Computing images of protein cores for protein threading

by

Hongchao Li

(Under the direction of Liming Cai)

Abstract

Protein threading is one of the most important computational approaches for protein tertiary structure prediction. The main computation portion in threading is to align a query protein sequence (with unknown structure) to each of existing protein structure templates. An often used technique for the sequence-structure alignment is to preprocess the query sequence with the template structure before combinatorial, linear-programming, or graph-theoretic algorithms are applied to the alignment. The preprocessing finds the top candidates in the query protein sequence for each of the cores (i.e., rigid secondary structure elements) in the structure template. Whether the top candidates include the true correspondence in the query sequence to each core in the template determines the overall score of the alignment and consequently whether the threading algorithm can correctly identify the right structure for the query sequence.

This work focuses on the threading preprocessing. In this work, three methods, direct scan, global alignment based scan and anchor-based hybrid scan, are presented for threading preprocessing. From the experiment evaluation with Dali dataset, it is found that the methods of global alignment based scan and anchor-based hybrid scan outperform the direct scan method when the template and the query are close in length; the direct scan achieves better performance when the lengths of the template and the query are very different. Based

on the overall experiment evaluation, it is believed that accurate energy functions are critical to effective threading preprocessing.

INDEX WORDS: Protein tertiary structure, Protein threading, Sequence-structure alignment, Core, α-helix, β-strand, Image

Computing images of protein cores for protein threading

by

Hongchao Li

B.E., Nankai University, China, 1999

M.E., Nankai University, China, 2002

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2007

Computing images of protein cores for protein threading

by

Hongchao Li

Approved:

Major Professor: Liming Cai

Committee: John A. Miller
Russell Malmberg

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2007

Acknowledgments

I would like to gratefully thank my advisor, Dr. Liming Cai for his guidance on my research, encouragement and help both in my study and life. His spirit and enthusiasm always inspire me. His friendship makes my master study such an enjoyable journey.

I would like to thank my committee members, Dr. Russell L. Malmberg and Dr. John A. Miller for their time and valuable advices.

I would like to thank Dr. Yinglei Song for his generous help and good suggestions on my thesis. I like to thanks Dr. Juntao Guo and Mr. Kyle Ellrott for their helps on biological knowledge and data.

I would like to thank Dong Zhang, Zhibin Huang, Yong Wu and other group members for having such a happy time together with them.

I would give my gratitude to my dearest parents, to my wife, Rui and to my little daughter, Anna. It is their love and endless supports that make this all possible.

Finally, I would like to thank the support of the University of Georgia.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent years, many efforts such as the Human Genome Project have resulted in a rapid increase of the number of new genes and completely sequenced proteins. The large number of newly sequenced proteins makes it a rather pressing task to understand the functions of these proteins. The tertiary (3D) structures of proteins carry the essential information for defining the proteins' biological functions. While experimentally determining the protein tertiary structure, finding fast and reliable computational methods to predict the tertiary structures becomes the key to relieve the pressure.

1.1 Existing Methods for Protein Tertiary Structure Prediction

Generally, existing computational methods for protein tertiary structure prediction can be classified into two categories: *ab initio* modeling and comparative protein modeling.

An *ab initio* modeling method attempts to derive 3D protein structures directly from the protein sequences, without referring to any previously solved protein structures, by using physi-chemiccal principles [5]. The *ab initio* modeling is based on the thermodynamic hypothesis of protein folding: the native structure of a protein sequence should correspond to the state with the minimum global free energy [20] [2]. In the *ab initio* modeling, a protein tertiary structure is represented with a combination of the values of its 3D spatial position, called a conformation. An objective function defines the physical energy for each of the allowed conformations. Then, the *ab initio* modeling tries to find the global minimum within the space of the allowed conformations [15] [12]. However, due to the large number of

physi-chemical parameters needed to be considered, correctly modeling a protein structure with such parameters has far been close to satisfactory.

Comparative protein modeling methods instead use previously solved structures as references to predict the structures of new proteins. It is believed that the total number of different structural folds in nature is relative small [11] [19], possibly in the range of a few thousand, although the number of protein sequences is vast. With the tertiary structures of a large number of proteins having been identified, the tertiary structure of a newly sequenced protein may have existed among these previously identified structures. Based on this thought, the comparative modeling methods are to find the solved protein or protein groups who have the same structure as the new protein by comparing the new protein to the solved proteins or the profiles of protein groups [20].

In comparative protein modeling methods, the proteins with solved tertiary structures are usually called templates and the newly sequenced protein whose structure is to be solved is called the query sequence, or simply query. According to the principles and techniques used in query-template comparisons [20], the comparative modeling methods could be further classified into two groups: homology modeling and threading modeling.

Homology modeling assumes that two homologous proteins have very similar structures. Thus, the kernel part of homology modeling methods is to find the homologous protein or protein family among the template set. Multiple sequence alignments are usually used to measure the homology the distance between the amino acid sequences of the query and templates [26] [13] [21]. With the increasing of the experimentally determined protein structures, protein homology modeling has made continuing progress.

However, a protein's fold is more evolutionarily conserved than its amino acid sequence. Some proteins may still have same 3D structures although they are distant homologous sequences (i.e. their amino acid sequences are very different). Then, although sometimes the homologous templates can not be found for the query protein, some templates in the database have already existed with the same fold as the query. Protein threading modeling

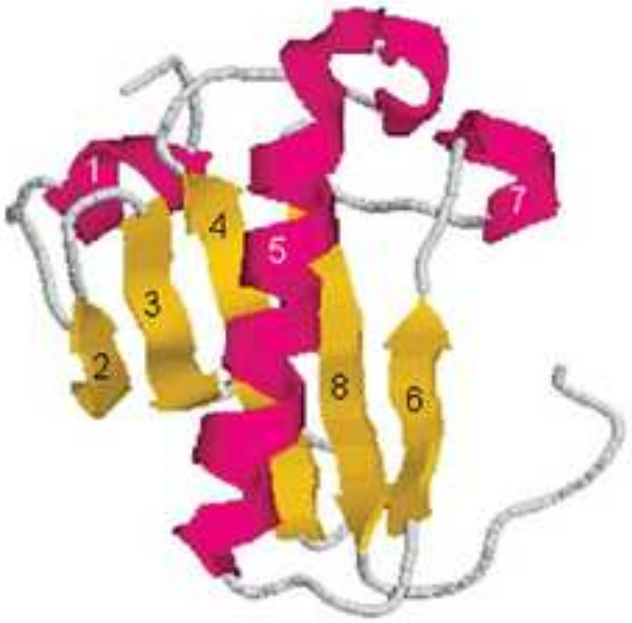

Figure 1.1: Folded ChainB of Protein Kinase C (PDB-ID 1AV)[28]

addresses this problem. Instead of evaluating homology distance between the query and the templates, protein threading assesses the compatibility of the query and the structures of the templates. The procedure of threading can be described as follows [26]: given a query protein sequence s of unknown structure, threading searches a structure template library T to find the template structure t, $t \in T$ most compatible with the sequence s and take this template structure as the predicted structure for the sequence s. In next two sections, the challenges of threading modeling will be introduced and the tree-decomposition-based threading will be presented to exemplify the computational formulation of protein threading.

1.2 Challenges in Threading Modeling

A protein tertiary structure consists of cores and loops. A structure core, typically an α-helix or a strand of a β-sheet, contains a stretch of amino acids where the tertiary structure is highly conserved during the evolution. In contrast, a loop consists of the residues in between two consecutive cores and its tertiary structure can be variable during the evolution. Moreover, in the spatial conformation of the protein structure, some pairs of amino acids are close to each other although they are not neighboring residues along the sequence. When the distance between a pair of non-neighboring residues is smaller than a specified cutoff, we called there is a two-body interaction between the pair of residues. Figure 1.1 depicts the tertiary structure of Protein Kinase C (PDB-ID 1AV) which has eight cores and some two-body interactions along the sequence. Meanwhile, the cores 1, 5 and 7 are α-helices and cores 2, 3, 4, 6 and 8 are β-strands.

Protein threading evaluates the compatibility of a query protein with each structure template and finds out the structure template most compatible to the query protein based on the evaluation. Meanwhile, the compatibility evaluation between the protein sequence and each structure template is the core computational part. Generally, a knowledge-based energy value, calculated by an alignment between the query sequence and the structure template, is used to measure the sequence-structure compatibility. The general form of energy function could be written as follows [28]:

$$w_f E_f + w_p E_p + w_g E_g \tag{1.1}$$

where E_f measures the overall fitness of putting individual residue types into specific structural environments, E_p measures the total interaction energy of all two-body interacting pairs, and E_g represents the total penalty for the gaps in a sequence-structure alignment. The scaling factors w_f, w_p and w_g are typically determined empirically through optimizing the performance of a threading program on a representative set of protein pairs. To measure the fitness in several aspects, E_f may be split into several sub-energy-terms. More

details about the energy functions and sequence-structure alignment will be discussed in next chapter.

With the optimized w_f, w_p and w_g values, the threading procedure finds an alignment of the query protein and each structure template with the minimum free energy value. By comparing the minimum energy values of the alignments between the query and all the structure templates, the best compatible structure is picked out for the query protein.

In threading, if two-body interaction energy E_p is not considered, the sequence-structure alignment can be done like sequence-sequence alignment as it finds an alignment between a sequence of amino acids and a sequence of structural positions. A global dynamic programming procedure can efficiently find the optimal alignment solution and energy value. However, when two-body interactions are considered, the sequence-structure compatibility evaluation becomes NP-hard [9].

To address the algorithmic challenges, a number of threading programs [29] [30] [27] [8] [24] [25] have employed various heuristic strategies and/or biological premises to solve the optimal sequence-structure alignment problem. For example, some methods [29] [30] assumed that the contacting residues are the same in the template and in the query sequence when calculating the energy of a two-body interaction. Some methods [27] only consider those two-body interactions between spatially close pairs.

One of them is the tree-decomposition-based threading method [17][28], which reduces the sequence-structure alignment problem into a subgraph isomorphism problem and solves the subgraph isomorphism problem with the tree-decomposition method in polynomial time [3] [10] [22] [23]. It will be introduced in next section.

1.3 Tree-Decomposition-based Threading Method

The tree-decomposition based method models two-body interactions in a structure template with a conformational graph. In particular, in such a model, cores are represented by vertices,

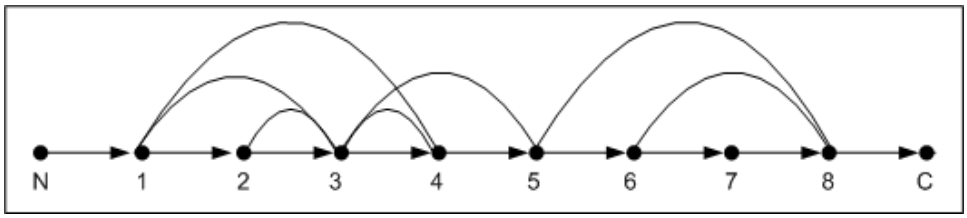

Figure 1.2: The conformation graph of folded ChainB of Protein Kinase C (PDB-ID 1AV)[28]

and cores next to each other in the backbone are connected with directed edges from the N-terminal to the C-terminal. In addition, undirected edges connect two vertices if there exist two-body interactions between some amino acids contained in the two corresponding cores. Figure 1.2 shows the conformational graph of the protein tertiary structure given in Figure 1.1. The conformational graph is the *prototype* of the subgraph in the deduced subgraph isomorphism problem and the *full graph* in the deduced problem is also constructed based on this conformational graph.

