New Approaches to Protein Structure Prediction

by

Shuai Cheng Li

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2009

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Protein structure prediction is concerned with the prediction of a protein's three dimensional structure from its amino acid sequence. Such predictions are commonly performed by searching the possible structures and evaluating each structure by using some scoring function. If it is assumed that the target protein structure resembles the structure of a known protein, the search space can be significantly reduced. Such an approach is referred to as *comparative structure prediction*. When such an assumption is not made, the approach is known as *ab initio structure prediction*. There are several difficulties in devising efficient searches or in computing the scoring function. Many of these problems have ready solutions from known mathematical methods. However, the problems that are yet unsolved have hindered structure prediction methods from more ideal predictions.

The objective of this study is to present a complete framework for *ab initio* protein structure prediction. To achieve this, a new search strategy is proposed, and better techniques are devised for computing the known scoring functions. Some of the remaining problems in protein structure prediction are revisited. Several of them are shown to be intractable. In many of these cases, approximation methods are suggested as alternative solutions. The primary issues addressed in this thesis are concerned with local structures prediction, structure assembly or sampling, side chain packing, model comparison, and structural alignment. For brevity, we do not elaborate on these problems here; a concise introduction is given in the first section of this thesis.

Results from these studies prompted the development of several programs, forming a utility suite for *ab initio* protein structure prediction. Due to the general usefulness of these programs, some of them are released with open source licenses to benefit the community.

Acknowledgements

This thesis would not be possible without the supervision of my advisors Prof. Ming Li and Dr. Jinbo Xu. I am most indebted to Prof. Li for sharing with me his wealth of knowledge, and his guidance to solve many interesting problems. I thank Dr. Xu for sharing with me his experiences and invaluable insight into protein structures.

This thesis is not possible without the collaboration with Dr. Dongbo Bu, with whom I have discussed and solved many details of the problems with. I also wish to thank Dr. Yen Kaow Ng, who has read many of my manuscripts, and provided many substantial comments.

I thank my thesis examiners, Prof. C. Perry Chou, Prof. Jean-Claude Latombe, Prof. Brendan J. McConkey, and Prof. Ian Munro for sparing their invaluable time to read my thesis.

I also have to thank my friends and colleagues Babak Alipanahi, Xuefeng Cui, Xin Gao, Xi Han, Daniel Holtby, Richard Jang, He Lin, Francis Ng, Dandan Song, Jinsong Tan, Nan Wang, Wang Yue, and Yuzhong Zhao, for working with me and for proofreading my thesis.

This thesis is dedicated to my parents, who have patiently listened to my complaints and encouraged me.

Dedication

This is dedicated to the one I love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Biology would not be what it is today without the laws of hereditary discovered by Gregor Johann Mendel. However, there are many unknown laws in nature, many of which are more elusive than Mendel's laws, and more difficult to discover. Computational methods have emerged as fast and elegant tools to assist in the discovery of such laws. The development and use of computational methods in biology are generally known as *bioinformatics* today. These methods can be applied to discover the laws behind biological systems and to solve routine problems in biology, relieving biologists from their tedious tasks. The former is considered the *informatics aspect* and the latter is the *computational aspect* of bioinformatics. The boundary between the two are blurred on occasions.

This thesis addresses problems from both aspects. In particular, the focus is on *ab initio* protein structure prediction. Protein structure prediction is one of the fundamental problems in bioinformatics and theoretical chemistry. It is known that the amino acid sequence (primary sequence) of a protein folds into some stable structures which correspond to its minimum energy states. The discovery of these protein structures through "wet lab" techniques, such as nuclear magnetic resonance (NMR) spectroscopy, or X-ray crystallography, is time-consuming and costly. The use of protein structure prediction methods can reduce this time and cost in obtaining protein structures by several orders of magnitude. The methods have significant impact on medicine, biotechnology, and related fields.

Most of the difficulties in predicting a protein structure come from the potentially enormous size of the protein structure's conformation space, as well as our only partial understanding of the physical basis behind protein structural stability. These two factors determine the performance of any protein structure prediction method. The process to search through the conformation space is called the *search strategy*, and the physical basis of the structural stability is called the *energy (or score) function*.

To reduce the search space, one approach in protein structure prediction is to assume that a (near) native structure exists in a pool of known structures. The only need is to identify the structure by using an energy function. Protein prediction

methods with this approach are referred to as comparative structure prediction (also called homology modeling) or *threading*. For *ab initio* structure predictions, the native structure is not assumed bo be in any known structure space. They rely on a structure construction method and energy function to discover the native structure.

The design of promising *ab initio* methods is a challenging task. The methods require tremendous amounts of computational power. For this reason, most of the present *ab initio* methods depend on a simplified representation of protein structures, rather than on an atomic level representation. The use of atomic level representation in *ab initio* methods seems unlikely in the near future. The design of energy functions for this class of methods poses another challenge. In spite of the difficulties, *ab initio* protein prediction is under active research today due to the impact and potential benefits [172].

In this thesis, a fragment-based approach to *ab initio* protein structure prediction is adopted. Such an approach involves several steps: local structure prediction, structure assembly or sampling, side chain packing, quality assessment, structural comparison, and alignment. The subproblems of local structure prediction, side chain packing, structure assembly, and quality assessment are explored from both their computational and informatics aspects. Also, the structural comparison and alignment problems are explored from the computational aspect. In this thesis, it is assumed that a plausible energy function is given. Consequently, the focus is on the design of more efficient search strategies. Figure 1.1 provides an overall view of the design. In Chapter 2, preliminary concepts are introduced for further discussions. In Chapter 3, the structural fragment library problem is examined. The structural fragments serve as building blocks for constructing structures. A method for building structures from this fragment library is proposed in Chapter 4. The method results in a backbone structure of the protein. With this backbone structure, branches, called *side chains*, are added to complete the structure. The problem of adding these side chains is studied in Chapter 5. Often, this structure generation is repeated, resulting in a very large collection of structures, called *decoys*. From these structures, a few structures which are believed to be closest to the native structure are selected. They are either reported as the closest structures, or used as references to obtain even more accurate structures. This decoy selection problem is studied in Chapter 6. The next problem is to assess the quality of the reported structures, by comparing them to the native structure (given that the native structure is known). This problem is studied in Chapter 7. Finally, in Chapter 8, various theoretical results of computational problems, related to the alignment of two 3D structures are shown, through extending the proposed decoy comparison techniques in Chapter 7, with other graph theoretic methods.

Now, a brief introduction to each of the problems studied in this thesis is provided.

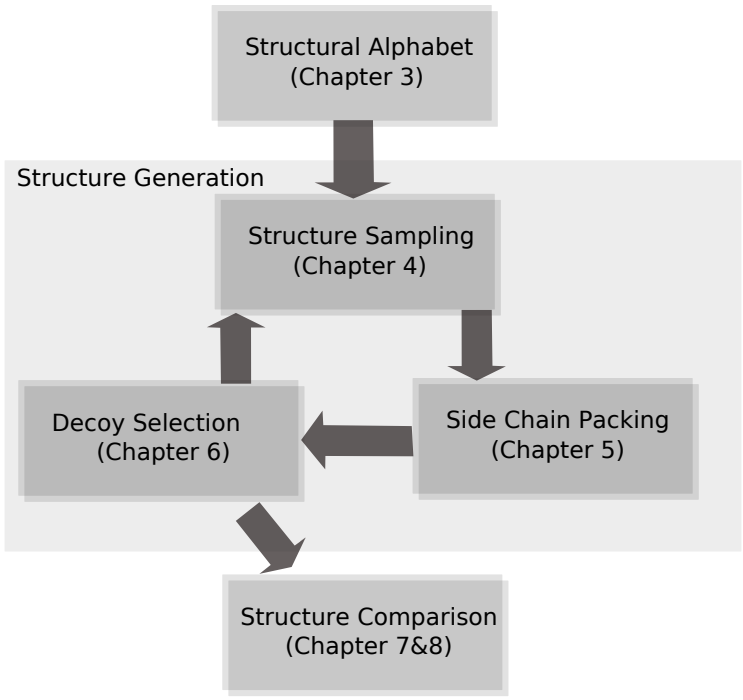


Figure 1.1: Overview of the proposed protein structure prediction method

1.1 Structural Fragment Libraries

A possible approach to the *ab initio* protein structure prediction is to identify small building blocks for a protein structure at the initial stage. These building blocks are called *structural fragment libraries*.

The subproblem is approached from both its computational and informatics aspects. From the computational aspect, the problem studied is on how to select from a set of these building blocks, a small subset that represents the entire set. From the informatics aspect, why a peptide sequence favors certain substructures is examined. The details of the problems follow.

1.1.1 Computational Aspect: The Fragment Library Size

Each structure is considered as a sequence of l points in 3D space, where l is a fixed natural number. Given a set of structures, the idea is to identify a small set of structures (that is, a structural alphabet) that can aptly represent the set.

The problem is formulated as follows. Two structures, f_1 and f_2, are equivalent if and only if there exist transformations R and T such that $f_1 = Rf_2 + T$, where R and T are a rotation and a translation, respectively. The distance between two structures is considered to be the sum of the squared Euclidean distance between all their corresponding points. Given a set of n structures, the problem is to find k, $k \leq n$, structures g_1, g_2, ..., g_k, so as to minimize the sum of the distances between

each of f_1, f_2, ..., f_n to its nearest structural fragment in g_1, ..., g_k. In this thesis, a polynomial-time approximation scheme (PTAS) is proposed for the problem by using a sampling strategy.

1.1.2 Informatics Aspect: Peptide Subsequence Favors a Partial Subset of Substructures

Reducing the size of the structural fragment library can reduce the search space in fragment-based protein structure prediction. However, it is difficult to make a structural fragment library succinct without losing accuracy. An alternative approach is to consider the fact that a peptide subsequence (sequence segment) does not adopt all the structural fragments in a library with uniform probabilities; that is, each peptide subsequence has a set of preferred structural fragments. Hence, a reduction in the search space can be achieved by using a different set of structural fragments for each sequence segment.

To identify the preferred subset of structural fragments for a sequence segment, classical approaches such as ROSETTA depend on the sequence profile and secondary structure information to predict the structural fragments. In this thesis, more structural information, such as solvent accessibility and contact capacity are utilized for finding the structural fragments. To derive the best combination of sequence profile and structural information items, integer linear programming is employed.

1.2 Sampling Structures from Building Blocks

Given a set of candidate substructures for each sequence segment, the intention is to build a structure which is similar to the native structure. By designing a search method such that the conformation space to be searched is greatly reduced, it might be possible to sample a native-like structure where the conformations are compatible with the derived local biases.

A position specific hidden Markov model (HMM) based on the above idea is applied to predict the protein structures. Inherently, the new framework repeats itself to converge to a final target, conglomerating fragment assembly, clustering, target selection, refinement, and consensus, all in one process.

An implementation of the method results in the FALCON system to predict protein structures. In addition, an automatic system for determining protein structures from NMR spectra is obtained by extending the FALCON sytem.

