amino acid sequence against each solved protein structure in the protein databases to decide how to fold the given protein sequence. However, it is known that protein threading is NP-hard [1], which implies that finding the optimally matched template is unlikely to be computable in polynomial time. Many programs try to avoid such computation by not considering pairwise contacts of amino acids because without them the problem becomes tractable. However, pairwise contacts between some key amino acids are important for FR targets, where the sequence homology levels are usually very low.

RAPTOR, RApid Protein Threading predictOR, was designed to solve such problems. In RAPTOR, we consider all constraints, including pairwise contacts of amino acids. We have formulated the protein threading problem as an integer programming (IP) problem. However, integer programming is still known to be NP-hard [2]. Thus, we have relaxed our IP formulation to a linear programming (LP) formulation by allowing solutions to be non-integral. LP systems can be solved optimally and efficiently by existing powerful software packages. We have further designed our LP formulation so that the solutions are integral 99% of the times, in practice. Integral solutions for the LP system are precisely the solutions for the original IP system.

We first describe our formulation for the biological audience in Section 2. We will then assess RAPTOR's performance in CAFASP3/CASP5, explain why it supercedes other existing automatic individual methods in Section 3, and point out its strengths, limitations, extensions and prospects for improvements in Section 4.

## 2. MATERIALS AND METHODS

Here we describe our new method for protein threading for non-mathematicians. We refer the readers to [3, 4] for a full mathematical study.

### Linear Programming

A Linear Programming (LP) problem is a special case of the so-called Mathematical Programming problem. A mathematical program tries to identify an extreme (i.e., minimum or maximum) point of a function $f(x_1, \ldots, x_n)$, which furthermore satisfies a set of constraints, e.g., $g(x_1, \ldots, x_n) \geq$

$b$. When both the objective function $f$ and the problem constraints are linear, we have a linear program. When $x_1, \ldots, x_n$ are required to be integers, we have an integer program (IP). A good reference on integer programming is [5].

Linear programming is a powerful optimization tool. It has many applications in technology, economics, and business planning. An important factor for the applicability of the linear programming methodology in various application contexts is the computational tractability of the resulting analytical models. Theoretically, the first polynomial-time algorithm was not discovered until 1979 by the Russian mathematician Khachiyan. In reality, the Simplex algorithm developed by G. Dantzig in 1947 runs very efficiently in expected time and has been widely used. There are many powerful linear programming software packages available to solve practical problems. Interestingly, the integer programming problem, where the solutions are required to be integral, is known to be NP-hard [2], which means a polynomial-time algorithm is unlikely.

## Protein Threading

Protein threading attempts to match a given query sequence to each structure template and see which one matches the best, predicting the sequence's structure based on the best match. A key to the threading approach is its computational efficiency. It has been proved that protein threading is NP-hard when variable gaps and pairwise interactions are considered simultaneously [1]. Various types of approximation algorithms have been used to optimize the energy function. These methods include double dynamic programming [6], interaction-frozen approximation [7], and the Gibbs sampling algorithm [8]. When the energy function does not include the pairwise interaction preferences explicitly, a simple dynamic programming algorithm can be used to optimize the energy function by aligning the template sequence (profile) to the target sequence (profile). This approach is adopted by FUGUE [9], 3DPSSM [10], and GenTHREADER [11]. The prediction speed is fast and the fold recognition capability is also good for HM targets. However, this is not adequate for FR targets. Exact algorithms have been designed to optimize the energy function that includes the pairwise interaction preferences. Xu *et al.* have proposed a divide-and-conquer method [12], employed by PROSPECT-I, which runs fast on simple protein template (interaction) topologies, but could take a long time for protein templates with dense residue-residue interactions and long target sequences.

## Alignment Model

Given a template protein sequence $t_1 t_2 \ldots t_m$, of length $m$, and a query sequence $s_1 s_2 \ldots s_n$, of length $n$, an *alignment* between the template and the query sequence is a set of pairs $(\hat{t_1}, \hat{s_1}), (\hat{t_2}, \hat{s_2}), \ldots, (\hat{t_L}, \hat{s_L})$, where $L \leq m + n$, $\hat{t_1}\hat{t_2} \ldots \hat{t_L}$ is an expansion of $t_1 t_2 \ldots t_m$ by inserting some gaps, $\hat{s_1}\hat{s_2} \ldots \hat{s_L}$ is an expansion of $s_1 s_2 \ldots s_n$ 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.