The construction of the full graph consists of two steps: gathering vertices and connecting edges. In the conformational graph, the vertices represent the cores of the structure template. Accordingly, the vertices of the full graph are the representations of the k top candidates of each template core on the query sequence, where k is a parameter determined by a statistical cut-off which can be naturally small. The value of k is the key factor deciding the running time of the following steps of protein threading and a larger value could significantly increase the running time. Usually, k is less than 10. The top candidates are called *images* of the template core and the full graph is called *image graph* as well. To locate the top images of each template core, the core is aligned with all subsequences of the query. The k subsequences with the best alignment scores are kept as the top-k images of the core. In general, the step of gathering top images for template cores is called the *preprocessing* of threading procedure.

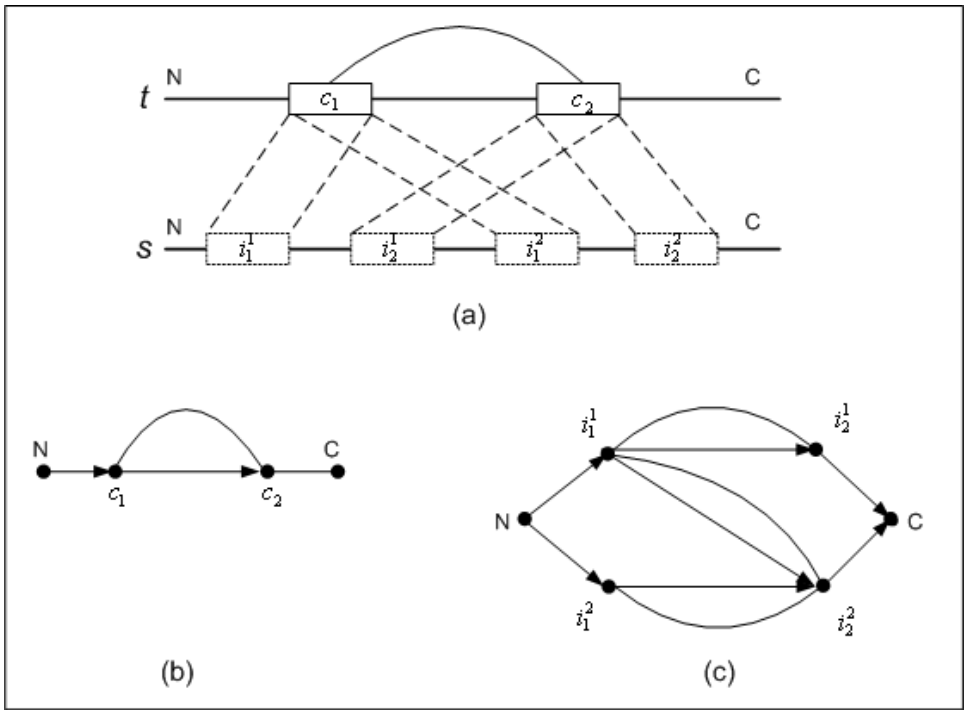

Figure 1.3: The construction of image graph for the alignment between query s and template t. (a) The preprocessing result where core c_1 has top images i_1^1 and i_1^2, and c_1 has top images i_2^1 and i_2^2. (b) The conformation graph of template t. (c) The constructed image graph.

The second step is to connect the edges between images according to the conformational graph. Specifically, two images i_u and i_v are connected with an undirected edge if their corresponding cores u and v are connected with an undirected edge in the conformational graph and images i_u and i_v are in the same order along the query sequence as cores u and v along the template. Image i_u is connected to image i_v with a directed edge if i_u is to the left of i_v and there exists a directed edge from u to v in the conformational graph. Figure 1.3 demonstrates an example of the construction of an image graph based on the threading preprocessing result and the conformational graph of the template.

In the image graph, weights are also assigned to vertices and edges to convey the compatibility information between the query sequence and the template as follows. (1) $w_c(u)$, the weight associated with a vertex is its alignment score between the image and the core. Details of the weight calculation will be demonstrated in next chapter. (2) Since a directed edge $< u, v >$ represents a loop between two images, its weight $w_l(u, v)$ is the score of aligning the loop sequence segment to the corresponding loop profile in the structure template, which includes the contributions from energy terms E_f and E_{gap}. (3) The weight $w_i(u, v)$ of an undirected edge (u, v) is computed by summing up the energies of all two-body interactions between the two images.

Then the sequence-structure alignment for protein threading corresponds to an embedding of the conformational graph into the image graph. The alignment score is the sum of the values of all vertices and edges in a selected subgraph in the image graph isomorphic to the conformational graph. In particular, the problem of optimally aligning a sequence to a structure profile can be formulated as a minimum valued subgraph isomorphism problem. The detailed steps to solve subgraph isomorphism problem with tree-decomposition method can be found in Song et al's work [17] [18].

1.4 The Work of this Thesis

Similar to the tree-decomposition based threading, some other threading methods such as RAPTOR [24] and PROSPECT [27] try to locate the top images for each template core in the beginning stage of sequence-structure alignment. The preprocessing is critical to the overall sequence-structure compatibility evaluation. Whether the top images in the query contain the true image of the template core determines the correctness of the sequence-structure compatibility, thus the correctness of the final structure prediction. So, it is safe to say that the preprocessing result is very important to protein threading.

So far, the performance of threading has not been completely satisfactory. Yet no much work has been done to carefully investigate the preprocessing part. This thesis focuses on threading preprocessing: 1) identify the components and the steps of threading preprocessing; 2) present three preprocessing methods and evaluate these methods with experiments; 3) recognize the factors constraining the preprocessing performance.

The rest of the thesis is organized as follows. Chapter 2 introduces the components of threading preprocessing; three preprocessing methods are presented in Chapter 3 and the experiment evaluations of the three methods are demonstrated in Chapter 4. Chapter 5 concludes the work of this thesis and addresses the future work.

Chapter 2

Threading Preprocessing Components

A threading method typically consists of four components [16]: (1) a query sequence s and a library T of representative 3D protein structures for use as templates; (2) an energy function for measuring the compatibility between the query s and a structure template t, where $t \in T$; (3) an algorithm to search for an alignment with the lowest energy for a given s-t pair; and (4) a criterion for estimating the confidence level of the predicted structure. The threading preprocessing is one early step in the third component, which aims at finding in the query sequence s the top candidates for each core in the template t. Beside the first two components of a threading method, the threading preprocessing requires a compatibility evaluation function of an image to the corresponding core. In the reminder of this chapter, these three components will be introduced.

2.1 query sequence and structure template

The query sequence is the input data of threading procedure. It basically provides the primary structure of the protein, i.e. the amino acid sequence. In addition to this, the predicted secondary structure information is also required by some threading methods. The secondary structure information mainly refers to the probabilities of each residue being in a α-helix, a β-strand and a loop. There are tools such as PHD [14] that can predict the secondary structure of a protein sequence with a fair accuracy.

A structure template used in threading is usually a structure profile of multiple alignments of some proteins that have the same tertiary structure. Such a profile conveys the following information:

- the template length,
- the frequencies of the 20 amino acid types at each position
- the secondary structure at each position
- the solvent accessibility at each position
- the position pairs of all the two-body interactions

2.2 Energy Function

As we mentioned in Chapter 1, the energy function generally includes three general terms (as Equation 1.1): residue-position fitness energy E_f, two-body interaction energy E_p and gap penalty E_g. This section will describe each term with the formula to calculate it.

Gap Penalty in an alignment is evaluated with the affine gap penalty function specified with two parameters representing the penalties for opening a gap and for extending a gap. Equation 2.1 defines the penalty for a particular continuous gap.

$$e_g = \gamma + \delta(|g| - 1)) \tag{2.1}$$

where $|g|$ is the length of the continuous gap g, γ is the penalty for opening a gap and δ is the penalty for extending a gap.

The overall gap penalty is the sum of penalties of all continuous gaps in the in the sequence-structure alignment and the formula is defined as equation 2.2.

$$E_g = \sum_{g \in G} e_g \tag{2.2}$$

where G represents the set of all the continuous gaps.

The interaction energies of all possible two-body residue pairs are defined by a knowledge-based energy matrix. Suppose that the two residues of a two-body interaction i are l_i and r_i. The energy of this two-body interaction can be looked up in the knowledge-based energy matrix. That is,

$$e_p(i) = M_I(l_i, r_i) \tag{2.3}$$

where M_I is a 20×20 matrix defining the interaction energy between all possible residue pairs. The overall energy of two-body interactions in the sequence-structure alignment is the sum of energy of all two-body interactions as defined in equation 2.4.

$$E_p = \sum_{i \in I} M_I(A(l_i), A(r_i)) \tag{2.4}$$

where I is the set of all two-body interactions on the structure template, l_i and r_i are the two positions of the two-body interaction i, $A(l_i)$ and $A(r_i)$ are the query sequence residues aligned to the template positions l_i and r_i.

In order to take multiple factors into account of the residue-position fitness evaluation, energy term E_f is usually split into several sub-terms. The most common sub-terms are three: singleton energy E_s, mutation energy E_m and secondary structure energy E_{ss} [8].

The singleton energy e_s evaluates the preference of the placement of an amino acid in a residue position within certain secondary structure and solvent accessibility. The secondary structure for a given location can be an α-helix, a β-strand, or a loop while the solvent accessibility can belong to one of the three categories: buried, exposed, and intermediate based on the comparison of its solvent accession area with two cut-off thresholds [27]. For example, if a residue r in the query sequence is aligned to the position i of the structure template. Then, the singleton energy of this residue-position pair can be calculated as in equation 2.5.

$$e_s(r, i) = M_S(r, ss_i, sa_i) \tag{2.5}$$

where ss_i and sa_i are the secondary structure type and the solvent accessibility category at position i of the structure template. M_S is a $20 \times 3 \times 3$ matrix defining the knowledge-based energy values of all the combinations of residue type, secondary structure type and solvent accessibility level. The overall singleton energy of the sequence-structure alignment is the sum of the singleton energies of all the aligned residue-position pairs (as equation 2.6).

$$E_s = \sum_r e_s(r, i_r) \tag{2.6}$$

where i_r is the template position aligned to residue r.