1.3 Side Chain Packing

After the protein structures are predicted, a natural step is to identify the molecules, called *side chains*, which are attached to the core structure. Again, the problem is addressed from both aspects.

1.3.1 Informatics Aspect: Backbone Codes Sufficient Information for Side Chain Conformation

A typical side chain packing method consists of two steps. First, side chain conformations, known as rotamers, are extracted from known protein structures or rotamer libraries for each residue. Secondly, an energy function is used to eliminate the ambiguities of the side chain conformations. A model of substructures called *hexagon substructures* is used in the search for the rotamer candidates. The hexagon substructures code some distant contact information to implicitly capture some subtle issues in the energy function for the side chain conformations.

The results indicate that the number of rotamer candidates can be reduced to around three. This leads to the following study.

1.3.2 Computational Aspect: The Problem Remains NP-hard for a Constant Number of Rotamers

It has been proven that the side chain packing problem is NP-complete [7, 125]. However, the results do not consider geometric constraints; that is, the number of residues that a rotamer can interact with is constant. The residues can be bounded in a convex shape of a constant volume. Using this constraint, a PTAS has been proposed under the model of geometric neighborhood graph and an NP-complete proof is given [167]. In the proof, each residue is assumed to have m rotamers, for some unbounded constant m. However, this is not valid, as it is possible to reduce the number of rotamer candidates for each residue to a small constant by the dead end elimination preprocessor or by the hexagon structures proposed in this thesis. It is useful to know if, in this case, the problem remains NP-hard. This thesis gives a proof that the problem remains NP-complete under the geometric constraint, even if each residue has at most three rotamers.

1.4 Reporting Final Decoys

Typically, *ab initio* prediction methods are repeated many times, with each run producing one structure. This results in a large collection of structural candidates, which are referred to as decoys. The task of selecting one decoy as the final answer is an interesting problem, which is studied from both the computational and informatics aspects.

1.4.1 Computational Aspect: a Faster Clustering Method

Within the decoy set, the decoy with the highest number of neighbors within a threshold distance is often better than the other decoys in the set. This decoy is sometimes reported as the best decoy within the set. Conventional methods for such a computation are based on pairwise RMSD (Root Mean Squared Deviation) evaluation, resulting in runtimes that are, at best, quadratic in the number of decoys. Three heuristics are proposed to speed up the RMSD evaluation and their effectiveness are examined through experimentation in this thesis.

1.4.2 Informatics Aspect: Distance Function Favorites Native Structure

The quality assessment of decoys is an essential and difficult step in protein structure prediction. It is possible to utilize ensemble-based methods such as clustering techniques, or employ machine learning techniques with the alignment score entries as features. However, these methods are based on an un-verified hypothesis that if two decoys are close to each other, both of them should be close to the unknown native structure. A strict verification of this hypothesis can provide not only an explanation, but also a stochastic foundation for further applications such as model selection and refinement.

The relationship between the distance of any pair of decoys and the distance of these decoys to the native structure is investigated. It is observed that for any two decoys with a fixed distance, their distance to the native structure roughly conforms to a Gaussian distribution. Based on this observation, a novel model selection method is designed to explain why ensemble-based methods work well in practice.

1.5 Model Comparison

How to evaluate the quality of models is a fundamental problem in the field of protein structure prediction. Numerous evaluation criteria have been proposed, and one of the most intuitive criteria is to find a *largest well-predicted subset*, that is, a maximum subset of the model which matches the native structure [137].

In this problem, one is given two equal length structures and the task is to find a superposition of the two structures to maximize the overlapping residues between the two structures. The problem is solvable in $O(n^7)$ time, albeit too slow for practical usage. A $(1+\epsilon)d$ distance approximation algorithm is proposed that runs in time $O(n^3 \log n/\epsilon^5)$ for general protein structures. In the case of globular proteins, the result can be enhanced to a randomized $O(n \log^2 n)$ time algorithm with a probability of at least $1 - O(1/n)$. In addition, a $(1+\epsilon)$-approximation

algorithm is devised to compute the minimum distance to fit all the points of a model to its native structure in time $O(n(\log\log n + \log 1/\epsilon)/\epsilon^5)$.

1.6 Structure Alignment

Structure alignment is a problem very similar to model comparison. In structure alignment, one is given two structures of possibly non-equal lengths and the task is to find out a superposition, as well as a maximum set of corresponding residues of the two structures by the superposition. Three open questions on this topic are answered.

1.6.1 Structure Alignments under the Proximity Requirement

In this version of the problem, each protein is modeled as a sequence of 3D points, and a contact edge is placed between each pair of these points that are sufficiently close. Given two proteins represented this way, the problem is to find a subset of points from each protein, and a bijective matching of points between these two subsets with the objective of maximizing either

- the size of the subsets (largest common point set, the LCP problem), or
- the number of edges that exist simultaneously in both subsets (Contact Map Optimization, the CMO problem)

with the requirement that only the points within a specified proximity can be matched. It is known that the general CMO problem (without the proximity requirement) is hard to approximate. However, with the proximity requirement, it is known that if a minimum inter-residue distance is imposed on the input, approximate solutions can be efficiently obtained. The two questions of the CMO problem under these conditions were open. In this thesis the CMO problem under these conditions is shown to be

- NP-hard, but
- allows a PTAS.

In addition, we show approximation schemes for the LCP problem which improve on the time complexities of known algorithms.

1.6.2 Crossed Contact Map

The contact map pattern problem is to identify a largest pattern from a given contact map with a certain property. The problem left unanswered at present is if there is a polynomial time solution for the contact map pattern problem, when the pattern is *crossed*. We present a reduction from the clique problem to show that the problem is NP-hard.

The contact map pattern matching problem is to decide if a contact map (called the *pattern*) is a substructure of another contact map (called the *target*). In general, the problem is NP-hard, but when there are restrictions on the form of the pattern, the problem can, in some cases, be solved in polynomial time. In particular, a polynomial time algorithm has been proposed [66, 67] when the patterns are so-called *crossing contact maps*. In this thesis, it is confirmed that the problem is actually NP-hard and an error in the analysis of the previous algorithm is noted.

Chapter 2

Preliminary Issues

In this chapter, protein structures and some related concepts in the field of protein structure prediction are introduced. The most important molecules in biochemistry are proteins and many issues are related to protein structure. The review in this chapter is brief. However, it gives a reader sufficient biological background to continue to read this thesis.

2.1 Amino Acids and Primary Sequences

The building blocks of proteins are amino acids. Twenty types of amino acids have been found in protein structures. A typical amino acid has an amine functional group (-NH_2) and a carboxyl functional group (-COOH). In an amino acid, these two groups are attached to a C atom, which is referred to as C_α (see Figure 2.1). In addition, an R group is attached to the C_α in the amino acid, and is referred to as the *side chain* of the amino acid. The amino acids differ mainly by the types of their side chains. An *R* group can be as simple as an H atom, or as complex as to contain aromatic substructure. Twenty formulas for the side chains have been found in protein structures, resulting in twenty types of amino acids.

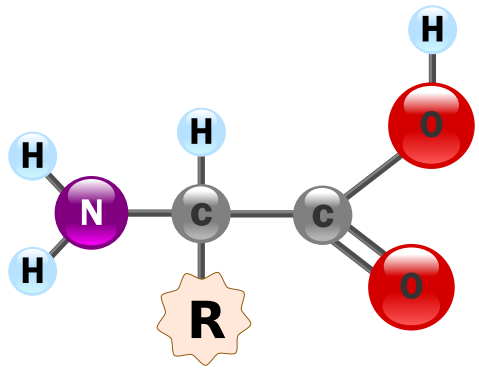


Figure 2.1: Amino acid [161]

Two amino acids can react to form a peptide bond. In particular, the amine group of one amino acid reacts with the other amino acid's carboxyl group to create a new peptide bond, releasing one water molecule in the process (see Figure 2.2). One end of this new molecule is an amine group, and the other end

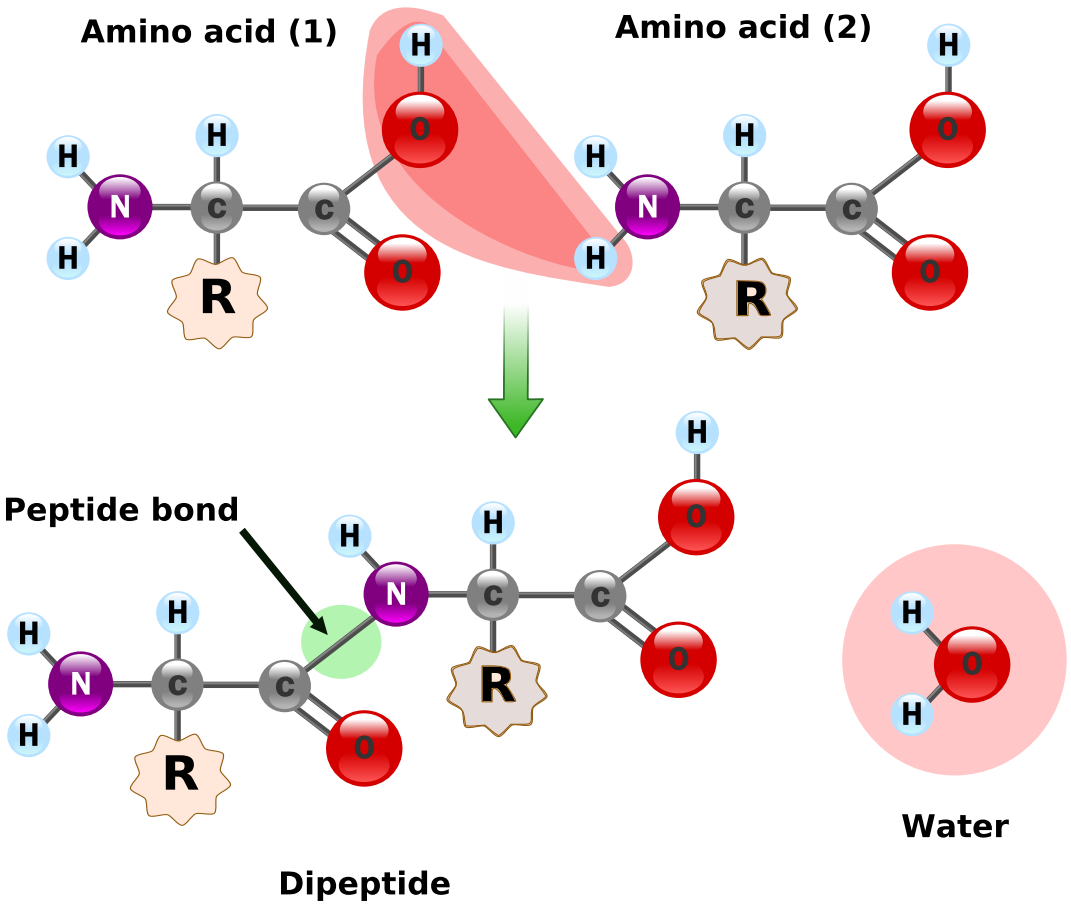


Figure 2.2: Formation of peptide [161]

is a carboxyl group. This enables the molecule to continually react with other amino acids to form a polymerization of amino acids. A protein sequence is such a polymerization of amino acids.