In formulating the protein threading problem, the following basic restrictions to the above general alignment model are assumed: (1) Each template sequence is parsed as a linear series of cores with the connecting loops between the adjacent cores. Each core is the most conserved subsegment of an $\alpha$-helix or $\beta$-sheet secondary structure. Although the secondary structure is often conserved, insertion or deletion may occur within the two ends of a secondary structure. Let $c_i = core(head_i, tail_i)$ denote all cores of one template, where $i = 1, 2, \ldots, M$, $M$ is the number of the cores, and $1 \leq head_1 \leq tail_1 < head_2 \leq tail_2 < \ldots < head_M \leq tail_M \leq m$. The region between $tail_i$ and $head_{i+1}$ is a loop. The length of $c_i$ is $len_i = tail_i - head_i + 1$. When aligning a query protein sequence with a template, alignment gaps are confined to only loops. (2) We consider only contacts (interactions) between core residues. It is generally believed that interactions involving loop residues can be ignored as their contribution to fold recognition is relatively insignificant. We say that a contact exists between two residues if the spatial distance between their $C_\beta$ 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. (3) Global alignment and global-local alignment methods are employed to align one template to one sequence. For the detailed description, see Fischer *et al* [13].

## IP Formulation

Our threading energy function consists of an 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 described in [4].

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 the $i^{th}$ 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. The simplified contact map is used here for the purpose of reducing the number of variables in our IP formulation so that RAPTOR runs efficiently. The unsimplified connectivities, shown as dashed arcs in Figure 1, are eventually taken into account by RAPTOR when determining the valid alignments.

For simplicity, when we say that core $c_i$ is aligned to position $s_j$, we always mean that core $c_i = (head_i, tail_i)$ is aligned to segment $(s_j, s_{j+len_i-1})$.

Consider the alignment bipartite graph of one threading pair. Each core of the template corresponds to one vertex, labeled as $c_i$, $i = 1, 2, \ldots, M$, each residue in the query sequence corresponds to one vertex, labeled as $s_j$, $j = 1, 2, \ldots, n$. If $c_i$ is aligned to $s_j$, we draw an alignment edge in this graph. Informally, for any two different edges $e_1 = (c_{i_1}, s_{j_1})$ and $e_2 = (c_{i_2}, s_{j_2})$, if they cross, we say $e_1$ and $e_2$ are in *conflict*. An alignment edge is valid if it is not in conflict with other alignment edges. An alignment between template and the target sequence is valid if all alignment edges are valid.

Let $D[i]$ denote all valid query sequence positions that $c_i$

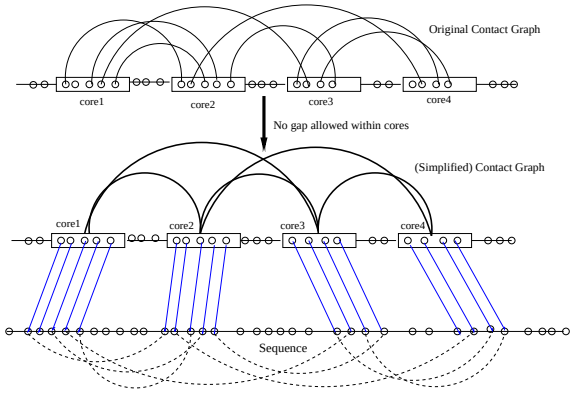


**Figure 1: Template contact graph and an alignment between one template and one sequence. A small circle represents a residue. A solid arc in the original contact graph means that its two end residues have an interaction. A dashed arc means that if two sequence residues are aligned to two template residues having interaction to each other, then the interaction score of these two sequence residues must be counted in the energy function. The interaction score between two segments of the sequence is the sum of the interaction scores of two sequence residues which are aligned by two interacted template residues.**

could be aligned to. Let $R[i,j,l]$ denote all valid alignment positions of $c_j$ given $c_i$ is aligned to $s_l$. Figure 2 gives examples of $D[i]$ and $R[i,j,l]$.