The mutation energy E_m measures the preference of the mutations between query sequence amino acids and the residue types at the aligned template positions. The mutation energy of the residue-position pair (r, i) can be calculated as in equation 2.7.

$$e_m(r,i) = \sum_{t=1}^{20} r \cdot p_i(t) \cdot M_M(r,t) \tag{2.7}$$

where t is an amino acid type, $p_i(t)$ is the frequency of amino acid type t at the template position i and M_M is a 20×20 matrix defining the knowledge-based energy value for mutations between any residue-residue pair.

The overall mutation energy of the sequence-structure alignment is the sum of the mutation energies of all the aligned residue-position pairs (as equation 2.8).

$$E_m = \sum_r e_m(r, i_r) \tag{2.8}$$

For secondary structure matching energy, a function is defined to a residue-position alignment pair (r, i) as equation 2.9.

$$e_{ss}(r,i) = \tau - rint(p_r(ss_i) \cdot \rho) \tag{2.9}$$

where $p_r(ss_i)$ is the probability of r being in secondary structure type ss_i, and rint(x) is to get the closest integer of x. The value of $p_r(ss_i)$ is usually provided by the query sequence presentation and can also be calculated by protein secondary structure prediction tools such as program PHD [14]. τ and ρ are the two empirical constants. The overall secondary structure matching energy of the sequence-structure alignment is the sum of the secondary structure energies of all the aligned residue-position pairs (as equation 2.10).

$$E_{ss} = \sum_{r \in S} e_{ss}(r, i_r) \tag{2.10}$$

The overall energy function used in the sequence-structure alignment is defined as

$$E_{total} = w_s e_s + w_m e_m + w_{ss} e_{ss} + w_g e_g + w_p e_p \tag{2.11}$$

where w_s, w_m, w_{ss}, w_g and w_p are the empirical relative weights for the respective energy terms.

2.3 Energy Calculation for an Image of a Template Core

In threading, energy values are used to measure the compatibility between query sequences and structure templates. Accordingly, the selection of the top images for a template core is also based on the energy values of the images aligned to the core. As we mentioned in Chapter 1, the gaps are only allowed at two ends of an image. In this case, the core positions aligned to gaps are regarded as being deleted in evolution. In the energy calculation of the image, the contiguous gap at one end of the image is considered as a part of the loop connecting this end of the image, instead of a part of the image. Thus, the gap penalty term will not be involved in the energy calculation of an image. To make the explanation easier, we will discuss the image energy calculation in two scenarios: no gaps allowed at two ends of an image and allowing gaps at two ends of an image.

Suppose the template core c has $|c|$ positions and the selected image is the subsequence q of $|c|$ residues starting at position a on the query. When no gaps are not allowed at the two ends of the image, the energy of the image i is the overall energy of the direct alignment between core c and the subsequence q as defined in equation 2.12.

$$E_i = \sum_{j=0}^{|c|-1} (w_s e_s(q_j, j) + w_m e_m(q_j, j) + w_{ss} e_{ss}(ss_j, j)) + w_p \sum_{i \in I} e_p(i) \tag{2.12}$$

where j is the j-th position of the core (starting 0), q_j is the j-th residue along the subsequence q, and I is the set of all two-body-interactions within the core.

If gaps are allowed at the two ends of an image, we probably need to count only a part of the subsequence q as the image of the core c if this part has smaller energy than the whole subsequence. In this case, we first calculate the overall energy of the whole subsequence q to the core c as same as the previous scenario. Then, we check how many continuous residues can be chopped off at either end of the image without hurting the positions involving two-body interactions. By chopping some continuous residues at each end, we find the remaining part with the minimum energy. The remaining part is indeed the best image of the core within subsequence q. Figure 2.1 demonstrates two examples of image evaluation, of which

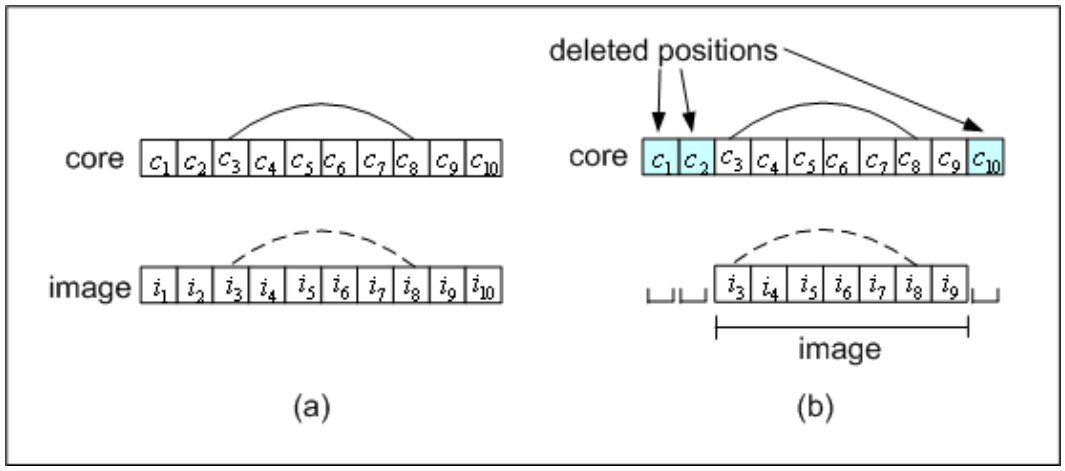

Figure 2.1: Image evaluation examples (a) No gaps allowed in the image (b) Some gaps allowed at the two ends of the image.

one is no gap allowed in the image and the other is gaps allowed at the two ends of the image.

Figure 2.2 is an algorithm to find the best image. In the algorithm, ss is the secondary structure array of the subsequence q, δ is the maximum number of gaps allowed at either end of an image and img is the best image with the minimum energy score. The algorithm first divides the subsequence q into 3 parts: left gap-allowable part, central continuous part and right gap-allowable part. Then, it finds the optimal gap number at each gap-allowable part. Eventually, it combines the leftovers of the two ending parts with the central part together to construct the best image in q again. The most time-consuming part in **Eval-Image** is the calculation of $E[0, ..., |c| - 1]$ and overall running time of **Eval-Image** is in $\Theta(|c|)$.

Eval-Image(c, q, ss)

▽ Calculate the energy at each position excluding two-body interaction energy

1 **for** $j : 0$ **to** $|c| - 1$

2 $\quad E[j] \leftarrow (w_s e_s(q[j], j) + w_m e_m(q[j], j) + w_{ss} e_{ss}(ss[j], j))$

▽ Calculate the overall two-body interaction energy

3 $P \leftarrow w_p \sum_{i \in I} e_p(i)$

▽ Find the boundary between left part and central part

4 $l \leftarrow$ the left-most position involving a two-body interaction on the core

5 $l \leftarrow min\{l, (\delta - 1)\}$

▽ Find the boundary between right part and central part

6 $r \leftarrow$ the right-most position involving a two-body interaction on the core

7 $r \leftarrow max\{r, (|c| - \delta + 1) - 1)\}$

▽ Calculate the overall energy of the central part

8 $E_{central} \leftarrow 0$

9 **for** $j : (l + 1)$ **to** $(r - 1)$

10 $\quad E_{central} \leftarrow E_{central} + E[j]$

▽ Find out the optimal gap number in the left part

11 $E_l[l + 1] \leftarrow 0$

12 $loc_l \leftarrow (l + 1)$

13 $min_l \leftarrow 0$

14 **for** $j = l$ **downto** 0

15 $\quad E_l[j] \leftarrow E_l[j + 1] + E[j]$

16 $\quad$ **if** $(min_l > E_l[j])$

17 $\quad min_l \leftarrow E_l[j]$

18 $\quad loc_l \leftarrow j$

▽ Find out the optimal gap number in the right part

19 $E_r[r - 1] \leftarrow 0$

20 $loc_r \leftarrow (r - 1)$

21 $min_r \leftarrow 0$

22 **for** $j \leftarrow r$ **to** $|c| - 1$

23 $\quad E_r[j] \leftarrow E_r[j - 1] + E[j]$

24 $\quad$ **if** $(min_r > E_r[j])$

25 $\quad min_r \leftarrow E_r[j]$

26 $\quad loc_r \leftarrow j$

▽ Combine the three parts together into the best image

27 $E_{tot} \leftarrow min_l + E_{central} + min_r + P$

28 $img \leftarrow q[loc_l, ..., loc_r]$

29 **return** img

Figure 2.2: The algorithm to find the best image within a subsequence of query

Chapter 3

Protein Threading Preprocessing Methods

The ideal goal of threading preprocessing is to directly locate the true image in a query corresponding to each core in a given structure template. However, many constraints such as lacking the accurate energy functions make this goal unrealistic. Threading preprocessing usually locates multiple top images for each template core to include the true one. The intuitive way to do so is to scan each core along the query sequence and to locate the subsequences of the query which have the best energy values against the core. However, the accurate energy function may not be available to allow the obtained top images to always contain the corresponding true image of each template core. On the other hand, some additional information such as the relative locations of the template cores could be integrated to help the top image scanning. This chapter, starting with the intuitive method, presents three methods for threading preprocessing.

3.1 Direct Scan Method

Direct scan is a simple method for top image search. This is to scan the whole query sequence with each template core. The template core slides along the query sequence position by position and the images with top-k best energy values are kept. The algorithm is presented as in Figure 3.1 where **Handle-New-Image** handles all images and keeps the top-k images and **Direct-Scan-Method** is the main body of the method. **Direct-Scan-Method** takes the inputs of a query sequence s, an array ssa containing the predicted secondary structure of each residue of s and a structure template t.

Handle-New-Image(c, img)
1 topImageSet$[c] \leftarrow img \cup$ topImageSet$[c]$
2 **if** sizeof(topImageSet$[c]$) $> k$
3 remove the image with the highest energy from topImageSet$[c]$

Direct-Scan-Method(s, ssa, t)
1 **for** each core $c \in t$
2 **for** $i : 0$ **to** $|s| - |c| - 1$
3 $q \leftarrow s[i, ..., (i + |c| - 1)]$
4 $ss \leftarrow ssa[i, ..., (i + |c| - 1)]$
5 $img \leftarrow$ Eval-Image(c, q, ss)
6 Handle-New-Image(c, img)

Figure 3.1: Direct scan method

If **Handle-New-Image** is implemented with insertion sort, its time complexity is just in $\Theta(k)$ every run which actually is in $\Theta(1)$ since k is usually chosen smaller than 10. Suppose the number of cores is C, the query sequence has n residues, and the maximum length of a core is l, which is usually less than 30. Then, the overall running time of **Direct-Scan-Method** is in $O(lCn)$ or $O(Cn)$ considering l as a constant. Moreover, the core number of a template is far less than 100. Thus, if we regard C as a constant as well, **Direct-Scan-Method** is in the linear time complexity to the length of the query sequence. Its small time complexity is the advantage over other two methods introduced later.

However, this method completely relies on the accuracy of the energy functions. As we know, the energy terms used in threading are knowledge-based and they are just the statistical values. These parameters might not accurately reflect the sequence-structure compatibility. So, the maturity of the energy functions constrains the performance of direct scan method.