If we write each amino acid type as a single letter, the protein sequence can be written as a string over an alphabet of size twenty. Each amino acid is also coded with three letters according to its name. These are the two common schemes for coding protein sequences.

2.2 Shapes of Proteins

Usually, protein sequences that occur in nature are fewer than 1,000 residues long. Occasionally, there are protein sequences which contain more than 100,000 residues. Often, these long proteins contain some repetitions in their amino acid sequences, and are referred to as *fibrous proteins*. They are inert and play important roles in bones, skin, hair cells, and so on.

Many known protein structures are classified as *globular proteins*. Frequently, they have a unique compact structure which they fold into, and they are, typically, fewer than a hundred to several hundred residues in length. Many globular proteins are crystallisable, and the structures can be determined by X-ray. In addition, NMR techniques have been utilized to determine the structures.

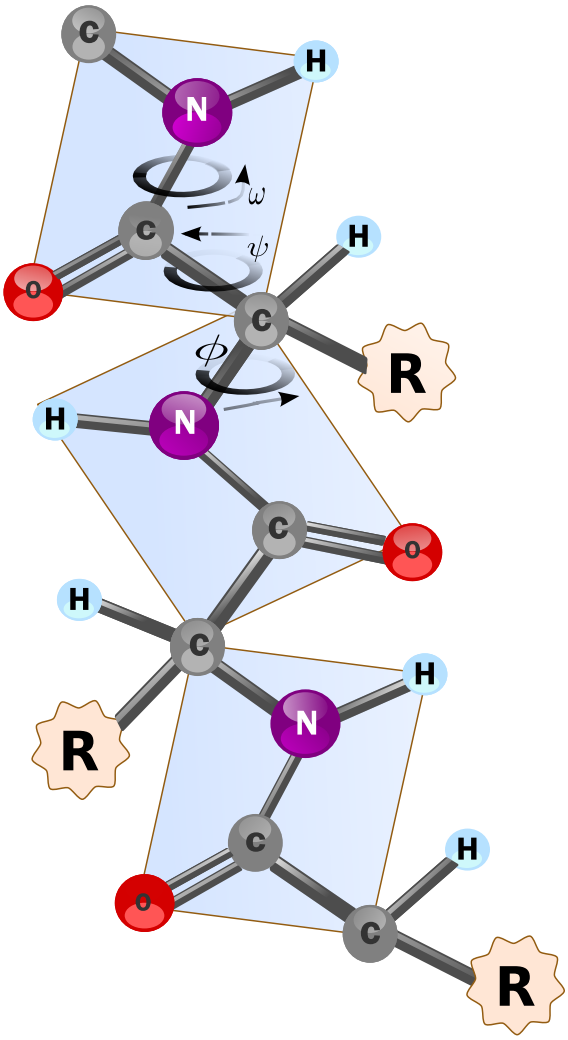


Figure 2.3: Geometry of backbone atoms

Some proteins are also classified as *membrane proteins*. These proteins are attached to or associated with the cells. They exhibit a small tail that is anchored or embedded in the membranes. The membrane proteins with a small tail anchored in the membranes are often also globular. More than half of these proteins interact with the membranes, and can transfer materials through the membranes they interact with.

2.3 Geometric Representations and Some Notations

Typically, protein structures are specified to various levels of detail, as they are needed. A full representation requires at least all the coordinates of the proteins's atom, but since the positions of the hydrogens in the protein are difficult to detect with NMR or X-ray techniques, a protein structure is often specified with only the coordinates of its heavy atoms. For the purpose of protein structure prediction and comparison, only the backbone heavy atoms are commonly specified. For this reason, a protein structure means only the backbone heavy atoms's coordinates in protein structure prediction, usually. Furthermore, in some programs and methods, only the C_αs' coordinates for the protein structure are considered.

Another approach to fully specify a protein structure is by stating all of its bond lengths, bond angles, and dihedral (torsional) angles. The bond lengths and bond angles are commonly considered invariant for the purpose of protein

structure prediction. As a result, the structure can be recovered using only the dihedral angles. In addition, the dihedral angles (referred to as ω) around the peptide bonds are approximately 180°, and occasionally, 0°. Therefore, a backbone is often parameterized by ϕ (dihedral angles around bond C-C_α) and ψ (dihedral angle around bound C_α-N) angles.

In this thesis, the protein structure and C_α's coordinates are referred to interchangeably, when the context is clear. Therefore, a protein structure is a sequence of 3D coordinates. That is, it is possible to denote the C_α trace of a protein structure as s_1, s_2, ..., s_n, $s_i \in \mathbb{R}^3$, $1 \leq i \leq n$ for a protein structure of n residues. The distance between two adjacent C_αs is approximately 3.8$\mathring{A}$ with small variances.

Each C_α atom corresponds to a point in 3D space. For two protein structures $S_1 = (s_{1,1}, s_{1,2}, \ldots, s_{1,n})$ and $S_2 = (s_{2,1}, s_{2,2}, \ldots, s_{2,n})$, each $s_{i,j}$, $1 \leq i \leq 2$, $1 \leq j \leq n$, is a 3D point, indicating a C_α atom in the backbone. The C_α RMSD of S_1 and S_2 is defined as

$$C_\alpha \text{ RMSD}(S_1, S_2) = i \qquad (2.1)$$
$$\min_{R \in \mathcal{R}, T \in \mathcal{T}} \sqrt{\frac{\sum_{i=1}^{n} ||Rs_{1,i} - s_{2,i} - T||^2}{n}}$$

where $\mathcal{R}$ is the set of all rotations, and $\mathcal{T}$, the set of all translations.

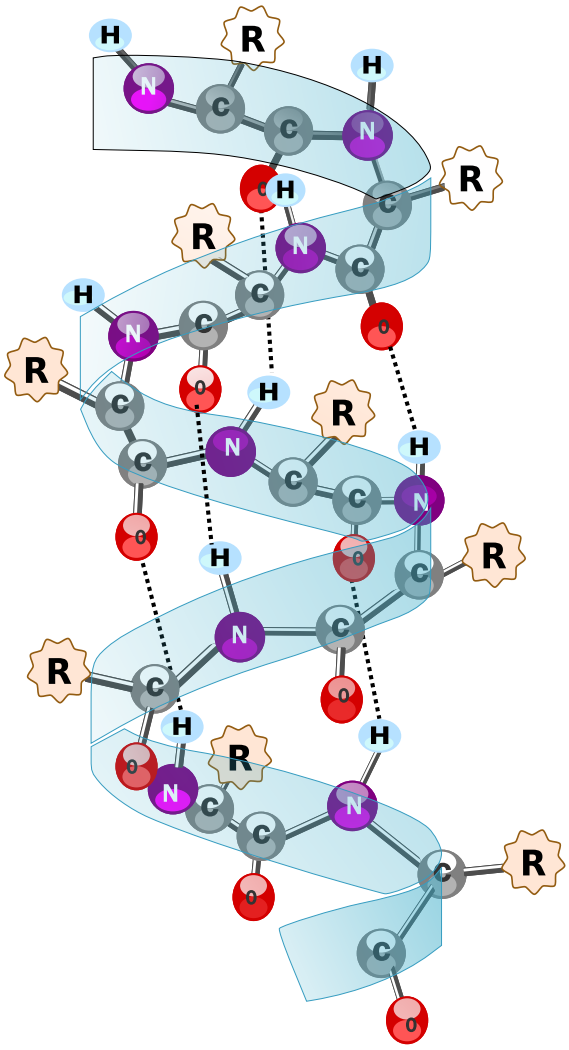


Figure 2.4: Helix structure

Technically speaking, the RMSD between two protein structures should be defined on a full representation of the protein structures. However, most of the present *ab initio* methods depend on a simplified representation of protein structures, which consists of only the C_α atoms. For this reason, the term RMSD is used to refer to the C_α RMSD throughout this thesis.

Torsional angles are used to parameterize the backbone structure, where the sequence of torsional angle pairs is denoted as (ϕ_1, ψ_1), (ϕ_2, ψ_2), ..., (ϕ_n, ψ_n), from the N-terminus to the C-terminus.

2.4 Secondary Structure

Secondary structures are the local conformations or the spatial relationships of the amino acids that reside close to each other in the primary sequences. Three types of secondary structures exist in globular proteins: α-helices, β-strands, and turns.

The most well-known secondary structure is the right-handed α-helix. Each turn of a regular α-helix has 3.6 residues. The values of the dihedral angles for ϕ and ψ in a regular helix are around $-57°$ and $-47°$, respectively. Hydrogen bonds are formed between the carbonyl group of residue i and the amide group of residue $i + 4$. Hydrogen bonds are the dominant reason that helix structures are formed. Helices can contain four residues to 40 residues. The structure is stable at the length of approximately ten residues.

Occasionally, helix structures are observed, where the hydrogen bonds are formed between residue i and residue $i + 3$ (3_{10} helix), or between residue i and residue $i + 5$ (π helix). Most software programs to predict secondary structures do not distinguish between one type of helices and another. Amino acid types do not appear in helix structures with uniform probability. Amino acid P (proline) does not form helices. An α-helix is drawn as a helix cartoon without the atomic details.

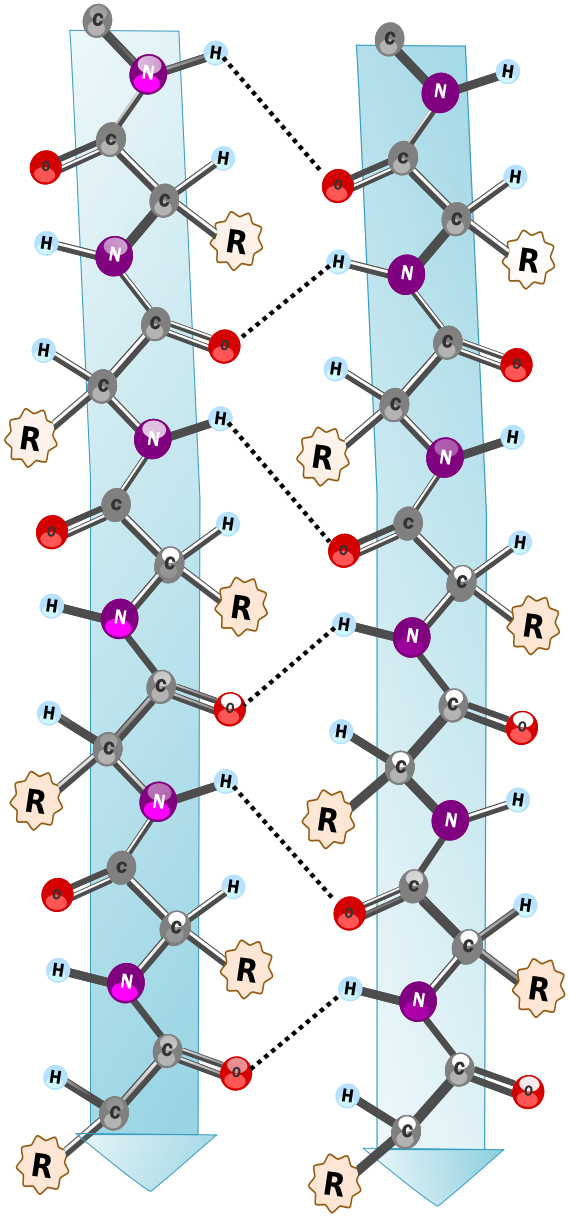


Figure 2.5: Beta strand

Typically, β-strands are the second major secondary structures in proteins and they are chains of five to ten residues. The β-strand is a helix arrangement, and a single β-strand is not stable due to the lack of local contact. If two strands are paired, then hydrogen bonds are formed, and the stability increases significantly. When two or more β-strands form a sheet-like structure, this structure is referred to as a β-sheet. Hence, in a β-sheet, two adjacent strands can be either parallel or anti-parallel and two β-strands are connected by three or more hydrogen bonds. Usually, parallel

β-sheets contain at least four strands, whereas anti-parallel β-sheets usually contain only two strands. β-sheets are often extensively curved and are called β-barrels. The typical β-barrels are formed by eight parallel strands which are linked together by helical segments. The β-strands are seldom stretched perfectly, and are somewhat distorted. The ideal values of the dihedral angles ϕ and ψ are -135° and 135°, respectively. Again, residues do not occur with uniform probability in β-strands, or even at different portions of the strands. Frequently, the β-strands are drawn as an arrow cartoon directed from the N-terminus to the C-terminus without the atomic details.