Now we formulate the problem of minimizing the energy function $E$ under the contact map constraints as an IP problem. 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 $x_{i_2,l_2}$.

Now the objective function of the protein threading problem

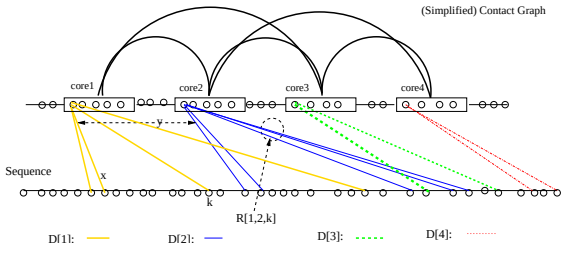


**Figure 2: Example of $D[i]$ and $R[i,j,l]$. $R[1,2,k]$ is the set of potential alignment positions of core 2 given core 1 is aligned to sequence position $k$. Core 1 has five residues which have to be aligned to the sequence based on our assumption in the "Alignment Model" subsection. Thus, the first two candidate alignment positions of core 2 are invalid if core 1 is aligned to position $k$ in order to avoid overlap.**

can be formulated as an IP as follows:

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

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

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

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

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

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

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

where $\delta(u,v) = 1$ if there is an interaction between residues at position $u$ and $v$ in the template, otherwise 0. $G(i,l,k)$ is the gap potential between $c_i$ and $c_{i+1}$ when they are aligned to target sequence position $l$ and $k$ respectively. Given $i,l,k$, $G(i,l,k)$ is the sequence alignment score between the segment between core $i$ and core $i+1$ and the segment from sequence position $l$ to $k$. The constraint set is as follows:

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

$$x_{i,l} + x_{i+1,k} \le 1, \forall k \in D[i+1] - R[i, i+1, l]; \quad (2)$$

$$y_{(i,l)(j,k)} \le x_{i,l}, k \in R[i,j,l], (c_i, c_j) \in E(CMG); \quad (3)$$

$$y_{(i,l)(j,k)} \le x_{j,k}, l \in R[j,i,k], (c_i, c_j) \in E(CMG); \quad (4)$$

$$y_{(i,l)(j,k)} \ge x_{i,l} + x_{j,k} - 1, (c_i, c_j) \in E(CMG); \quad (5)$$

$$x_{i,l},\ y_{(i,l)(j,k)} \in \{0,1\}. \quad (6)$$

Constraint 1 says that one core can only be aligned to one sequence position. Constraint 2 forbids the conflict between the alignments of two adjacent cores. As such, it guarantees no conflict between the alignments of any two cores, i.e., the integral solution corresponds to a valid alignment. Constraints 3, 4 and 5 enforce that if two interacted cores $i$ and $j$ are aligned to two positions $l$ and $k$, respectively (i.e., $x_{i,l} = 1$ and $x_{j,k} = 1$), then $y_{(i,l)(j,k)}$ should be set to 1, i.e., the interaction scores between two sequence subsegments starting from $l$ and $k$ should be counted in the energy function.

In fact, the above formulation is not the most efficient, but it is the easiest to understand. In the more comprehensive study in [3, 4], we present three different kinds of linear (integer) program formulations to formulate the target sequence-template alignment problem and show how they are derived from one another.

## LP Relaxation and Branch-and-Bound

Although we have formulated the protein threading problem as a mathematically satisfying IP problem, the IP problem is still NP-hard, hence not likely to be solvable in polynomial time. We now take one more step to relax the integral constraint (6) to the following (7).

$$x_{i,j},\ y_{(i,l)(j,k)} \geq 0, \quad j \in D[i],\ i = 1, 2, \ldots, M. \qquad (7)$$

Constraints 1 and 7 now restrict $x$ and $y$ to be in the range $[0, 1]$. The resulting LP problem can now be efficiently solved. RAPTOR uses IBM OSL (Optimization and Solution Library) package.

If the LP solution turns out to be integral, then it is precisely the solution for our original IP system. Surprisingly, in our tests and CAFASP3 runs we had integral solutions 99% of the time. However, if the LP solutions for some $x$ and $y$ variables are fractional between 0 and 1, we select one non-integral variable according to certain criteria, and generate two subproblems by setting this variable to 0 and 1 respectively. These two subproblems are solved recursively, repeating the above process for each subproblem. In practice, only 1% of the instances in our tests required such recursion and the solutions become integral quickly in no more than a few branches.