3.2 Global-Alignment-based Scan Method

Beside the energy functions, the relative spatial locations of the cores in a template can be helpful for the top images scan. If a method could consider all the cores in one scan round, it might help accurately find top-k images for all the cores. In addition, a lot of experiments indicate that the energy term of two-body interactions actually only contributes a very small part to the overall energy value. Based on these two points, global alignment based scan method has been designed, in which a global alignment without considering two-body interactions is first conducted between the query and the template. The global alignment result provides a reference location on the query for each template core and the top images of the template core are then scanned within the neighborhood of this reference location. The steps of the global-alignment-based (GA-based) scan method are as follows.

1. Align the query sequence to the template structure with a dynamic programming algorithm in which the energy function includes all the terms other than the two-body interaction terms;

2. Trace back the alignment result and find the aligned location for each core. Around the aligned location of each core, set a scan range with the radius of r;

3. Scan the top images for each core within the scan range set in Step2.

3.2.1 Sequence-Structure Global Alignment

One point distinguishes the sequence-structure alignment from common sequence-sequence alignment: no gaps are allowed in the middle of an image of a core. To simplify the alignment, our algorithm even does not allow any gaps even at two ends of an image. This simplification may not hurt the overall preprocessing since the global alignment only provides a rough reference location for the image scanned for each template core.

In the dynamic programming, three objective functions are used at same time: $Q(i,j), T(i,j)$ and $H(i,j)$ ($i = 0, ..., n$ and $j = 0, ..., m$) where n and m are the lengths of the query s and

the template t. Three $(n+1) \times (m+1)$ matrices Q, T and H are used to store the values of $Q(i,j), T(i,j)$ and $H(i,j)$. Meanwhile, $Q[i,j]$ is the optimal alignment energy score where $s[i-1]$ and $t[j-1]$ are the last query residue and template position that the alignment has used and currently $q[i-1]$ is aligned to a gap; $T[i,j]$ is the optimal alignment energy score where $s[i-1]$ and $t[j-1]$ are the last query residue and template position that the alignment has used and currently $t[j-1]$ is aligned to a gap; $H[i,j]$ is the optimal alignment energy score where $s[i-1]$ and $t[j-1]$ are the last query residue and template position that the alignment has used and currently $s[i-1]$ is aligned to $t[j-1]$. In the alignment, three $(n+1) \times (m+1)$ matrices B_Q, B_T and B_H are used to respectively record the alignment paths of Q, T and H for the final trace back.

I. Initialization

Initialization of Q and B_Q:

$$Q[0,0] = +\infty \tag{3.1}$$

$$B_Q[0,0] =' NIL' \tag{3.2}$$

$$Q[i,0] = \begin{cases} \gamma, & \text{for } i = 1 \\ Q[i-1,0] + \delta, & \text{for } i = 2,3,... \end{cases} \tag{3.3}$$

$$B_Q[i,0] = \begin{cases} 'FROM - H - TABLE', & \text{for } i = 1 \\ 'FROM - Q - TABLE', & \text{for } i = 2,3,... \end{cases} \tag{3.4}$$

and

$$Q[0,i] = +\infty \, (i = 1,2,...) \tag{3.5}$$

$$B_Q[0,i] =' NIL' \tag{3.6}$$

Initialization of T and B_T:

$$T[0,0] = +\infty \tag{3.7}$$

$$B_T[0,0] =' NIL' \tag{3.8}$$

$$T[i,0] = +\infty \, (i = 1, 2, ...) \tag{3.9}$$

$$B_T[i,0] =' NIL' \tag{3.10}$$

and

$$T[0,i] = \begin{cases} \gamma, & \text{for } i = 1 \text{ and t[1] is not in a core} \\ +\infty, & \text{for } i = 1 \text{ and t[1] is in a core} \\ Q[0,i-1] + \delta, & \text{for i = 2, 3, ... and t[i] is not in a core} \\ T[0,i] = +\infty, & \text{for i = 2, 3, ... and t[i] is in a core} \end{cases} \tag{3.11}$$

$$B_T[0,0] = \begin{cases} 'FROM - H - TABLE', & \text{for } i = 1 \text{ and t[1] is not in a core} \\ 'FROM - T - TABLE', & \text{for i = 2, 3, ... and t[i] is not in a core} \\ 'NIL', & \text{otherwise} \end{cases} \tag{3.12}$$

Initialization of H and B_H:

$$H[0,0] = 0 \tag{3.13}$$

$$B_H[0,0] =' NIL' \tag{3.14}$$

$$H[i,0] = +\infty \, (i = 1, 2, ...) \tag{3.15}$$

$$B_H[i,0] =' NIL' \tag{3.16}$$

and

$$H[0,i] = +\infty \, (i = 1, 2, ...) \tag{3.17}$$

$$B_H[0,i] =' NIL' \, (i = 1, 2, ...) \tag{3.18}$$

II. Recursive Construction

$$Q[i,j] = \begin{cases} +\infty, & \text{if t[j]and t[j+1]are} \\ & \text{both in a core} \\ min(Q[i-1,j]+\delta, T[i-1,j]+\gamma, H[i-1,j]+\gamma), & \text{otherwise} \end{cases} \tag{3.19}$$

$$B_Q[i,j] = \begin{cases} 'FROM-Q-TABLE', & \text{if } Q[i,j] = Q[i-1,j]+\delta \\ 'FROM-T-TABLE', & \text{if } Q[i,j] = T[i-1,j]+\gamma \\ 'FROM-H-TABLE', & \text{if } Q[i,j] = H[i-1,j]+\gamma \\ 'NIL', & \text{otherwise} \end{cases} \tag{3.20}$$

$$T[i,j] = \begin{cases} +\infty, & \text{if } t[j] \text{ is a core position} \\ min(Q[i,j-1]+\gamma, T[i,j-1]+\delta, H[i-1,j]+\gamma), & \text{otherwise} \end{cases} \tag{3.21}$$

$$B_T[i,j] = \begin{cases} 'FROM-Q-TABLE', & \text{if T}[i,j] = Q[i,j-1]+\gamma \\ 'FROM-T-TABLE', & \text{if } T[i,j] = T[i,j-1]+\delta \\ 'FROM-H-TABLE', & \text{if } T[i,j] = H[i,j-1]+\gamma \\ 'NIL', & \text{otherwise} \end{cases} \tag{3.22}$$

$$H[i,j] = \begin{cases} H[i-1,j-1]+E(i-1,j-1), & \text{if t[j-1]is} \\ & \text{in a core} \\ min(Q[i-1,j-1], T[i-1,j-1], H[i-1,j-1]) + E(i-1,j-1), & \text{otherwise} \end{cases} \tag{3.23}$$

$$B_H[i,j] = \begin{cases} 'FROM-Q-TABLE', & H[i,j] = Q[i-1,j-1]+E(i-1,j-1) \\ 'FROM-T-TABLE', & \text{if } H[i,j] = T[i-1,j-1]+E(i-1,j-1) \\ 'FROM-H-TABLE', & \text{if } H[i,j] = H[i-1,j-1]+E(i-1,j-1) \\ 'NIL', & \text{otherwise} \end{cases} \tag{3.24}$$

where $E(i, j)$ is the overall energy of the alignment between query residue i and template position j.

III. Trace Back

The trace back procedure is as in Figure 3.2 in which array aln is the alignment result and $aln[i]$ is the query residue location aligned to position $(i-1)$ of the template.

The main part of the dynamic programming is filling in the six $n \times m$ table, which costs $O(mn)$ running time. The trace back procedure is in $O(m+n)$. So the total running time of the global alignment is in $O(mn)$.

3.2.2 Scanning Algorithm

Suppose the template has C cores in total and the array $a[1, ..., C]$ contains the aligned query locations of the starting positions of the cores. Then, we can specify a radius ρ of the search range for each core. That is, we only search the top images in location range of $[a[i]-\rho, a[i]+\rho]$ on the query sequence. Figure 3.3 depicts the top image scan based on the global alignment result $a[1, ..., C]$.

In **GA-based-Scan-Method**(s, ssa, t), steps 5$\sim$8 is still in constant time if the maximum length of cores is considered to be a constant. Thus the running time of **GA-based-Scan-Method**(s, ssa, t) is in $O(C\rho)$. The overall time of GA-based scan method including the global alignment is in $O(mn)$ considering C and ρ as constants. Usually, m and n are comparable. So, we can say the GA-based scan method is in $O(n^2)$ which is 1 magnitude slower than direct-scan method.

Global alignment aims at getting a *global* optimal alignment between the query and the template, which may sacrifice local optimal alignment during the alignment procedure. Especially, when some template cores do not have corresponding true images on the query sequence (these cores might have been deleted in the evolution), it will be difficult for the global alignment to correctly align the rest template cores to their counterparts on the query.

Trace-Back

```
1     E_min ← H[n, m]
2     A_min ←' FROM − H − TABLE'
3     if E_min > T[n, m]
4         E_min ← T[n, m]
5         A_min ←' FROM − T − TABLE'
6     if E_min > Q[n, m]
7         E_min ← Q[n, m]
8         A_min ←' FROM − Q − TABLE'
9     i ← n
10    j ← m
11    while (i > 0 or j > 0)
12        if A_min ='' FROM − H − TABLE'
13            tmpAln[j] ← 1
14            A_min ← B_H[i][j]
15            i ← i − 1
16            j ← j − 1
17        else if (A_min ='' FROM − Q − TABLE')
18            A_min ← B_Q[i][j]
19            i ← i − 1
20        else if (A_min ='' FROM − T − TABLE')
21            tmpAln[j] ← 1
22            A_min ← B_T[i][j]
23            j ← j − 1
24        else
25            print("ERROR")
26            EXIT
27    aln[0] ← 0
28    for j : n downto 0
29        aln[n − j + 1] ← aln_[n − j] + tmpAln[j]
30    return aln
```

Figure 3.2: The trace back algorithm of the dynamic programming global alignment

GA-based-Scan-Method(s, ssa, t)

```
1     for i : 1 to C
2         bs ← max{0, a[i] − ρ}
3         e ← min{|q| − 1 − |c_i|, a[i] + ρ}
4         for j : b to e
5             q ← s[b, ..., (i + |c_i| − 1)]
6             ss ← ssa[i, ..., (i + |c_i| − 1)]
7             img ← Eval-Image(c_i, q, ss)
8             Handle-New-Image(c_i, img)
```

Figure 3.3: Global-alignment-based scan method

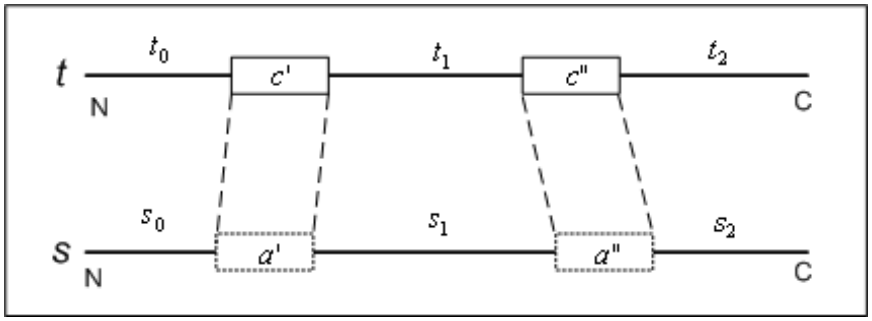

Figure 3.4: Anchor-based hybrid scan method

Such wrong alignment will produce wrong reference locations for the top image scanning of some cores, consequently degenerating the preprocessing results.