Turn is the third secondary structure type. In some structures, more than 30% of the residues are turn structures. Frequently, they are not random structures. Turns are formed to redirect the polypeptide. Analyses have indicated that there are at least ten types of turns, each with lengths of three to four amino acids. A γ turn contains three residues, and it is between two adjacent anti-parallel β strands. G (Glycine) is the preferred residue type for the second residue in the turn. The first and the third residues in the γ turn can form a hydrogen bond. A β turn contains four residues, and it is possible for the first and forth residues to form a hydrogen bond.

2.5 Tertiary Structure

The Protein Data Bank (PDB) [17] contains nearly 57,800 structures at this moment, and the number of structures is increasing. These structures are obtained by NMR and X-ray techniques. Recently, Electron Microscopy (EM) has emerged as an alternative tool for determining protein structures experimentally. There are around 200 structures in the database found by using the EM method at this moment. These deposited structures are tertiary structures for some protein sequences.

Tertiary structures are the folded polypeptide chains. The term fold and tertiary structures are used interchangeably. A fold links together the secondary structure elements to form a compact molecule, called a *native structure*. For the purpose of protein structure prediction, it is typically assumed that the native structure is unique for each protein.

There are several sources of interactions in the formation of a tertiary structure: disulfide bridges, hydrophobic effects, charge-charge interactions, hydrogen bonds, and van der Waals interactions. Disulfide bridges are the bonds formed by two cystine animo acids at least five residues apart in the sequence. They break at high temperatures. Some amino acid types are hydrophobic. The hydrophobic effect is that non-polar atoms prefer non-aqueous environments. Charge-charge interactions occur between oppositely charged side chains. Hydrogen bonds form the secondary

structure elements, and can occur between side chains. Van der Waals' interactions are crucial to protein folding, and control the interactions among atoms.

2.6 Additional Remarks

It is assumed that the protein backbone structure can be parameterized by the dihedral angle pair sequence, (ϕ_1, ψ), ..., (ϕ_n, ψ_n), based on the assumption that the bond angles and bond lengths are invariant. However, the bond angles and bond lengths often have small variations, and these variations can result in large changes in the 3D structure.

This issue can be treated as follows. Given a structure S, its torsional pairs, (ϕ_1, ψ), ..., (ϕ_n, ψ_n) can be extracted. With these torsional angles and ideal bond lengths and bond angles, a new backbone structure S' can be constructed. The distance between S' and S is denoted $d(S', S)$. If the distance d is the RMSD, on average, $d(S', S)$ can be as large as $6.5\mathring{A}$ [78]. One possible way to overcome this problem is to distort the torsional angles slightly to absorb the errors introduced by the bond angles and bond lengths. By doing so, the error can be reduced to 0.5Å usually. This problem can be formalized as follows.

Structure Idealization Problem	
Input:	Structure S, backbone dihedral angle pairs (ϕ_1, ψ_1), ..., (ϕ_n, ψ_n) and a constant θ
Output:	A structure S' with (1) dihedral angle pair sequence $(\phi'_1, \psi'_1), ..., (\phi'_n, \psi'_n)$; (2) ideal bond angles and bond lengths; (3) $\lvert\phi_i - \phi'_i\rvert \leq \theta$ and $\lvert\psi_i - \psi'_i\rvert \leq \theta_i$, $1 \leq i \leq n$; and (4) minimized $d(S', S)$

This idealized problem can be extended to all the dihedral angles and all the atoms in a protein structure. Whether this problem can be solved in polynomial time is unknown, and a related problem has been noted in [46].

Chapter 3

Structural Fragment Libraries

A small set of structural fragments suffices to model protein structures accurately [93]. The building of such structural fragment libraries has attracted intensive research. A fragment-based protein structure prediction is done in two steps:

- identify the building blocks which are the fragments of known structures; and
- construct the protein structure with those building blocks by applying some search or simulation algorithms.

The design of succinct and highly accurate structural fragment libraries is crucial to the approach. Compact libraries enable efficient searches, while accurate libraries result in better models. A constant factor reduction in the library size results in an exponential reduction of the search space.

The two best automatic methods in CASP7 [120], namely ROSETTA and TASSER, involve the use of the fragment assembly strategy. Fragment-based protein structure prediction methods can be traced back to [122], in which a protein fold is simplified into smaller parts by using regular secondary structures. Research intensified after the work of [86], which uses the known structures to refine the predicted structures. Various fragment based structure prediction methods have also been developed in the literature [23, 40, 86, 104, 138, 146, 153, 160], the most notable being ROSETTA [24, 37, 141].

Studies on the problem have resulted in very compact independent libraries, and it is difficult to reduce the sizes of these libraries further. However, it has been noted that a sequence segment does not adopt all the structural fragments in a library with uniform probability. Therefore, it is more reasonable to build a customized structural candidate list for each sequence segment. This way, the size of the structural candidate list can be even more succinct.

3.1 Fragment Libraries

Structural fragment libraries are also referred to as "structural alphabet" in the literature. The size of a fragment library varies from dozens to hundreds of structural fragments, and the fragments can have fixed or variable lengths. Typically, the fragments in these libraries have lengths of no more than nine, since the structure database does not contain representative resemblances for longer fragments [56].

Kolodny *et al.* [93] have studied fragment library with k-means clustering methods, and shown that it is unnecessary to have a large fragment library to accurately model protein structures and construct near native structures. In that study, fragments with lengths of four to seven are built with library of size from four to 250. The criteria of evaluating fragment libraries for building protein structures have been investigated in [78]. Besides extracting the structural fragments from known proteins, research has also been conducted on constructing structural fragments with *ab initio* methods, and such methods produce longer fragments [114]. Some recent fragment assembly algorithms [73, 82, 101] rely on longer fragments and/or different simulation algorithms. In another related work, De Brevern *et al.* [4] proposed a 16-letter alphabet to predict the local structures of a protein.

For the purpose of protein structure prediction, it is more desirable to have a position specific structural fragment list for each sequence segment of the target. Only a limited number of structural fragments in the fragment libraries are adopted as candidate structural fragments for a sequence segment. ROSETTA implemented this idea according to two types of information: the sequence profile and the secondary structure similarity [129, 141]. Specifically, sequence profiles, for the query sequence and each sequence in the structure database, are generated by PSI-BLAST [11]. A profile-profile similarity between a query sequence segment and a structural fragment is calculated using a distance function called the City Block Metric (CBM) such that

$$DISTANCE = \sum_{i=1}^{\ell} \sum_{aa=1}^{20} |S(aa, i) - X(aa, i)| \tag{3.1}$$

where ℓ is the fragment length, and $S(aa, i)$ and $X(aa, i)$ are the frequencies of amino acid aa at position i in the sequence segment and in the structural fragment, respectively. In addition, for a query sequence segment, its predicted secondary structure is compared with the known secondary structure of each structural fragment.

In TASSER [171] similar ideas are adopted. However, unlike ROSETTA, where the fragments are a fixed length, TASSER extracts the structural fragments with varying fragment lengths, from the structural models generated by threading programs.

3.2 Contributions

In this chapter, two problems regarding structural fragments are studied. The first problem is concerned with finding succinct structural fragments, while the second problem is on the construction of position specific fragment libraries.

The first problem is as described in Section 1.1.1. It is formalized as a problem of constructing a few consensus structures to represent a given set of structural fragments. It will be shown here that that there is a PTAS for the problem through a sampling strategy. More precisely, an optimal solution, obtained by sampling the smaller subsets of the input suffices to provide an approximate solution, and the approximation ratio improves as the size of the sampled subsets increased.

For the second problem, the following ideas are explored.

- By introducing structural information items, such as the secondary structure, the solvent accessibility, and the contact capacity, more accurate predictions can be achieved.
- By using integer linear programming, the best combinations of both the sequence and structural information items can be derived.

It is noteworthy that these strategies can significantly improve the protein structure prediction, with all the other conditions unchanged.

3.3 Independent Structural Fragment Libraries

From here onwards the discussion proceeds formally. First, the problem is formalized and some relevant concepts are introduced.

3.3.1 Problem Formulation

Throughout this discussion, ℓ is a fixed non-zero natural number. A *structural fragment* is a sequence of ℓ 3D-points. The *mean square distance (MSD)* between two structural fragments, $f = (f[1], \ldots, f[\ell])$ and $g = (g[1], \ldots, g[\ell])$, is defined as

$$\mathrm{MSD}(f, g) = \min_{R \in \mathcal{R}, \tau \in \mathcal{T}} \sum_{i=1}^{\ell} \parallel f[i] - (R \cdot g[i] + \tau) \parallel^2, \tag{3.2}$$

where $\mathcal{R}$ is the set of all rotation matrices, $\mathcal{T}$ is the set of all the translation vectors, and $\parallel x - y \parallel$ is the Euclidean distance between x and y.

The root of the MSD measure, that is, the RMSD, has been extensively studied. $R \in \mathcal{R}$, $\tau \in \mathcal{T}$ that minimizes $\sum_{i=1}^{\ell} \parallel f[i] - (R \cdot g[i] + \tau) \parallel^2$ to yield $\mathrm{MSD}(f, g)$

also gives RMSD(f, g), and vice versa. Since, given any f and g, there are equations [13, 152] for finding R and τ that yield RMSD(f, g), MSD(f, g) can be computed efficiently for any f and g.