## Implementation Details

We calculated the averaged energy over a set of homologous sequences rather than a single target sequence, as demonstrated in PROSPECT-II [14]. Given a target sequence of length $n$, we use PSI-BLAST to generate a $n \times 20$ position-specific frequency matrix $p_{n\times 20}$ for it. If a sequence position $j$ is aligned to a template position $i$, then the mutation score $Mutation(i, j)$ and fitness score $Fitness(i, j)$ are defined as $\sum_a p_{j,a} M(t_i, a)$ and $\sum_a p_{j,a} F(env_i, a)$ respectively. Meanwhile, $M(t_i, a)$ represents the mutation potential between two residues, and $F(env, a)$ denotes the fitness potential of residue $a$ being in environment $env$. The local environment is defined by combining three types of secondary structures and three solvent status (buried, intermediate and accessible). If the two ends (at template position $i_1$ and $i_2$) of a template contact are aligned to target sequence position $j_1$ and $j_2$ respectively, then the pairwise score $P(i_1, i_2, j_1, j_2)$ is calculated as $\sum_a p_{j_1,a} \sum_b p_{j_2,b} P(a, b)$, where $P(a, b)$ denotes the distance-independent pairwise potential between two residues $a$ and $b$. $M$ is the PAM250 matrix and $F, P$ are knowledge-based energy parameters. Please refer to PROSPECT-II [14] for the estimation of $F, P$.

Let $Loop(j)$, $Helix(j)$ and $Sheet(j)$ denote the secondary structure prediction confidences (scale 10) at sequence position $j$. If sequence position $j$ is aligned to template position $i$, then the secondary structure score $SS(i, j)$ is defined as $Loop(j) - Helix(j)$ if the secondary structure at template position $i$ is helix, or $Loop(j) - Sheet(j)$ if sheet. The gap penalty function is assumed to be an affine function with open penalty of 10.6 and elongation penalty of 0.8 per gap [15].

We trained the weight factors for the energy items through maximizing the number of correctly-aligned positions generated by RAPTOR on 95 protein pairs chosen from Holm *et al*'s test set [16], each of which is in the fold-level similarity. An alignment for a residue is regarded as correct if it is within 4 residue shift away from the structure-structure alignment generated by SARF [17]. The five weight factors are: $W_m = 1.0$, $W_s = 0.274$, $W_p = 0.796$, $W_g = 17.203$ and $W_{ss} = 5.684$.

We use SVM (Support Vector Machines) [18] to recognize the best templates for each target. For a given threading pair, some features are extracted to train our SVM model such as the traditional $z$ score, the scores of various energy items, total alignment length, total gap length, the number of template contacts with both ends being aligned, the number of template contacts with only one end being aligned, the template size and the sequence size. A pair is treated as positive if and only if they are in the same fold according to SCOP classification. Tested on Lindahl *et al*'s benchmark [19], 5% more targets can be recognized by our SVM model than by the traditional $z$ score. For a given target, all templates are ranked by SVM output. The reliability score for each template is the standardized SVM output.

# 3. RESULTS

## RAPTOR in CAFASP3 and LiveBench6: Evaluation

Large scale experimental results using RAPTOR ware reported in [4]. Here we focus on reporting RAPTOR's results in CAFASP3 [20] and LiveBench6 [21]. CAFASP3 organizers ranked RAPTOR first among all non-meta servers in terms of both alignment accuracy and fold recognition capability for FR targets. RAPTOR also ranks very top in hard homology modeling target recognition. We will explain later why RAPTOR did poorly with HM targets. The specificity result of RAPTOR in CAFASP3 was underestimated because we were still tuning parameters late into July, 2002 and the scale of the reliability score generated by RAPTOR changed radically during the tuning process.

Table 1 shows RAPTOR's performance in each target category. We can see that RAPTOR failed to predict any NF targets, which is normal for a protein threading approach. RAPTOR also performed poorly in the FR(A) and FR/NF categories. A better energy function is needed for these two categories.