3.3 Anchor-based Hybrid Method

To absorb strong local signals and integrate the relative location information of cores in the overall image scan, an anchor-based hybrid method has been designed for threading processing. In the reminder of the thesis, we also call it **hybrid scan** sometimes. This method consists of three steps (see Figure 3.4):

1. Search top images for each core with the direct scan method. Analyze the images of each core and pick the top-1 image, which could be the true one with high confidence, as an anchor. As shown in Figure 3.4, suppose images a' and a" are believed to be the true image of template cores c' and c". Then, we use a' and a" as two anchors.

2. Break query sequence and the template into subsequences and subtemplates with the selected anchors and the corresponding cores; globally align each subsequence-subtemplate pair; according to the alignment results, find the reference locations in

the subsequence for the cores in the subtemplate and set scan range on the subsequence for each of these template cores. As in Figure 3.4, global alignments are done for the pairs s_0-t_0, s_1-t_1 and s_2-t_2.

3. Scan top images for each non-anchor core within its scan range.

The key of anchor-based method is the anchor selection algorithm. Correct anchors could significantly benefit the image scan of other cores while a wrong anchor will dramatically hurt the preprocessing performance. Due to the limitation of the biology knowledge, the author only designed a couple of preliminary algorithm designs of anchor selection, which will be introduced in this section. They have not produced satisfactory results, and more sophisticated methods are expected in the future work. In the experiment evaluation on the performance of anchor-based method, we assume a perfect anchor selection algorithm has been designed, which can pick all the right anchors from top images produced by the direct scan.

3.3.1 Anchor Selection base on the Scores of the Top Images

Assuming the energy functions accurately reflect the sequence-structure compatibility, the energy score of the true image of a template core should be significantly smaller than other images. Based on this assumption, we want to identify the true image from the top images of a template core by comparing the scores of these top images and report this true image as an anchor.

Suppose that direct scan finds the top k images $I_1, I_2, ..., I_k$ for a template core and the energy scores of these images are $e_1, e_2, ..., e_k$. Let $d_1 = |e_2 - e_1|$ and $\bar{d} = (e_k - e_2)/(k-2)$. We set a threshold t for the ratio of $r_d = d_1/\bar{d}$. if the $r_d > t$, image I_1 is judged as an anchor. Otherwise, I_1 is treated as a normal image.

However, although many threshold values were tried, no good value was found to produce both acceptable true positive rate and false positive rate because the energy functions are

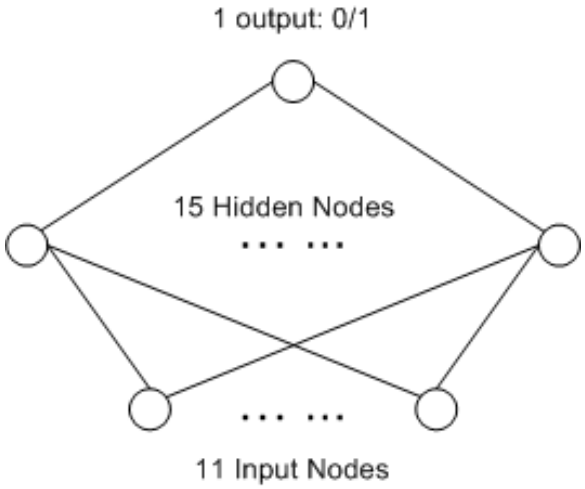

Figure 3.5: BP-neural network for anchor selection

not accurate enough to ensure that the true images always have significantly lower energy values.

3.3.2 Anchor Selection with Artificial Neural Network

Artificial neural networks [1] [4] have been broadly used in solving classification problems. We also tried using back-propagation (BP) neural network to construct an anchor classifier. Figure 3.5 is the architecture of the BP neural network. The network has 11 inputs which are the 11 features about the top images of a template core: the template core length, the energy scores of the top-5 images, the starting positions of the top-5 images on the query. It has 15 intermediate nodes in the hidden layer and 1 output which produces 1 or 0 indicating anchor or non-anchor.

Unfortunately, the training of the neural network could not converge even if only the features of 200 template cores were used. By analyzing the extracted features, it was fount

that, among the 200 template cores, only 17 of them are anchors. That the positive records only take less than 10

Chapter 4

Experimental Evaluation

This chapter demonstrates the experiment evaluations of the three preprocessing methods. The first section describes the methodology of experiments conducted, including the experiment data source, the used energy functions, evaluation metrics and the parameters of the scan methods. The second section evaluates the three methods.

4.1 Experiment Methodology

4.1.1 Experiment Data

In the experiments, dali dataset [6] was used as the evaluation benchmark. Dali dataset provides structure template profiles in .XML format, query sequence profiles in .ssp format and alignment result of query-template pairs.

The structure templates of dali dataset are in profile formats. Figure 4.1 is the file of structure template *1ldm*. In the file, the tag *seq* defines the projected the amino acid sequence; the *dssp* tag includes the secondary structure, the solvent accessibility and some 3D position of each template position; *frequency* gives the amino acid frequencies at each position; *cores* and *two-body total* are the lists of cores and two-body interactions. In *two-body total* part, loops are represented as Core #-1.

```
<?xml version="1.0"?>
<template name="1ldm" version="5">
<seq>
ATLKDKLIGHLATSQEPRSYNKITVVGVGAVGMACAISILMKDLADEVALVDVMEDKLKGEMMDLQHGSLFLHTAKIVSGKDYSVSAGSKLVVITAGAR
QQEGESRLNLVQRNVNIFKFIIPNIVKHSPDCIILVVSNPVDVLTYVAWKLSGLPMHRIIGSGCNLDSARFRYLMGERLGVHSCSCHGWVIGEHGDSVP
SVWSGMNVASIKLHPLDGTNKDKQDWKKLHKDVVDSAYEVIKLKGYTSWAIGLSVADLAETIMKNLCRVHPVSTMVKDFYGIKDNVFLSLPCVLNDHGI
SNIVKMKLKPNEEQQLQKSATTLWDIQKDLKF
</seq>
<dsspData>
REM F RS NUM SS ACC   x-Ca     y-Ca     z-Ca     x-Cb     y-Cb     z-Cb     x-Cm     y-Cm     z-Cm      PHI     PSI
RES 1 A    1  L 106  -36.407    4.166   19.295  -37.830    3.944   18.826  -37.830    3.944   18.826   360.0   117.3
RES 1 T    2  L  82  -34.410    7.042   17.837  -33.401    7.609   18.899  -33.080    8.131   18.891  -159.9   176.6
RES 2 L    3  H 136  -32.653    6.427   14.463  -32.185    7.359   13.359  -32.241    6.767   12.032   -62.9   -61.0
RES 2 K    4  H 147  -29.536    5.080   16.123  -28.563    6.028   16.836  -26.203    5.978   17.749   -51.6   -42.3
RES 2 D    5  H  96  -31.432    3.131   18.819  -32.315    3.698   19.916  -31.586    4.386   21.095   -66.3   -33.6
 ... ...
</dsspData>
<frequency>
0.430915 0.020124 0.020386 0.020659 0.009642 0.020786 0.030082 0.045057 0.008096 0.017990 0.030156 0.028153 0.009303 0.010618 0.023570 0.187894 0.040301 0.003259 0.010559 0.032452
0.095536 0.020161 0.031155 0.022863 0.008107 0.020370 0.038060 0.032668 0.007615 0.028335 0.031978 0.026908 0.009643 0.010556 0.020706 0.180488 0.356482 0.003149 0.010400 0.044820
0.087189 0.014478 0.010824 0.011344 0.007729 0.013405 0.016469 0.016948 0.005247 0.120198 0.306287 0.016827 0.059537 0.020423 0.013630 0.027301 0.073692 0.003738 0.012198 0.162537
0.050292 0.123021 0.021583 0.020886 0.004144 0.052613 0.039634 0.063227 0.009198 0.013776 0.051937 0.399799 0.023042 0.008701 0.018242 0.035410 0.033254 0.002871 0.009562 0.018809
0.050126 0.022678 0.094274 0.133778 0.004147 0.093711 0.212639 0.114276 0.013779 0.010452 0.018210 0.031790 0.006343 0.007272 0.029049 0.097909 0.026588 0.002584 0.008279 0.022118
 ... ...
</frequency>
<cores>
Total = 15
    Left   Right  Length
      22      25       4
      32      39       8
      46      50       5
      57      64       8
      76      78       3
 ... ...
</cores>
<twobody_total>
Total = 609
     RS#     RS#   Core#   Core#
       0       4      -1      -1
       0       5      -1      -1

 ... ...

     287     299      13      14
     288     300      13      -1

 ... ...
</twobody_total>

 ... ...
 ... ...

</template>
```