Furthermore, it is known that to minimize $\sum_{i=1}^{\ell} \parallel f[i] - (R \cdot g[i] + \tau) \parallel^2$, the centroids of f and of g must coincide [13]. Due to this, without the loss of generality, it is assumed that all the structural fragments have centroids at the origin. Such transformations can be accomplished in $O(n\ell)$ time. After such transformations, in computing MSD(f, g), only the parameter $R \in \mathcal{R}$ needs to be considered, that is,

$$\text{MSD}(f, g) = \min_{R \in \mathcal{R}} \sum_{i=1}^{\ell} \parallel f[i] - R \cdot g[i] \parallel^2 . \tag{3.3}$$

In the problem, given a set of n structural fragments $f_1, f_2, \ldots, f_n$, the task is to find k structural fragments, $g_1, \ldots, g_k$, such that each structural fragment, f_i, is "near", in terms of the MSD, to at least one of the structural fragments in $g_1, \ldots, g_k$. Such a problem is formulated as

k-Consensus Structural Fragments Problem under MSD	
Input:	n structural fragments $f_1, \ldots f_n$, and a non-zero natural number $k < n$.
Output:	k structural fragments $g_1, \ldots g_k$, minimizing the cost $\sum_{i=1}^{n} \min_{1 \le j \le k} \text{MSD}(f_i, g_j)$.

In this chapter it is demonstrated that there is a PTAS for the problem.

The following notations are employed. The cardinality of set A is written as $|A|$. For a set A and a non-zero natural number n, A^n denotes the set of all length n sequences of elements of A. If the elements in set A are indexed, say $A = \{f_1, f_2, \ldots, f_n\}$, then $A^{m!}$ represents the set of all the length m sequences, $f_{i_1}, f_{i_2}, \ldots, f_{i_m}$, where $1 \le i_1 \le i_2 \le \ldots \le i_m \le n$. For sequence S, $S(i)$ signifies the i-th element in S, and $|S|$ indicates its length.

3.3.2 Core Set of Structural Fragments

The following lemma, from [127], is central to the PTAS in this thesis.

Lemma 1 ([127]). *Given a sequence of real numbers $a_1, a_2, \ldots, a_n$ and an integer r, $1 \le r \le n$. Then, the following equation holds:*

$$\frac{1}{n^r} \sum_{1 \le i_1, i_2, \ldots, i_r \le n} \sum_{i=1}^{n} (\frac{1}{r} \sum_{k=1}^{r} a_{i_k} - a_i)^2 = \frac{r+1}{r} \sum_{i=1}^{n} (\frac{1}{n} \sum_{k=1}^{n} a_n - a_i)^2. \tag{3.4}$$

Given a sequence of 3D points, $P_1 = (x_1, y_1, z_1), P_2 = (x_2, y_2, z_2), \ldots, P_n = (x_n, y_n, z_n)$, it can be derived that

$$\begin{aligned}
& \frac{1}{n^r} \sum_{1 \le i_1, i_2, \ldots, i_r \le n} \sum_{i=1}^{n} \| \frac{P_{i_1} + P_{i_2} + \cdots + P_{i_r}}{r} - P_i \|^2 \\
= & \frac{1}{n^r} \sum_{1 \le i_1, \ldots, i_r \le n} \sum_{i=1}^{n} (\frac{1}{r} \sum_{k=1}^{r} x_{i_k} - x_i)^2 + (\frac{1}{r} \sum_{k=1}^{r} y_{i_k} - y_i)^2 + (\frac{1}{r} \sum_{k=1}^{r} z_{i_k} - z_i)^2 \\
= & \frac{r+1}{r} \sum_{i=1}^{n} (\frac{x_1 + \ldots + x_n}{n} - x_i)^2 + (\frac{y_1 + \ldots + z_n}{n} - z_i)^2 + (\frac{z_1 + \ldots + z_n}{n} - z_i)^2 \\
= & \frac{r+1}{r} \sum_{i=1}^{n} \| \frac{P_1 + P_2 + \cdots + P_n}{n} - P_i \|^2 . \qquad (3.5)
\end{aligned}$$

Similarly, the equation can be extended for the structural fragments. Given n structural fragments, $f_1, \ldots, f_n$, Equation 3.5 can be rewritten as

$$\frac{1}{n^r} \sum_{1 \le i_1, \ldots, i_r \le n} \sum_{i=1}^{n} \| \frac{f_{i_1} + \cdots + f_{i_r}}{r} - f_i \|^2 = \frac{r+1}{r} \sum_{i=1}^{n} \| \frac{f_1 + \cdots + f_n}{n} - f_i \|^2 \quad (3.6)$$

It can be inferred from the equation that there exists a sequence of r structural fragments $f_{i_1}, f_{i_2}, \ldots, f_{i_r}$ such that

$$\sum_{i=1}^{n} \| \frac{f_{i_1} + \cdots + f_{i_r}}{r} - f_i \|^2 \le \frac{r+1}{r} \sum_{i=1}^{n} \| \frac{f_1 + \cdots + f_n}{n} - f_i \|^2 . \quad (3.7)$$

The new strategy relies on this fact, essentially, in the same way as in [127] to approximate the optimal solution for the k-consensus structural fragments problem; that is, by exhaustively sampling every combination of k sequences, each of the r elements from the space, $\mathcal{R}' \times \{f_1, \ldots, f_n\}$, where $f_1, \ldots, f_n$ is the input, and $\mathcal{R}'$ is a fixed selected set of rotations. This set of rotations is discussed next.

3.3.3 Discretized Rotation Space

Any rotation can be represented by a normalized vector, u, and a rotation angle, θ, where u is the axis around which an object is rotated by θ. If (u, θ) is applied to vector v, vector $\hat{v}$ is obtained, and expressed as

$$\hat{v} = u(v \cdot u) + (v - w(v \cdot w)) \cos\theta + (v \times w) \sin\theta, \quad (3.8)$$

where $\cdot$ represents dot the product, and $\times$ represents the cross product.

By the equation, it is possible to verify that a change of ϵ in u results in a change of, at most, $\alpha_1 \epsilon |v|$ in $|\hat{v}|$ for some computable $\alpha_1 \in \mathbb{R}$. Also, a change of

ϵ in θ results in a change of, at most, $\alpha_2\epsilon|v|$ in $|\hat{v}|$ for some computable $\alpha_2 \in \mathbb{R}$. Now, any rotation, around an axis through the origin, can be written in the form, $(\theta_1, \theta_2, \theta_3)$, where $\theta_1, \theta_2, \theta_3 \in [0, 2\pi)$ are, respectively, a rotation around each of the x, y, z axes. Similarly, a change of ϵ in θ_1, θ_2, or θ_3 results in a change of, at most, $\alpha\epsilon|v|$, for some computable $\alpha \in \mathbb{R}$.

The values that each θ_i, $1 \leq i \leq 3$ may take within the range $[0, 2\pi)$ are discretized into a series of angles at the angular difference, ϑ. Thus, there are at most, $O(1/\vartheta)$ of such values for each θ_i, $1 \leq i \leq 3$. $\mathcal{R}'$ denotes the set of all possible discretized rotations $(\theta_1, \theta_2, \theta_3)$. $|\mathcal{R}'|$ is of the order $O(1/\vartheta^3)$.

If $\boldsymbol{d}$ is the diameter of the smallest ball that is able to encapsulate each of $f_1, f_2, \ldots, f_n$, any distance between two points among $f_1, \ldots, f_n$ is, at most, $\boldsymbol{d}$. In this chapter, it is assumed $\boldsymbol{d}$ is a constant with respect to the input size. For a protein structure, $\boldsymbol{d}$ is of the order $O(\ell)$ [72]. For any $b \in \mathbb{R}$, ϑ can be chosen so small that for any rotation, R, and any point, $p \in \mathbb{R}^3$, there exists $R' \in \mathcal{R}'$ such that $\parallel R \cdot p - R' \cdot p \parallel \leq \alpha\vartheta\boldsymbol{d} \leq b$.

3.3.4 Polynomial-time Algorithm with Ratio $((1+\epsilon)D_{opt}+c)$

The newly proposed algorithm for the k-consensus structural fragments problem is summarized in Table 3.1.

The following explains the details of the algorithm. In Step ii, each combination of m distinct subsets $A_1, \ldots, A_m$ from $f_1, \ldots, f_n$ is evaluated. Since all possible such subsets are examined, it is clear that there is at least one combination such that each subset is from a distinct cluster in the optimal clustering. The score of each subset A_j is evaluated by sampling up to r structural fragments (allowing repeats) from A_j (from Step ii(a) onwards). Such an evaluation is possible due to Equation 3.7. The evaluation exhaustively checks every possible combination of all the transformations in $\mathcal{R}'$ on the structural fragments. This is attempted in Step ii(b). Each of these samplings of A_j produces a consensus structural fragment, u_j for A_j in Step ii(c), the score of which is evaluated by Step ii(d). Finally, in Step iii, the consensus patterns, $u_1, \ldots, u_m$, which give the best score are the output.

The time complexity of the algorithm should now be analyzed. The number of combinations for $F_1, F_2, \ldots, F_m$ in Step ii(a) is $O(n^{rk})$. An F_j can be represented by a length-r string of $n+1$ symbols, where n of the symbols each represents one of $f_1, \ldots, f_n$, while the remaining symbol represents "nothing". It is clear that for any given A_j, each $F_j \in A_j^{r!}$, or $F_j \in A_j^{|A_j|!}$ (where $|A_j| \leq r$) can be represented by one such string. Furthermore, any $F_1, F_2, \ldots, F_m$ can be entirely represented by k such strings. That is, to represent the case where $m < k$, $k - m$ strings are set to "nothing". From this, it is evident that there are at most $(n+1)^{rk} = O(n^{rk})$ possible combinations of $F_1, F_2, \ldots, F_m$.

For each of these combinations, there are $|\mathcal{R}'|^{rk}$ possible combinations of Θ_1, Θ_2, $\ldots$, Θ_m at Step ii(b), hence resulting in $O((n|\mathcal{R}'|)^{rk})$ iterations to run for Step

ii(c) to Step ii(e). Since Step ii(c) can be done in $O(rk\ell)$, ii(d) in $O(nk|\mathcal{R}'|\ell)$, and ii(e) in $O(n)$ time, the algorithm completes in $O(k\ell(r+n|\mathcal{R}'|)(n|\mathcal{R}'|)^{rk})$ time.