**Table 1: RAPTOR's performance in each target category.**

| | CM/FR | FR(H) | FR(A) | FR/NF | NF |
|---|---|---|---|---|---|
| # correct | 6 | 4 | 2 | 1 | 0 |
| # targets | 7 | 7 | 6 | 5 | 5 |

In the latest LiveBench6 test, RAPTOR's performance was worse than that in CAFASP3 under the same evaluation criteria. It ranked No. 6 on easy targets, No. 5 on hard targets, and No. 7 on specificity. We will discuss the reason in Section 4.

## Sample Predictions

For the first domain of target T0136, 17 of 54 servers generated correct fold recognitions. RAPTOR produced the best alignment among all. Figure 3 presents the super-imposition between the experimental structure (grey color) and the

RAPTOR predicted structure (black color) of T0136_1. MaxSub[22] could superimpose a segment of 118 residues (the sequence size is 144) of the predicted structure to the experimental structure within RMSD 1.9A.

**Figure 3: The superimposition of experimental structure (grey color) and prediction structure (black color) of CAFASP3 target T0136_1. It is taken from CAFASP3 website and generated by RasMol and MaxSub.**

In LiveBench6, RAPTOR generated the best alignment for target fp8544 (1kzqA) although its alignment accuracy is low. Only three of about 30 servers predicted it correctly in the first models. In addition, RAPTOR also generated best results for targets fp8543 (1kvzA), fp9304 (1m44A) and fp9010 (1j6rA).

## Computing Efficiency

A set of approximately 62 targets was released by the CASP5 organizers. Our template database contains 3236 protein templates. We used Flexor at the University of Waterloo, which is a Silicon Graphics Origin 3800 system, with 40 400 MHz MIPS R12000 CPUs and 20 GB of RAM. Except for one target (T0174), which used 45 CPU-hours, the CPU time for the other targets averaged 7 hours, ranging from 1 to 20 hours.

# 4. DISCUSSION

## Glitches

RAPTOR did well on fold family (FR) targets and hard HM targets in CAFASP3. However, it has done poorly with family (HM) targets. This was due to three minor glitches: (1) We were still tuning parameters late into July, 2002. (2) We did not updated our database until late June and have not constantly updated it since then, and (3) A minor error occured in updating our database. We used FSSP list released at http://www.ebi.ac.uk/dali/fssp/TABLE1.html on June 16, 2002 to generate our template database.

One clear example is the HM target T0177 which RAPTOR missed even though PDB-blast could predict it correctly. The best two templates for it were 1konA and 1lfpA. 1konA was not incorporated into our template database (although it was in the FSSP database) due to some error, and 1kon was released on June 19, 2002. In addition, T0177 was treated as three targets in CAFASP3 evaluation so RAPTOR's miss on this target was amplified 3-fold.

The second problem was also manifested in the LiveBench6 test. RAPTOR missed the easy target fp10972 (1khyA) whose best templates were 1k6ka and 1ksfX released in PDB on September 27, 2002.

Fixing such glitches results in better performance not only for HM targets but also for FR targets. But since the HM targets are so easy to predict, RAPTOR's error manifested itself more glaringly here.

## Why has it worked?

Despite the minor glitches, the algorithmic advantage of RAPTOR still shone in the FR target recognition category. This is partially because RAPTOR's IP/LP formulation has allowed us to compute optimal solutions for the energy function, which no other program is able to do. Many competing programs simply ignored pairwise contacts in order to achieve computational efficiency. Some other programs, like PROSPECT-I, use a divide-and-conquer method to deal with this problem. Yet, in order to achieve higher speed, PROSPECT-II, which attended CAFASP3, did not use pairwise contact constraints when doing sequence-template alignment. In addition, a good energy function and the use of SVM are also two indispensable components for the success of RAPTOR.

## Strength and Limitations

Experimental results show that RAPTOR performs well in terms of both alignment accuracy and fold recognition for FR targets. The IP/LP formulation has allowed us to fully formulate the messy protein threading problem within an elegant mathematical framework. It has turned out that this elegant mathematical framework also has elegant solutions most of the time. While the problem is still NP-hard, in theory, experience with RAPTOR has shown that most practical instances of this problem can be solved optimally and efficiently. Because we have utilized powerful LP package, RAPTOR runs faster and uses less memory than algorithms that treat the pairwise potentials strictly, when the templates have complex interaction topology and target sequences are long. The divide-and-conquer algorithm employed by PROSPECT-I could deal with the simple interaction topology templates well, but it slows down significantly (exponentially) with more complex interaction topology.