Figure 4.1: Structure template profile *1ldm*

```
1 A C 0.98739 0.00203 0.01057 |  0.26932 0.02386 0.02862 0.02697 0.01009 0.02480 0.03595 0.04725 0.00971 0.01776 0.02991 0.03295 0.00966 0.01150 0.02599 0.33171 0.05225 0.00350 0.01173 0.03068
2 R C 0.97585 0.01975 0.00440 |  0.03500 0.30519 0.03416 0.01986 0.00360 0.07812 0.03748 0.02099 0.00975 0.00999 0.02065 0.28371 0.00729 0.00720 0.01603 0.07564 0.02408 0.00262 0.00898 0.01482
3 K E 0.48793 0.51201 0.00006 |  0.02756 0.08834 0.02454 0.01687 0.00277 0.02635 0.03242 0.01749 0.00703 0.00831 0.01585 0.64231 0.00569 0.00545 0.01449 0.02700 0.02756 0.00195 0.00696 0.01208
4 I E 0.26465 0.73534 0.00000 |  0.03366 0.00820 0.00640 0.00729 0.00545 0.00749 0.00989 0.01357 0.00308 0.29493 0.07588 0.01004 0.01218 0.01305 0.00876 0.01495 0.02068 0.00217 0.01382 0.40861
5 G E 0.00583 0.99411 0.00006 |  0.27821 0.01239 0.01357 0.01289 0.01175 0.01235 0.01734 0.08720 0.00479 0.03054 0.03182 0.01935 0.01231 0.01192 0.01357 0.13801 0.18548 0.00221 0.00722 0.10484
6 I E 0.00312 0.99685 0.00002 |  0.02819 0.00841 0.00623 0.00701 0.00537 0.00760 0.00984 0.00925 0.00306 0.21687 0.16210 0.00996 0.02012 0.02533 0.00855 0.01448 0.02001 0.00229 0.00881 0.39744
7 I E 0.00756 0.99240 0.00004 |  0.02793 0.00855 0.00689 0.00754 0.01361 0.00773 0.00994 0.02793 0.00313 0.28719 0.18594 0.01024 0.01469 0.01337 0.00869 0.01587 0.05573 0.00227 0.00814 0.25733
8 G E 0.02404 0.97299 0.00297 |  0.04719 0.00931 0.01528 0.01357 0.00682 0.00935 0.01591 0.77367 0.00434 0.00583 0.01051 0.01362 0.00361 0.00527 0.01002 0.03218 0.01385 0.00222 0.00454 0.00903
9 L E 0.38660 0.60445 0.00895 |  0.40781 0.02544 0.01370 0.03813 0.01718 0.01203 0.01746 0.05049 0.00467 0.03393 0.04200 0.01642 0.02049 0.00782 0.01319 0.10436 0.05825 0.00209 0.00687 0.11199
10 G C 0.94204 0.02483 0.03313 |  0.03836 0.01378 0.01525 0.02346 0.00322 0.00932 0.01264 0.75242 0.00430 0.00566 0.01018 0.01360 0.00351 0.00511 0.00984 0.05669 0.01414 0.00213 0.00445 0.00872
11 N C 0.65636 0.02228 0.32136 |  0.11923 0.04409 0.11826 0.04217 0.00376 0.11584 0.03086 0.20179 0.05062 0.01010 0.02325 0.04326 0.04362 0.04210 0.01166 0.04940 0.02497 0.00258 0.01495 0.01417
12 V H 0.30828 0.07697 0.61475 |  0.02443 0.00844 0.00685 0.00773 0.00531 0.00768 0.01019 0.02107 0.00313 0.30426 0.07405 0.01028 0.01657 0.01248 0.00898 0.01526 0.03311 0.00217 0.00815 0.38731
13 G H 0.48163 0.02830 0.49007 |  0.04007 0.00989 0.01517 0.01657 0.00341 0.02083 0.01322 0.75441 0.00452 0.01452 0.01194 0.01403 0.00391 0.00568 0.01037 0.03585 0.01417 0.00232 0.00489 0.01007
14 A H 0.00262 0.00304 0.99434 |  0.11065 0.02397 0.02649 0.01930 0.01386 0.18093 0.03218 0.09545 0.00689 0.03204 0.02980 0.02070 0.08859 0.00957 0.01230 0.16544 0.04872 0.00288 0.04792 0.04146
 ... ...
```

Figure 4.2: Query sequence profile *1hyha*

```
-1, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, ... ...

-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, ... ...
```

Figure 4.3: Alignment file between sequence *1hyha* and the template *1ldm*

The provided query sequences are protein sequence profiles instead of single protein sequences. Figure 4.2 is a part of the sequence profile *1hyha*. In each row, the first column is the residue position; the second column is the amino acid type at the position; the third column is the predicted secondary structure at this position. The possible values are H (α-helix), B (residue in isolated β-bridge), E (extended strand), G (3_{10}-helix), I (π-helix), T(hydrogen bonded turn), S (bend) and C (Cori) [7]. Following the widely used convention, H, G and I are classified as helices, E and B states are classified as strand, and all others are as loop [8]. The following three columns are the probabilities of the position being in the three types of secondary structures (loop, α-helix and β-strand); the 20 numbers behind "|" are the frequencies of the 20 amino acid types at the position.

The alignment file provides the position-to-position alignments between the query sequence and the template profile. Figure 4.3 is the alignment file between the sequence *1hyha* and the template *1ldm*. In that figure, the first 2 numbers in the first line mean that the first 2 residues of the query sequence are aligned to a gap and the 20th position of the template respectively. Similarly, the second line says that the first 15 template positions are aligned to all gaps.

4.1.2 Energy Function Parameter Values

In the experiments, we used an energy function improved from the one used in PROSPECT II [8]. Tables 4.1, 4.2 and 4.3 are respectively the two-body interaction energy matrix M_I, the singleton energy matrix M_S and the mutation energy matrix M_M used in the experiments.

Table 4.1: The two-body interaction energy matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	-453																			
R	171	427																		
N	264	344	-89																	
D	377	32	211	706																
C	-361	66	39	336	-2386															
Q	220	316	203	410	144	212														
E	391	5	425	921	448	434	790													
G	-81	200	41	244	-278	265	524	-246												
H	94	249	192	122	-367	250	340	38	-266											
I	-523	149	222	443	-591	180	362	6	-46	-940										
L	-488	83	341	540	-512	137	386	101	-9	-698	-698									
K	396	846	474	135	314	433	53	449	549	275	368	725								
M	-342	152	166	432	-576	193	441	-64	-104	-514	-434	402	-732							
F	-302	143	200	474	-554	184	485	-71	-91	-576	-515	384	-539	-601						
P	145	359	289	496	-159	346	546	108	148	92	126	679	25	-0	124					
S	91	251	55	228	-194	216	374	-6	32	53	125	397	46	33	198	-14				
T	-46	238	20	252	-213	155	360	-49	7	-196	-49	343	-37	-1	165	-13	-161			
W	-165	-55	63	424	-517	53	507	-83	-163	-348	-353	263	-396	-462	-234	58	46	-604		
Y	-202	-43	115	398	-472	123	438	-74	-84	-496	-408	101	-410	-451	-117	75	11	-433	-471	
V	-504	104	230	422	-612	174	371	-42	-53	-886	-721	297	-497	-538	48	45	-222	-320	-444	-946

Table 4.2: The singleton energe matrix

	α			β			Loop		
	buried	intermediate	exposed	buried	intermediate	exposed	buried	intermediate	exposed
A	-0.627	-0.122	-0.070	-0.026	0.699	1.148	-0.056	0.248	0.430
R	0.935	-0.595	-0.579	1.188	-0.445	-0.663	0.924	-0.068	-0.085
N	0.858	0.213	-0.072	0.851	0.259	-0.083	0.089	-0.280	-0.571
D	1.155	0.198	-0.454	1.108	0.310	-0.086	0.351	-0.179	-0.652
C	-0.383	0.600	2.421	-0.614	0.278	1.628	-0.727	-0.068	1.522
Q	0.652	-0.433	-0.796	0.980	-0.056	-0.550	0.851	0.154	-0.219
E	1.017	-0.344	-1.079	1.404	0.072	-0.749	1.326	0.485	-0.372
G	0.477	0.995	1.406	0.212	0.559	1.460	-0.483	-0.407	-0.481
H	0.432	-0.210	0.219	0.245	-0.326	0.023	-0.031	-0.266	0.016
I	-0.559	0.185	1.456	-0.852	-0.114	0.641	-0.159	0.541	1.506
L	-0.721	-0.162	1.000	-0.388	0.263	0.908	-0.224	0.241	1.275
K	1.645	-0.312	-0.895	1.826	-0.254	-0.889	1.891	0.236	-0.507
M	-0.656	-0.158	0.992	-0.250	0.245	0.980	-0.240	0.191	0.762
F	-0.470	0.229	1.490	-0.601	-0.086	1.044	-0.438	0.093	1.379
P	1.181	0.755	0.497	1.173	0.729	0.281	-0.430	-0.573	-0.559
S	0.359	0.346	0.050	0.303	0.056	-0.111	-0.145	-0.205	-0.270
T	0.256	0.180	0.356	0.088	-0.403	-0.574	-0.011	-0.096	-0.067
W	-0.374	-0.197	1.533	-0.295	-0.455	0.821	-0.338	-0.047	1.312
Y	-0.100	-0.233	0.950	-0.293	-0.619	0.228	-0.051	-0.153	0.927
V	-0.370	0.270	1.365	-0.893	-0.317	0.322	-0.039	0.406	1.208

Table 4.3: The mutation energy matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2																			
R	-2	6																		
N	0	0	2																	
D	0	-1	2	4																
C	-2	-4	-4	-5	12															
Q	0	1	1	2	-5	4														
E	0	-1	1	3	-5	2	4													
G	1	-3	0	1	-3	-1	0	5												
H	-1	2	2	1	-3	3	1	-2	6											
I	-1	-2	-2	-2	-2	-2	-2	-3	-2	5										
L	-2	-3	-3	-4	-6	-2	-3	-4	-2	2	6									
K	-1	3	1	0	-5	1	0	-2	0	-2	-3	5								
M	-1	0	-2	-3	-5	-1	-2	-3	-2	2	4	0	6							
F	-4	-4	-4	-6	-4	-5	-5	-5	-2	1	2	-5	0	9						
P	1	0	-1	-1	-3	0	-1	-1	0	-2	-3	-1	-2	-5	6					
S	1	0	1	0	0	-1	0	1	-1	-1	-3	0	-2	-3	1	2				
T	1	-1	0	0	-2	-1	0	0	-1	0	-2	0	-1	-3	0	1	3			
W	-6	2	-4	-7	-8	-5	-7	-7	-3	-5	-2	-3	-4	0	-6	-2	-5	17		
Y	-3	-4	-2	-4	0	-4	-4	-5	0	-1	-1	-4	-2	7	-5	-3	-3	0	10	
V	0	-2	-2	-2	-2	-2	-2	-1	-2	4	2	-2	2	-1	-1	-1	0	-6	-2	4

Since the queries are sequence profiles instead of single proteins, the calculations of singleton, mutation and two-body interactions are modified accordingly to adapt this change. Formulas 4.2, 4.3 and 4.1 respectively replace 2.5, 2.7 and 2.3 in the energy calculations.

$$e_p(i) = \sum_{a=1}^{20} \sum_{b=1}^{20} (p_{l_i}(a) \cdot p_{r_i}(b) \cdot MI(a,b)) \tag{4.1}$$

$$e_s(j,i) = \sum_{t=1}^{20} (p(j_t) \cdot MS(r_t, ss_i, sa_i)), \tag{4.2}$$

and,

$$e_m(j,i) = \sum_{p=1}^{20} \sum_{t=1}^{20} (p_j(q) \cdot p_i(t) \cdot MM(q,t)), \tag{4.3}$$

where i represents the i-th position of the template, j is the j-th position on the query, $p_{l_i}(a)$ is the frequency of amino acid type a at the query position which aligned to the left position of the i-th two-body interaction on the template and $p_{r_i}(b)$ is the counterpart of $p_{l_i}(a)$.

In the gap penalty calculation, the value of γ equals 10.8 and δ is set as 0.6. That is, the gap penalty function is as equation 4.4.

$$e_g = 10.8 + 0.6(|g| - 1)) \tag{4.4}$$

where $|g|$ is the length of the continuous gap.