Here, it can be argued that D_{min} eventually is at most $(r+1)/r$ of the optimal solution plus a factor. Let the disjoint clusters in the optimal solution be $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_m \subseteq \{f_1, \ldots, f_n\}$ where $m \leq k$. For each $\mathbf{A}_j$, $1 \leq j \leq m$, $\boldsymbol{u}_j$ denotes a structural fragment which minimizes $\sum_{f \in \mathbf{A}_j} \mathrm{MSD}(\boldsymbol{u}_j, f)$. Furthermore, for each $f \in \mathbf{A}_j$, $\boldsymbol{R}_f$ denotes a rotation where

$$\boldsymbol{R}_f \in \arg\min_{R \in \mathcal{R}} \parallel \boldsymbol{u}_j - R \cdot f \parallel^2 \tag{3.9}$$

and $\mathbf{D}_j$ is defined as

$$\mathbf{D}_j = \sum_{f \in \mathbf{A}_j} \parallel \boldsymbol{u}_j - \boldsymbol{R}_f \cdot f \parallel^2 \quad \text{(hence, the optimal cost, } \mathbf{D} = \textstyle\sum_{j=1}^{m} \mathbf{D}_j\text{)}. \tag{3.10}$$

By the property of the MSD measure, it can be shown that $\boldsymbol{u}_j$ is the average of $\{\boldsymbol{R}_f \cdot f \mid f \in \mathbf{A}_j\}$. For each $\mathbf{A}_j$ where $|\mathbf{A}_j| > r$, by Equation 3.6,

$$\frac{1}{|\mathbf{A}_j|^r} \sum_{F_j \in \mathbf{A}_j^r} \sum_{f \in \mathbf{A}_j} \parallel \frac{\sum_{k=1}^{r} \boldsymbol{R}_{F_j(k)} \cdot F_j(k)}{r} - \boldsymbol{R}_f \cdot f \parallel^2 \;=\; \frac{r+1}{r}\, \mathbf{D}_j. \tag{3.11}$$

Approximation Algorithm k-Consensus Structural Fragments

Input: structural fragments $f_1, \ldots f_n$, natural numbers $k < n$ and $r \geq 1$.
Output: up to k structural fragments $u_1, \ldots u_m$, $m \leq k$.

I. Let $D_{min} = \infty$, Consensus$= \emptyset$.
II. For each possible set of $m \leq k$ disjoint sets $A_1, \ldots, A_m \subseteq \{f_1, \ldots, f_n\}$,
II(a). For every possible set $F_1, F_2, \ldots, F_m$ of sequences where
$F_j \in A_j^{r!}$ if $|A_j| > r$, otherwise
F_j is the (unique) sequence in $A_j^{|A_j|!}$ that contains all the elements of A_j,
(Note that each distinct set of F_1, …, F_m needs to be considered only once.)
II(b). For each possible sequence $\Theta_1, \Theta_2, \ldots, \Theta_m$, where $\Theta_j \in \mathcal{R}'^{|F_j|}$ for $1 \leq j \leq m$,
II(c). For $j = 1$ to m, find u_j, the average structural fragment
for $\Theta_j(1) \cdot F_j(1)$,
$\Theta_j(2) \cdot F_j(2)$,
$\vdots$
$\Theta_j(|F_j|) \cdot F_j(|F_j|)$.
II(d). For $i = 1$ to n, find $d_i = \min\{\parallel u_j - R \cdot f_i \parallel^2 \mid 1 \leq j \leq m, R \in \mathcal{R}'\}$.
II(e). If $\sum_{i=1}^{n} d_i < D_{min}$,
set D_{min} to $\sum_j d_j$ and set Consensus to $\{u_1, \ldots, u_m\}$.
III. Output Consensus.

Table 3.1: PTAS for k-Consensus Structural Fragments.

For each such $\mathbf{A}_j$, let $\boldsymbol{F}_j$ be a member of $\mathbf{A}_j^r$ where

$$\sum_{f \in \mathbf{A}_j} \| \frac{\boldsymbol{R}_{\boldsymbol{F}_j(1)} \cdot \boldsymbol{F}_j(1) + \cdots + \boldsymbol{R}_{\boldsymbol{F}_j(r)} \cdot \boldsymbol{F}_j(r)}{r} - \boldsymbol{R}_f \cdot f \|^2 \;\; \leq \;\; \frac{r+1}{r} \mathbf{D}_j. \quad (3.12)$$

Without loss of generality it is assumed that each $\boldsymbol{F}_j \in \mathbf{A}_j^{r!}$. Define $\boldsymbol{\mu}_j$ as

$$\boldsymbol{\mu}_j = \begin{cases} \frac{\boldsymbol{R}_{\boldsymbol{F}_j(1)} \cdot \boldsymbol{F}_j(1) + \cdots + \boldsymbol{R}_{\boldsymbol{F}_j(r)} \cdot \boldsymbol{F}_j(r)}{} & \text{if } |\mathbf{A}_j| > r \\ \frac{\boldsymbol{R}_{\boldsymbol{F}_j(1)} \cdot \boldsymbol{F}_j(1) + \overset{r}{\cdots} + \boldsymbol{R}_{\boldsymbol{F}_j(|\mathbf{A}_j|)} \cdot \boldsymbol{F}_j(|\mathbf{A}_j|)}{|\mathbf{A}_j|} & \text{, otherwise.} \end{cases} \quad (3.13)$$

Then the following can be derived,

$$\sum_{j=1}^{m} \sum_{f \in \mathbf{A}_j} \| \boldsymbol{\mu}_j - \boldsymbol{R}_f \cdot f \|^2 \;\; \leq \;\; \frac{r+1}{r} \mathbf{D}. \quad (3.14)$$

For each rotation $\boldsymbol{R}_f$, R_f is the closest rotation to $\boldsymbol{R}_f$ within $\mathcal{R}'$. Also, define μ_j as

$$\mu_j = \begin{cases} \frac{R_{\boldsymbol{F}_j(1)} \cdot \boldsymbol{F}_j(1) + \cdots + R_{\boldsymbol{F}_j(r)} \cdot \boldsymbol{F}_j(r)}{} & \text{if } |\mathbf{A}_j| > r \\ \frac{R_{\boldsymbol{F}_j(1)} \cdot \boldsymbol{F}_j(1) + \overset{r}{\cdots} + R_{\boldsymbol{F}_j(|\mathbf{A}_j|)} \cdot \boldsymbol{F}_j(|\mathbf{A}_j|)}{|\mathbf{A}_j|} & \text{, otherwise.} \end{cases} \quad (3.15)$$

Since all possible $F_j \in A_j^{r!}$ for all possible A_j and for all $R \in \mathcal{R}'$ are exhaustively sampled, it is clear that

$$D_{min} \;\; \leq \;\; \sum_{j=1}^{m} \sum_{f \in \mathbf{A}_j} \| \mu_j - R_f \cdot f \|^2 . \quad (3.16)$$

Now the LHS (left hand side) of Equation 3.14 is related the RHS (right hand side) of Equation 3.16. The RHS of Equation 3.16 is

$$\begin{aligned} & \sum_{j=1}^{m} \sum_{f \in \mathbf{A}_j} \| \mu_j - R_f \cdot f \|^2 \\ = \;\; & \sum_{j=1}^{m} \sum_{f \in \mathbf{A}_j} \| \mu_j + (\boldsymbol{\mu}_j - \boldsymbol{\mu}_j) + (\boldsymbol{R}_f \cdot f - \boldsymbol{R}_f \cdot f) - R_f \cdot f \|^2 \\ \leq \;\; & \sum_{j=1}^{m} \sum_{f \in \mathbf{A}_j} (\| \boldsymbol{\mu}_j - \boldsymbol{R}_f \cdot f \| + (\| \mu_j - \boldsymbol{\mu}_j \| + \| \boldsymbol{R}_f \cdot f - R_f \cdot f \|))^2 \\ = \;\; & \sum_{j=1}^{m} \sum_{f \in \mathbf{A}_j} \| \boldsymbol{\mu}_j - \boldsymbol{R}_f \cdot f \|^2 + (\| \mu_j - \boldsymbol{\mu}_j \| + \| \boldsymbol{R}_f \cdot f - R_f \cdot f \|)^2 \\ & \qquad + 2 \| \boldsymbol{\mu}_j - \boldsymbol{R}_f \cdot f \| (\| \mu_j - \boldsymbol{\mu}_j \| + \| \boldsymbol{R}_f \cdot f - R_f \cdot f \|) \\ \leq \;\; & \sum_{j=1}^{m} \sum_{f \in \mathbf{A}_j} \| \boldsymbol{\mu}_j - \boldsymbol{R}_f \cdot f \|^2 + 8n\ell b. \end{aligned} \quad (3.17)$$

Hence, by Equation 3.14, D_{min} is, at most, $(r+1)/r = 1 + 1/r$ of the optimal solution plus a factor, $c = 8n\ell b$. Define $\epsilon = 1/r$,

Theorem 2. *For any $c, \epsilon \in \mathbb{R}$, a $((1+\epsilon)D_{opt} + c)$-approximation solution for the k-consensus structural fragments problem can be computed in*

$$O(k\ell(\frac{1}{\epsilon} + n|\mathcal{R}'|)(n|\mathcal{R}'|)^{\frac{k}{\epsilon}})$$

time.

The factor c in Theorem 2 is due to errors introduced by the use of discretization in the rotations. If it is possible to estimate a lower bound of D_{opt}, this error can be scaled by refining the discretization such that c is an arbitrarily small factor of D_{opt}. To do so, in the next section, a lower bound to D_{opt} is found.

3.3.5 Polynomial-time 4-approximation Algorithm

In this section, a 4-approximation algorithm for the k-consensus structural fragment problem is proposed. The case for $k = 1$ is investigated, and then is generalized for all $k \geq 2$.

The input n structural fragments be f_1, f_2, ..., f_n. Let f_a, $1 \leq a \leq n$ be the structural fragment such that

$$\sum_{1 \leq j \leq n \wedge j \neq a} \mathrm{MSD}(f_a, f_j)$$

is minimized. Note that f_a can be found in time $O(n^2\ell)$, since for any $1 \leq i, j \leq n$, $\mathrm{MSD}(f_i, f_j)$ (more precisely, $\mathrm{RMSD}(f_i, f_j)$) can be computed in time $O(\ell)$ by using equations from [152].

It can be argued that f_a is a 4-approximation. Denote the optimal structural fragment as f_{opt} and the corresponding distance as D_{opt}. For each b, $1 \leq b \leq n$, f_b denotes the fragment where $\mathrm{MSD}(f_b, f_{opt})$ is minimized.

Firstly, the cost of using f_a as the solution, $\sum_{i \neq a} \mathrm{MSD}(f_a, f_i)$, is no greater than $\sum_{i \neq b} \mathrm{MSD}(f_b, f_i)$ for any b, $1 \leq b \leq n$. Furthermore, the following can be established.

Claim 1. *$MSD(f, f') \leq 2(MSD(f, f'') + MSD(f'', f'))$.*

Proof. According to [22],

$$\mathrm{RMSD}(f, f') \leq \mathrm{RMSD}(f, f'') + \mathrm{RMSD}(f'', f'). \tag{3.18}$$

By squaring both sides,

$$\mathrm{MSD}(f, f') \leq \mathrm{MSD}(f, f'') + \mathrm{MSD}(f'', f') + 2\mathrm{RMSD}(f, f'')\mathrm{RMSD}(f'', f'). \tag{3.19}$$

Since

$$2\text{RMSD}(f, f'')\text{RMSD}(f'', f') \leq \text{MSD}(f, f'') + \text{MSD}(f'', f'), \tag{3.20}$$

$\text{MSD}(f, f') \leq 2(\text{MSD}(f, f'') + \text{MSD}(f'', f'))$. □

By such a claim,

$$\begin{aligned}
\sum_{i \neq b} \text{MSD}(f_b, f_i) &\leq 2 \sum_{i \neq b} (\text{MSD}(f_b, f_{opt}) + \text{MSD}(f_{opt}, f_i)) \\
&= 2 \sum_{i \neq b} \text{MSD}(f_b, f_{opt}) + 2 \sum_{i \neq b} \text{MSD}(f_i, f_{opt}) \\
&\leq 2 \sum_{i \neq b} \text{MSD}(f_b, f_{opt}) + 2D_{opt} \qquad (3.21) \\
&\leq 2 \sum_{j \neq b} \text{MSD}(f_j, f_{opt}) + 2D_{opt} \\
&\leq 2D_{opt} + 2D_{opt} = 4D_{opt}.
\end{aligned}$$

Therefore, $\sum_{i \neq a} \text{MSD}(f_a, f_i) \leq 4D_{opt}$. This result can be extended to k structural fragments.