The obvious limitation for the threading methods is that they cannot deal with proteins with novel folds.

## Extensions and Potential Improvements

RAPTOR's LP formulation allows not only pairwise contact information, but also allows users to add extra distance constraints between amino acids. This is useful when the users have prior knowledge of the protein from experience or preliminary NMR experiments.

Another natural extension is to allow gaps in the core regions. Gapped pairwise alignments have natural LP formulation. Allowing core region gaps might help improve alignment quality, which is a weak point for RAPTOR currently.

Due to parameter errors, the optimal energy function value does not necessarily mean the best alignment between target sequences and templates. Most often the best alignment corresponds to one of the lowest $N$ energy function values ($N$ is a small constant). A method to improve alignment accuracy is to generate $N$ alignments corresponding to the $N$ energy values and use some structure checking tool such as verify3D to select the best alignment. It is extremely easy for LP software to generate the best $N$ solutions at little extra cost.

When a short sequence could only be aligned to one domain of a multi-domain template, RAPTOR often missed the correct prediction due to the distortion of the SVM model selecting the correct templates. An example is the LiveBench6 target fp9293 (1l8dA). One of the correct templates is 1i1ip. RAPTOR generated a very good alignment score between 1i1ip and fp9293 but failed to rank 1i1ip within the top 10 due to the huge mismatch of their lengths. We need to improve our SVM model for this case.

RAPTOR also has problems in predicting a long target sequence. Sometimes RAPTOR could rank several correct short templates within the first 10 models, but it could not assemble these models to generate a better and longer model for the whole sequence. An example is target T0136. RAPTOR generated the best model for its first domain among all servers but ranked this best model ninth due to length mismatch. The first model submitted by RAPTOR was also a correct prediction for the second domain of T0136. The next step for RAPTOR is assembling these two models together to give a better prediction.

**Acknowledgements: The authors would like to thank D. Kim, G. Lin, D. Xu, and Y. Xu for collaborating with us on the RAPTOR project. The authors would also like to thank Professor Prabhakar Ragde for proofreading this paper.**

## 5. REFERENCES

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

[2] M. Garey and D. Johnson. *A guide to NP-completeness*. Freeman, 1979.

[3] J. Xu, M. Li, G. Lin, D. Kim, and Y. Xu. Protein threading by linear programming. pages 264–275, Hawaii, USA, 2003. Biocomputing: Proceedings of the 2003 Pacific Symposium.

[4] J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1, 2003.

[5] L.A. Wolsey. *Integer Programming*. JOHN WILEY & SON Inc., 1998.

[6] D.T. Jones, W.R. Taylor, and J.M. Thornton. A new approach to protein fold recognition. *Nature*, 358:86–98, 1992.

[7] A. Godzik, A. Kolinski, and J. Skolnick. A topology fingerprint approach to inverse protein folding problem. *J. Mol. Biol.*, 227:227–238, 1992.

[8] S. Bryant. Evaluation of threading specitity and accuracy. *Proteins: Struct. Funct. Genet.*, 26:172–185, 1996.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] D. Kim, D. Xu, J. Guo, K. Ellrott, and Y. Xu. PROSPECT II: Protein structure prediction method for genome-scale applications. 2002. submitted.

[15] G.H. Gonnet, M.A. Cohen, and S.A. Benner. Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445, 1992.

[16] L. Holm and C. Sander. Decision support system for the evolutionary classification of protein structures. *ISMB*, 5:140–146, 1997.

[17] N.N. Alexandrov. SARFing the PDB. *Protein Engineering*, 9:727–732, 1996.

[18] V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[19] E. Lindahl and A. Elofsson. Identification of related proteins on family, superfamily and fold level. *J. Mol. Biol.*, 295:613–625, 2000.

[20] D. Fischer. http://www.cs.bgu.ac.il/~dfischer/CAFASP3/, December 2002.

[21] L. Rychlewski. http://www.bioinfo.pl/livebench/6, 2002.

[22] 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.