In the secondary structure matching energy function, τ equals 3.0 and ρ takes 10.0. That is, the used secondary structure function is as equation 4.5.

$$e_{ss}(r, i) = 3.0 - rint(p_r(ss_i) \cdot 10.0) \tag{4.5}$$

The table 4.4 are the relative weights of the five energy terms used in the experiment.

Table 4.4: The relative weights of the five energy terms

w_s	w_m	w_{ss}	w_g	w_p
0.892100	0.002964	0.036929	0.312064	0.324656

4.1.3 Evaluation Metrics

The goal of threading preprocessing is to locate the true image on the query corresponding to each template core. Due to the accuracy of the energy function and many other constraints, it is almost impossible for the preprocessing methods to directly locate the true images on query sequences. So, the methods find the top-k images for each template core to include the true one.

With the preprocessing benchmark, we use *true-image hit rate* to measure the performance of the preprocessing methods. By checking the alignment file of sequence-template pair, we can find out the subsequence of the query aligned to a template core. If the query sequence file indicates the most residues of this subsequence have the same secondary structure as the template core, we say this subsequence is the *true-image* on the query corresponding to the template core. If the top-k images contain the true one, we say the true image is *hit*.

If the most residues of the aligned subsequence do not have the same secondary structure as the template core, it means the template core does not have a corresponding true-image on the query. It could have been deleted in the evolution.

If a template core does have the corresponding true-image on the query sequence while the top-k candidate images found by a preprocessing method do not include the true-image, we say the preprocessing method *missed* the true-image of the template core.

In the preprocessing of a query-template pair, the *true-image hit rate* equals the number of true-image hits over the number of template cores who have corresponding true-images in the query sequence.

4.2 Experimental Evaluation

From the dali dataset, 581 applicable sequence-template pairs are used in experiments to evaluate the performance of the three methods. For the simplicity of description, we use **hit rate** to representing **true-image hit rate** sometimes.

4.2.1 Performance on All Sequence-Template Pairs

In the first step, we applied the three methods on all the 581 sequence-template pairs to locate the top images for each template core. On the templates of these 581 pairs, there are 7088 template cores in total, 3958 of which have the corresponding true-images on the query sequences. Table 4.5 shows the average true-image hit rates by top 1, top 5 and top 10 images of the three methods where all the 581 pairs are considered.

Table 4.5: Average true-image hit rate of the three methods

	Direct Scan	GA-based Scan	Hybrid Scan
Top-1	8.94%	21.53%	26.20%
Top-5	27.13%	38.98%	45.68%
Top-10	39.54%	45.91%	53.21%
Top-15	48.38%	48.81%	56.77%

From Figure 4.5, we have two observations:

1. Hybrid scan achieves the best performance at all the three levels; GA-based scan follows hybrid scan; direct scan is always the worst one and the hit rate of top-15 images is still less than 50%.

2. The hit rates of GA-based scan and hybrid scan significantly outperform direct scan at top-1 image and top-5 images levels. Even the top-5 image hit rate of hybrid scan is much better than the top-10 image hit rate of direct scan.

The first observation tells us that the current energy functions are far from good enough to be the solo factor to locate true images of template cores and that integrating the relative location information of cores is very helpful to preprocessing.

The second observation gives us one hint. When the tree-decomposition is the bottleneck of the whole threading procedure, getting fewer top images for each core, using GA-based scan and hybrid scan, will speed up the tree-decomposition computation but not hurt the structure prediction accuracy.

The bigger number of the top images preprocessing used means the largger time cost of the following steps of threading. Usually, this number is no larger than 10. So, all the result analysis will be based on the true image hit rate at top 10 image level.

To investigate the performance of the three methods on concrete sequence-template pairs, we break the top-10 image true-hit rate into five bins [0, 0.2], (0.2, 0.4], (0.4, 0.6], (0.6, 0.8] and (0.8, 1], and count the number of pairs falling in each bin among the preprocessing result of each method. Figure 4.4 is the constructed histogram.

From Figure 4.4, we can see that,

- For direct scan method, the number of pairs falling in each bin decreases as the hit rate labeling the bins increases;

- For both GA-based scan and hybrid scan, the pair numbers in the first bin and last bin are much larger than the numbers in the middle bins.

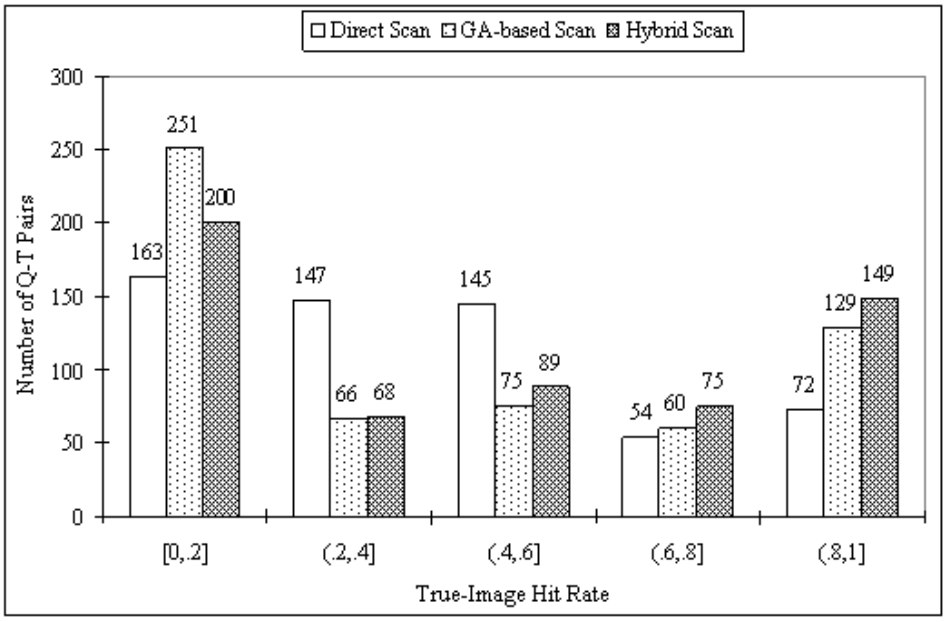

Figure 4.4: The histogram of the number of sequence-template pairs by true-image hit rate

These observations are very interesting. If we can find out what kind of sequence-template pairs can be preprocessed by GA-based scan and hybrid scan with very good accuracy and what kind of pairs can be preprocessed by direct scan with better performance, we can use each method in appropriate situation to get a better overall performance. By analyzing the data pairs in each bin, we found GA-based scan and hybrid scan are sensitive to the length difference between the query and the template. Next subsection will investigate the effect of the length difference on the three methods.

4.2.2 The Effect of the Length Difference Between the Query and the Template

To simply represent the length difference between the query and the template, one term is defined: *length difference ratio*. If the template is longer than the query, the *length difference*

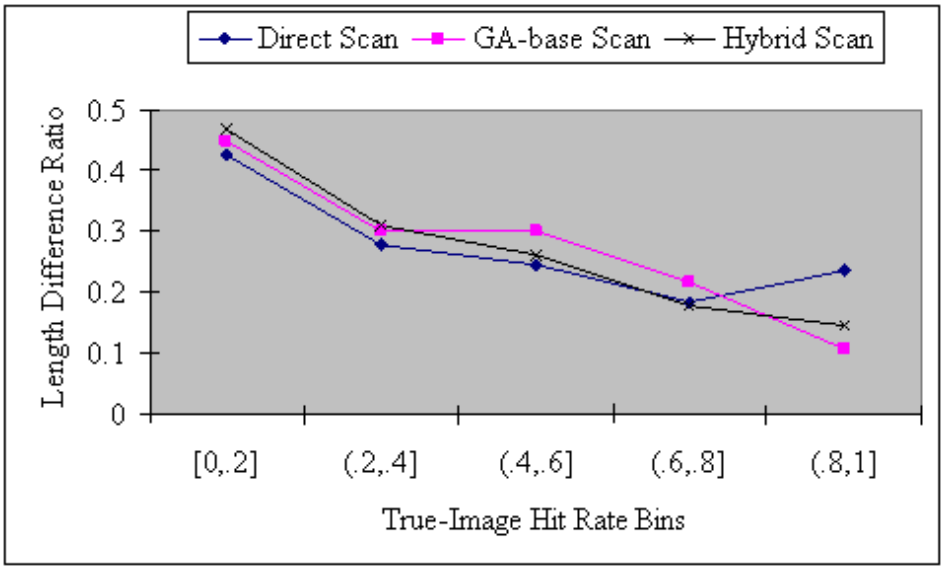

Figure 4.5: The average length difference ratios in each true-image hit rate bin

ratio equals the length difference between the template and the query divided by the length of the template; otherwise, it equals the length difference divided by the query sequence length.

Figure 4.5 depicts the average length difference ratio of each hit rate bin for all the three methods, which is corresponding to the histogram in Figure 4.4. From Figure 4.5, we can see that, as the hit rate increases, the average length difference ratios of GA-based scan and hybrid scan monotonously decrease. The sensitivity of GA-based scan and hybrid scan to query-template length difference also indicates the potential problems existing in energy functions, especially the gap penalties.

We also investigated the effect of query-template length difference in another way. We group all the query-template pairs into 6 bins by their length difference ratio where the ratio bins are [0, 0.1], (0.1, 0.2], (0.2, 0.3], (0.3, 0.4], (0.4, 0.5] and (0.5, 1]. Table 4.6 provides the characteristics of the pairs in these bins. The average true-image hit rates of the three

Table 4.6: Characteristics of query-template pairs in the six difference ratio bins

len-diff ratio bins	[0, 0.1]	(0.1, 0.2]	(0.2, 0.3]	(0.3, 0.4]	(0.4, 0.5]	(0.5, 1]	Total
number of q-t pairs	171	105	63	71	39	132	581
number of cores	2071	1396	784	1074	591	1172	7088
number of cores having true images	1390	830	424	531	271	512	3958

methods in each bin are demonstrated in Figure 4.6, which confirm the sensitivity of GA-based scan and Hybrid scan to the length difference ratio. In particular, we find that, the average true-image hit rates of GA-based scan and Hybrid scan are around 70% when the length difference ratio is less than 0.1; when the length difference ratio is larger than 0.4, direct scan method outperforms the two other methods.

4.2.3 the Effect of Anchors

In this subsection, another term is defined: *anchor ratio*, which equals the number of correctly located anchors over the number of those cores having true images on the query sequence. To investigate the effect of anchors to the hybrid scan methods, we group all the query-template pairs into 6 bins by their anchor ratio where the ratio bins are [0, 0.1], (0.1, 0.2], (0.2, 0.3], (0.3, 0.4], (0.4, 0.5] and (0.5, 1]. Table 4.7 describes the pairs in these bins. The average true-image hit rates of the three methods in each bin are demonstrated in Figure 4.7.

From Figure 4.7, we can see that:

1. The performances of all the three methods increase as the bins move. More anchors found among a query-template pair mean more locally strong signals existing, which benefit both the direct scan and the global alignment between the query-template pairs.