4-Approximation Algorithm k-Consensus Structural Fragments
Input: structural fragments $S = \{f_1, \ldots f_n\}$, natural number $k < n$.
Output: up to k structural fragments A

i. For every set $A \subseteq S$ of up to k structural fragments, do
ii. Compute $\text{cost}(A) = \sum_{f \in S - A} \min_{f' \in A} \text{MSD}(f, f')$
iii. Output A with the least $\text{cost}(A)$.

$\text{MSD}(f, f')$ is computed for each pair of $f, f' \in S$, which takes time $O(n^2\ell)$. Then, at Step i, there are, at most, $O(n^k)$ combinations of A, each of which takes $O(nk)$ time to compute as in Step ii. Consequently, the computation is performed in $O(n^2\ell + kn^{k+1})$ time. To verify that the solution has a 4-approximation factor, let $S_1, S_2, \ldots, S_m$, where $m \leq k$ denote an optimal clustering. Then, according to the earlier argument, there exists $f_{i_1} \in S_1$, $f_{i_2} \in S_2$, ..., and $f_{i_m} \in S_m$ such that each f_{i_x} has a 4-approximation factor for S_x. Thus, $f_{i_1}, f_{i_2}, \ldots, f_{i_m}$ is a 4-approximation algorithm for the k-consensus structural fragment problem. Since the algorithm is applied to exhaustively search for each combination of up to k fragments, the algorithm yields a solution at least as good as $f_{i_1}, f_{i_2} \ldots, f_{i_m}$, and hence, is a 4-approximation algorithm.

Theorem 3. *A 4-approximation solution for the k-consensus structural fragments problem can be computed in $O(n^2\ell + kn^{k+1})$ time.*

3.3.6 $(1+\epsilon)$ Polynomial-time Approximation Scheme

The algorithm in Section 3.3.4 has cost $D \leq (1+\epsilon)D_{opt} + 8n\ell b$, where $b = \alpha\vartheta\boldsymbol{d}$. From Section 3.3.5 there is a lower bound $\boldsymbol{D}_{opt}$ of D_{opt}. Our objective is to achieve $8n\ell b \leq \epsilon\boldsymbol{D}_{opt} \leq \epsilon D_{opt}$. To do so, it suffices to choose $\vartheta \leq \epsilon\boldsymbol{D}_{opt}/(8n\ell\alpha\boldsymbol{d})$. This results in an $|\mathcal{R}'|$ of order $O(1/\vartheta^3) = O((n\ell\boldsymbol{d})^3)$. By substituting $|\mathcal{R}'|$ in Theorem 2, and combining with Theorem 3, the following is attained.

Theorem 4. *For any $\epsilon \in \mathbb{R}$, a $((1+\epsilon)D_{opt})$-approximation solution for the k-consensus structural fragments problem can be computed in time*

$$O(n^2\ell + kn^{k+1} + k\ell(\frac{2}{\epsilon} + n\lambda)(n\lambda)^{\frac{2k}{\epsilon}})$$

, where $\lambda = (n\ell\boldsymbol{d})^3$.

3.4 Position Specific Fragment Libraries

As mentioned in Section 3.2, to construct a fragment library, an approach based on integer programming is proposed. This method is implemented in a program called FRazor ("F" stands for fragment).

3.4.1 Problem Statement

The objective here is to obtain a set of structural fragments for each sequence segment. As candidates for these structural fragments, a collection of fragments are first generated, by parsing the known protein structure database.

Given a protein target sequence t of length n, t is parsed into a collection of sequence segments. A sliding window of fixed length ℓ and step size 1 is employed to parse t, and these segments are $qe^1, qe^2, \ldots, qe^p$, $p = n - \ell + 1$ (qe stands for *query sequence element*), and the *native structural* fragments of these segments are denoted as $ns^1, ns^2, \ldots, ns^p$.

Denote this collection of structural fragments as

$$\mathcal{S} = \{se^1, se^2, \ldots, se^q\}$$

This collection of structural fragments is referred to as the *structural space* (se stands for *structural element*).

The next task is to select some structural fragments for each sequence segment such that the selections contain adequate structural fragments, close to the native structure of the sequence segment. Intuitively, the more native-like the structural fragments, the better are the decoys that can be constructed.

Stated formally, given qe^j, $1 \leq j \leq p$, integer k and k', $k' \leq k$, and a distance threshold θ, it is desirable to select a set of structural fragments, denoted as $\mathbb{S}_j \subset \mathcal{S}$, in accordance with the following conditions.

- $|\mathbb{S}_j| = k$,
- $\exists F_j \subset \mathbb{S}_j$ with $|F_j| \geq k'$, and
- $\forall s \in F_j$, $dist(s, ns^j) \leq \theta$. $dist$ is a given distance function.

F_j is referred to as a subset of *near native* structures for qe^j. If F_j is nonempty, qe^j is *covered* by $\mathbb{S}_j$. $\mathbb{S}_j$ is referred to as the *structural candidate list*, or simply, the *candidate list*, of qe^j.

Structural Distance Criteria

To compute the distance between structural fragments, the RMSD is used. The measure satisfies the triangle inequality for fragments of equal length. While only the RMSD is considered here, the proposed method should be applicable to other distance measures.

Structural Space

The structural space $\mathcal{S}$ used in this study is obtained by parsing the 40 proteins listed in Table 3.2 Part (A), selected from the PDB. It is observed that the set covers 99% of the sequence/structural segments from the CASP7 proteins.

An alternative approach to the structural space is to use an existing library such as Kolodny's Fragment Library [93]. However, the structural space adopted should make little difference as long as it is ensured that any structural fragment is represented by at least one resembling fragment in the structural space. It is noteworthy that ROSETTA and TASSER's approaches to structural space are also based on direct selections from PDB [17].

3.4.2 Generalized Linear Model

It is reasonable to assume that introducing more structural information can help structural fragment prediction. However, how to combine these information remains a difficulty. In this section, an integer linear programming model is proposed to integrate both the sequence and structural information items optimally.

Between each structural fragment se^i in the structural space and each sequence segment qe^j, a feature vector, $\mathcal{V}^{i,j} = \langle v_1^{i,j}, \ldots, v_d^{i,j} \rangle$, of length $d = 4 \times 9$, is computed to measure the quality that se^i and qe^j match for $1 \leq i \leq q$ and $1 \leq j \leq p$. Each entry in $\mathcal{V}^{i,j}$ is a linear or a nonlinear scoring function. $\mathcal{V}^{i,j}$ is labeled as $+1$ if $dist(se^i, ns^j) \leq \theta$, and -1, otherwise.

Traditional machine learning approaches, such as Support Vector Machines (SVM) can be readily employed to classify $\mathcal{V}^{i,j}$ into two classes:

- class +1 contains the feature vectors labeled with +1, and
- class −1 contains the feature vectors labeled with −1.

Then, the set of all the se^i where $\mathcal{V}^{i,j}$ is labeled +1 is treated as the candidate list for qe^j. However, such an approach is too simplistic for this investigation, since not all the structural elements are required to be classified correctly. More precisely, it is fine even if most of the structural elements are classified incorrectly, as long as at least one of the structural fragments for qe^j are classified correctly. It is often the case that a large number of native-like structural fragments exist for a particular sequence segment. However, it suffices here that a candidate list of size say, $k = 25$, is selected, where one of them is native-like. Furthermore, k is a constant much smaller than the total number of native like structural fragments. Hence, most of the structural elements do not need to be classified correctly. While it is possible to design an SVM to classify subsets of k elements with at least one correct element, the approach would significantly increase the learning dimension, requiring more data. Furthermore, since the features are for individual elements, the use of SVM would be more involved than other approaches that are more customized to the problem.

In contrast, the classification task can be easily modeled by a linear model, since a candidate list is separable by a hyperplane with a high probability. To see this, the feature vectors are treated as high dimensional points. For a set of random points, where each point is labeled with +1 or −1, a subset of the point set is to be identified, such that the subset contains at least one point with the label +1. To achieve this, the smallest convex hull containing all points is first formed. For each vertex labeled with +1 of the convex hull, a hyperplane is then used to separate the vertex from the rest. The probability that no vertex on the hull is labeled as +1, can be estimated as $1-(1-P)^{|H|}$, where P is the probability that a point is in class +1, and $|H|$ is the expected number of points on the convex hull. According to [143], $P \approx (1/1.6)^9 = 0.015$ for the fragments of length nine. According to [54], it can be assumed that $|H|$ is sufficiently large to make $1-(1-P)^{|H|}$ close to 1. Thus, with a high probability, a linear separator can be trivially obtained.

In this thesis, these two observations give us the insight to design a system of linear models for solving the structural fragment selection problem.

Generalized Linear Model Formulation

A general linear model has the following form:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_k \phi_k(\mathbf{x}), \tag{3.22}$$

where $\mathbf{w} = (w_0, ..., w_M)^T$, $\mathbf{x}$ is the input data, and $(\phi_1, ..., \phi_M)^T$ are the *basis functions*. Here, $\mathbf{w}$ is the *weight vector or the parameters* to be trained, and w_0 is called

a bias parameter and used for any fixed offset in the data. Typically, the basis functions, ϕ_ks, are nonlinear, and are applied to the original data variables. In a linear model, $y(\mathbf{x}, \mathbf{w})$ is a nonlinear function of the input variables due to the non-linearity of the basis functions. A comprehensive treatment of the linear models can be found in [18].