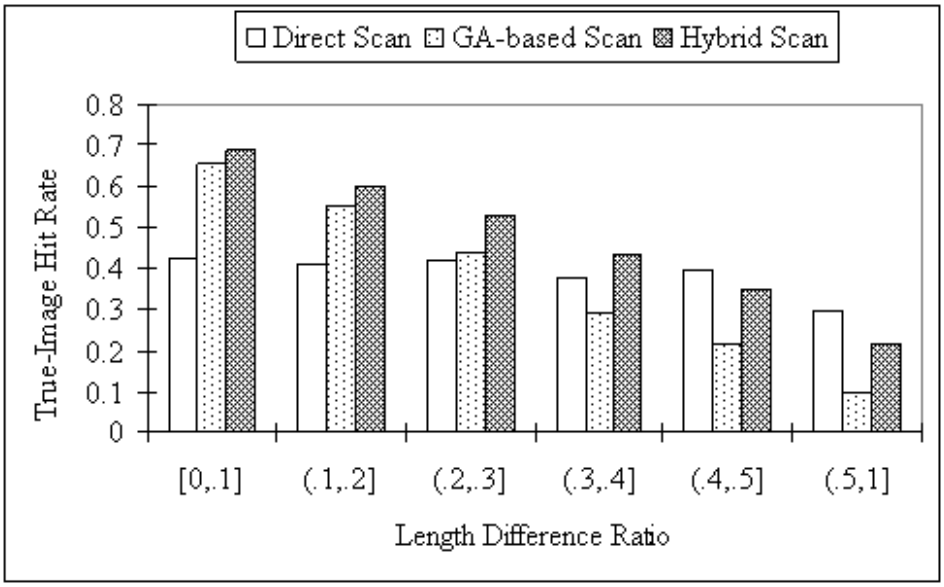

Figure 4.6: Average true-image hit rates of the three methods in each length difference ratio bin

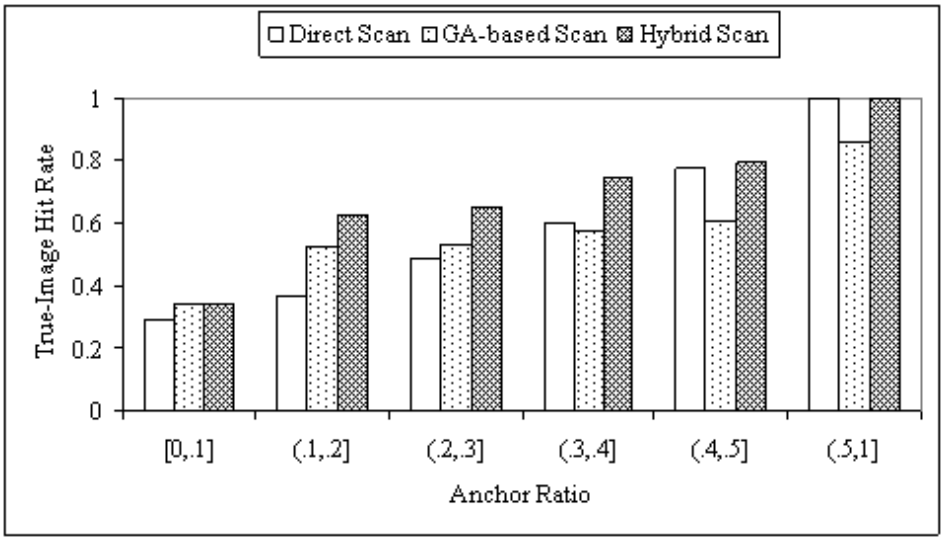

Figure 4.7: Average true-image hit rates of the three methods in each anchor ratio bin

Table 4.7: Characteristics of query-template pairs in the six anchor ratio bins

anchor ratio bins	[0, 0.1]	(0.1, 0.2]	(0.2, 0.3]	(0.3, 0.4]	(0.4, 0.5]	(0.5, 1]	Total
number of q-t pairs	324	70	102	53	22	10	581
number of cores	3451	1474	1457	561	122	23	7088
number of cores having true images	1587	1040	944	317	53	17	3958

2. Hybrid scan outperforms GA-based scan much in most bins except the first one, which means these anchors do help the preprocessing.

Chapter 5

Conclusion

Threading has become one of the most effective methods for protein tertiary structure prediction and it exhibits big potential on structure prediction for those proteins whose homologous sequences are not found among structure-solved proteins. To solve the NP-hard problem caused by the two-body interactions, many threading methods begin with a preprocessing phase in which several top images are picked out for each core on template. The following steps of threading modeling are usually all based on combinatorial computation of these images. Whether the top images really contain the true ones on the query sequence determines structure prediction accuracy of the threading methods.

This thesis focused on threading preprocessing. First, we identified the components of threading preprocessing and recognized the two factors to locate top images for each template core: the knowledge-based energy value and the relative location of the cores. Second, we introduced three preprocessing methods: direct scan, global alignment based scan and anchor based hybrid scan and evaluated the methods with experiments on dali benchmark. From the results, we found that global alignment based scan and hybrid scan perform very well when the lengths of the query and template are close while the direct scan method is insensitive to the length difference between query and the template. Based on the overall performance of all the three methods, we feel that the current available energy functions are the primary constraint for the preprocessing performance. Only using these energy functions are far from to make good top image scan.

Due to the limitation of the biology knowledge, we only did some preliminary algorithm design of anchor selection, which could not produce satisfactory results. We hope more

sophisticated methods could be proposed with the help and collaboration from biologists in future. A good anchor selection algorithm will significantly improve the preprocessing performance which was indicated by the experiment evaluation on the anchor effects.

Bibliography

[1] Abdi, H. (1994). A neural network primer. *Journal of Biological Systems*. 2: 247-281

[2] Anfinsen, C. B. (1973). Principles that govern the folding of protein chains. *Science*. 181: 223-230.

[3] Arnborg, S., and Proskurowski, A. (1989). Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*. 23: 11-24

[4] Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

[5] Chan, H. S., Kaya, H., and Seishi, S. (2002). Computational Methods for Protein Folding: Scaling a Hierarchy of Complexities. *Current Topics in Computational Molecular Biology*. Tao, J., Xu, Y. and Zhang, M. Q.(eds), 403-448, The MIT Press

[6] Holm, L., and Sander, C. (1997). Decision support system for evolutionary classification of protein structures. *Proceedings of International Conference on Intelligent Systems for Molecular Biology*. 5: 140-146

[7] Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*. 22(12): 2577-2637

[8] Kim, D., Xu, D., Guo, J., Ellrott, K., and Xu, Y. (2003). Prospect II: protein structure prediction program for genome-scale applications. *Protein Engineering*. 16(9): 641-650.

[9] Lathrop, R. H. (1994). The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Engineering*. 7(9): 1059-1068.

[10] Leaver-Fay, A., Kuhlman, B., and Snoeyink, J. (2005). An adaptive dynamic programming algorithm for the side chain placement problem. *Proceedings of the Pacific Symposium on Biocomputing*. 10: 16-27.

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

[12] Pedersen, J. T., and Moult, J. (1997). Protein folding simulations with genetic algorithms and a detailed molecular description. *Journal of Molecular Biology*. 269: 240-259.

[13] Peitsch, M. C., Schwede, T., Diemand, A., and Guex, N. (2002). Protein Structure Prediction by Comparison: Homology-Based Modeling. *Current Topics in Computational Molecular Biology*. Tao, J., Xu, Y. and Zhang, M. Q.(eds), 449-466, The MIT Press

[14] Rost, B., Sander, C. and Schneider, R. (1994). PHD–an automatic mail server for protein secondary structure prediction. *Computer Applications in the Biosciences*. 10: 53-60.

[15] Sali, A., Shakhnovich, E., and Karplus, M.(1994). Kinetics of protein folding. A lattice model study of the requirements for folding to the native state. *Journal of Molecular Biology*. 235: 1614-1636.

[16] Smith, T. F., Conte, L. L., Bienkowska, J., Gaitatzes, C., Rogers, R., and Lathro, R. (1997). Current limitations to protein threading approaches. *Journal of Computational Biology*. 4(3): 217-225

[17] Song, Y. (2006). Effective Models and Efficient Algorithms for Structural Bioinformatics (PhD Dissertation). The University of Georgia

[18] Song, Y., Liu, C., Malmberg, R., Pan, F., and Cai, L. (2005). Tree decomposition-based fast RNA pseudoknot search in genomes. *Proceedings of IEEE Computer Society Computational Systems Bioinformatics Conference*. 223-234

[19] Wang, Z. X. (1998). A re-estimation for the total numbers of protein folds and superfamilies. *protein Engineering*. 11: 621-626

[20] Wooley, J. C., and Ye, Y. (2006). A Historical Perspective and Overview of Protein Structure Prediction. *Computational Methods for Protein Structure Prediction and Modeling*. Vols I, Xu, Y., Xu, D. and Liang, J. (eds), 1-43, Springer

[21] Xiang, Z. (2006). Homology-Based Modeling of Protein Structure. *Computational Methods for Protein Structure Prediction and Modeling*. Vols II, Xu, Y., Xu, D. and Liang, J. (eds), 319-358, Springer

[22] Xu, J. (2005). Rapid Problem Side-Chain Packing via Tree Decomposition. *Proceedings of the Ninth Annual International Conference on Research in Computational Molecular Biology*. 423-429.

[23] Xu, J., Jiao, F., and Berger, B. (2005). A Tree-Decomposition Approach to Protein Structure Prediction. *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference*. 247-256.

[24] Xu, J., Li, M., Kim, D., and Xu, Y. (2003). RAPTOR: Optimal Protein Threading by Linear Programming. *Journal of Bioinformatics and Computational Biology*. 1: 95-117.

[25] Xu, J., Li, M., Kim, D., and Xu, Y. (2003). Protein threading by linear programming. *Biocomputing: Proceedings of the 2003 Pacific Symposium*. 264-275.

[26] Xu, Y., and Xu, D. (2002). Protein Structure Prediction by Threading and Partial Experimental Data. *Current Topics in Computational Molecular Biology*. Tao, J., Xu, Y. and Zhang, M. Q.(eds), 467-502, The MIT Press

[27] Xu, Y., and Xu, D. (2000). Protein threading using PROSPECT: Design and evaluation. *Proteins*. 40: 343-354.

[28] Xu, Y., Liu, Z., Cai, L., and Xu, Dong. (2006). Protein Structure Prediction by Protein Threading. *Computational Methods for Protein Structure Prediction and Modeling*. Vols II, Xu, Y., Xu, D. and Liang, J. (eds), 1-42, Springer

[29] Xu, Y., Xu, D., and Uberbacher, E. C. (1998). An efficient computational method for globally optimal threading. *Journal of Computational Biology*. 5(3): 597-614

[30] Zhang, B., Jaroszewski, L., Rychlewski, L., and Godzik, A. (1997). Similarities and differences between nonhomologous proteins with similar folds: Evaluation of threading strategies. *Folding and Design*. 2: 307-317