The essence of the linear model in Equation 3.22 will now be generalized, towards an Integer Linear Program (ILP) for the problem. As mentioned previously, a feature vector, $\mathcal{V}^{i,j} = \langle v_1^{i,j}, \ldots, v_d^{i,j} \rangle$, is employed to measure the similarity between structural fragment se^i and sequence segment qe^j. Without loss of generality, it is assumed that $-1 \leq v_l^{i,j} \leq 1$. Each structural fragment, se^i, is associated with a weight vector, $\mathcal{W}^i = \langle w_1^i, \ldots, w_d^i \rangle$. The distance between a structural fragment, se^i, and a sequence segment, qe^j, is computed by the dot product between $\mathcal{W}^i$ and $\mathcal{V}^{i,j}$ such that

$$\mathcal{D}^{i,j} = \sum_{l=1}^{d} w_l^i v_l^{i,j}. \tag{3.23}$$

$\mathcal{V}^{i,j} = \langle v_1^{i,j}, \ldots, v_d^{i,j} \rangle$ can be regarded as a set of basis functions, similar to ϕ_ks in Equation 3.22; the task is to adjust the weight vectors $\mathcal{W}^i = \langle w_1^i, \ldots, w_d^i \rangle$ so that for some n_j, where se^{n_j} is a native-like structure for qe^j, $\mathcal{D}^{n_j,j}$ is ranked within the top k among other $\mathcal{D}^{i,j}$ for $1 \leq i \leq q$. This results in a system of pq linear models, for $1 \leq i \leq q$ and $1 \leq j \leq p$. It is noteworthy that only one set of $\mathcal{W}^i = \langle w_1^i, \ldots, w_d^i \rangle$ is trained for each structural element, se^i; and eventually, only one of the $\mathcal{D}^{i,j}$, $1 \leq i \leq q$, which is a native-like structure for qe^j, is to be ranked well.

This newly developed model is generic. Although a linear combination of the features is assumed, any linearity about $v_l^{i,j}$ is not, and they can contain quadratic terms, and so on. For example, in Equation 3.1, a feature vector with length of 180 is used. Each structure or sequence segment of length nine is represented by 9×20 frequency distribution matrices. The feature vector has a size of 180, and each entry is the absolute value of the differences between the corresponding entries. Here, the $\mathcal{V}^{i,j}$ values are pre-calculated.

The task here is to train the weight vectors $\mathcal{W}^i$. Thirty protein sequences, whose structures are known in Table 3.2 Part(B) are used as the training set. These proteins are parsed with a sliding window of length length nine, with step size one, to obtain the set of qe^j's. The structure space, se^i's, are obtained from the 40 proteins, also with the known structures in Table 3.2 Part (A).

For each sequence segment, qe^j, $\mathcal{Q}^j$, the set of structural fragments where the distance to qe^j's native structure is less than the distance threshold θ, is computed. The objective here is to optimize the weight vectors, $\mathcal{W}^i$, of the distance function such that there is a distance function that ranks at least k' elements in $\mathcal{Q}^j$ to the top k places. In the following formulation, $k' = 1$, for simplicity. It is easy to extend the proposed model such that, in each candidate list, at least k', $1 \leq k' \leq k$, native-like structures are included.

For $1 \leq i \leq q$, indexing the structural space, and $1 \leq j \leq p$, indexing the sequence segments, the ILP is as follows:

$$\min \sum_{j=1}^{p} g_j, \tag{3.24}$$

$$\mathcal{D}^{n_j,j} - \mathcal{D}^{i,j} \leq d_{n_j,i,j}(2+\epsilon) - \epsilon, \qquad n_j \in \mathcal{Q}^j, i \notin \mathcal{Q}^j, \forall j, \tag{3.25}$$

$$\sum_{1 \leq i \leq q, i \notin \mathcal{Q}^j} d_{n_j,i,j} \leq k - 1 + f_{n_j,j}(q - (k-1)), \qquad n \in \mathcal{Q}^j, \forall j, \tag{3.26}$$

$$\sum_{n_j \in \mathcal{Q}^j} f_{n_j,j} \leq |\mathcal{Q}^j| - 1 + g_j, \qquad \forall j, \tag{3.27}$$

$$\sum_{l=1}^{d} w_l^j \leq 1, \qquad \forall j, \tag{3.28}$$

$$\text{and } d_{n_j,i,j}, f_{n_j,j}, g_j \in \{0,1\}, \qquad w_i^j \in [0,1]. \tag{3.29}$$

The ILP formulation is summarized as follows.

- Variable $g_j = 1$ indicates no element in $\mathcal{Q}^j$ is included as one of the k elements for $\mathbb{S}_j$. Therefore, the objective of the ILP, Equation 3.24, is to minimize $\sum_{j=1}^{p} g_j$.
- Constant ϵ in Equation 3.25 is created as a gap to separate the native-like and non-native-like structural fragments. Ideally, the parameter settings should be

  $$\mathcal{D}^{n_j,j} + \epsilon \leq \mathcal{D}^{i,j}, \tag{3.30}$$

  where se^{n_j} is a native-like structure for qe^j, and se^i is a non-native-like structure for qe^j. Equation 3.25 is used to achieve this goal. It is clear that $-2 \leq \mathcal{D}^{n_j,j} - \mathcal{D}^{i,j} \leq 2$. If $d_{n_j,i,j} = 0$, the native-like structure se^{n_j} is ranked higher than the non-native-like structure, se^i.
- Equation 3.26 is used to check if a native-like structure, se^{n_j}, is in the candidate list of size k. If $f_{n_j,j} = 0$, the number of non-native-like structural fragments for qe^j is the candidate list is less than k. When this fails, $f_{n_j,j} = 1$.
- If $g_j = 0$, Equation 3.27 ensures that at least one native-like structure in $\mathcal{Q}^j$ for sequence segment qe^j is in qe^j's candidate list. The objective function, Equation 3.24, is used to minimize the number of sequence segments whose candidate list of size k does not contain a native-like structure. Equation 3.28 is to normalize the parameter distributions.

3.4.3 Basis Functions $V^{i,j}$

The basis functions, described in this chapter, are the entries extracted from the sequence and structural information. Specifically, $4 \times \ell$ entries, the $v_l^{i,j}$s, are created

for each structural fragment and sequence segment pair; that is, for each se^i and qe^j pair, the $\mathcal{V}^{i,j}$ feature vector has $d = 36$ entries (i.e. the ϕ_ks), with four entries, corresponding to the four types of scores, for each position.

For simplicity, in this section, a structural fragment is denoted as se, and a sequence segment, as qe, without superscripts. Both have length $\ell = 9$. The i-th positions of qe and se are signified as $qe[i]$ and $se[i]$, respectively. For each position i, the following four types of score entries are created.

Mutation Scores

The mutation score here is similar to that of ROSETTA, as shown in Equation 3.1, which computes the similarity score between profiles. The profiles for both the template and the sequence are obtained from five-rounds of PSI-BLAST with a cutoff of 9×10^{-4}. The mutation score between se and qe consists of ℓ entries. One entry is calculated for each corresponding pair of positions. The value at position i, $1 \leq i \leq \ell$ is defined as

$$\sum_{aa=1}^{20} S(aa, i) \times \log \frac{X(aa, i)}{S(aa, i)}, \tag{3.31}$$

where, from Equation 3.1, $S(aa, i)$ and $X(aa, i)$ are the frequencies of amino acid aa at position i for sequence segment and structural fragment, respectively. For our purpose, this score appeared to be more stable than other scores such as the City Block Metric, dot product, as well as the function in [90].

Secondary Structure Score

The secondary structure score measures the similarity between the secondary structure of se^j and that of qe^j. The secondary structure for a structural element, se^j, is computed by DSSP [88]. Then, after the secondary structure of a sequence is predicted by PSIPRED [85], the secondary structure sequence is parsed into qe^j. The program predicts the confidences α_i, β_i, and l_i for position i to be α-helix, β-sheet, and loop, respectively.

The secondary structure score at position i is computed as follows [166]:

- If the secondary structure type of $se[i]$ is α-helix, α_i is chosen.
- If the secondary structure type of $se[i]$ is β-sheet, β_i is used.
- If the secondary structure type is loop, l_i is selected.

Contact Capacity Score

For each structural position $se[i]$, a contact number, n_i, is calculated. There is a contact between two residues, if the distance between their C_β atoms is within the given cutoff, 7Å. The contact capacity is meant to measure the capacity that a residue has c contacts with any other of the residues in a protein.

Given a protein structure, $N(aa, c)$ is the number of residues with type aa and c contacts, $N(c)$ is the total number of residues with c contacts, $N(aa)$ is the number of residues with type aa, and N is the total number of residues. For an amino acid type aa, the capacity of it to have c contacts is defined as

$$CC(c, aa) = -\log \frac{N \times N(aa, c)}{N(c)N(aa)}.$$

The contact capacity score for position i is computed as $\sum_{aa=1}^{20} S(aa, i) \times CC(n_i, aa)$.

Environmental Fitness Score

The environment for each structural position is defined by a combination of the secondary structure type and solvent accessibility. Three secondary structure types are used: α-helix, β-strand, and loop; three accessibility levels are defined: buried, intermediate, and accessible. So in total there are nine states of the structural environment, and each structural position belongs to one of the nine environmental states. $F(E_i, aa)$ is the fitness score for an amino acid aa in environment state E_i. The fitness score between $se[i]$ and $qe[i]$ is calculated by $\sum_{aa=1}^{20} S(aa, i) \times F(E_i, aa)$. More details are given in [90].

3.4.4 Results

The methods described resulted in a program for generating structural fragments called FRazor. The program is implemented in C++ on Linux. The ILP is implemented with the package CPLEX. Additionally, some heuristics are built into the program to handle the cases where ILP cannot find an optimal solution within a reasonable amount of time.

Evaluation Criteria

Several criteria are used to evaluate the quality of the structural fragments obtained using FRazor. These criteria are namely, fragment coverage (fc-score), local fit approximation (lf-score), and position coverage (pc-score).

One way to evaluate the significance of the selected structural fragments for each target is to simply count the percentage of sequence segments, covered by the

structural candidate lists, for a given structure distance threshold. This percentage is referred to as the *fragment coverage*.

Local Fit Approximation is a criterion developed to evaluate the quality of a fragment library [93]. For each sequence segment, the most similar structure in terms of the RMSD from the structural candidate list is calculated. Then, the average of the RMSD values over the entire sequence segment is taken as the *local fit score*.

An evaluation more relevant to the purpose of protein structure prediction is to count the number of positions "correctly predicted" in target t. Here, "correctly predicting a position" means that at least one sequence segment containing the position is covered. The percentage of the positions which are correctly predicted is referred to as the *position coverage* (pc). This criterion was originally used in [141]. The positions are divided into three cases *α-helix, β-sheet, and loop*. The coverage is evaluated for each type of position.

Data Set

The data set consists of three parts: (1) the structure space, (2) the training set, and (3) the testing set. The structure space is a collection of structural fragments from which the candidate structural fragments are selected for a sequence segment. The training set consists of the fragments used to compute the parameters and the testing set contains proteins for evaluating the new method.

The proteins for structure space and training set are both from a non-homologous (less than a 30% homology) list with resolution $< 2\mathring{A}$, dated March 26, 2006. The list of these proteins is created by the program PISCES [158], and there are 3177 chains. The first 70 chains are used. The structure space consists of 40 protein chains, as shown in Table 3.2, Part A. These proteins are parsed with a sliding window of size $\ell = 9$ and step size 1. In total, there are 9,658 residues. The resulting structural space consists of 9,338 length-9 structural fragments. The training data comprises the other 30 chains, which are also shown in Table 3.2, Part B. They are parsed into length-$\ell = 9$ segments with a sliding window of step size 1. In total, there are 6,584 residues.

For the testing set, the proteins from CASP7, which were created after April, 2006 are employed. There are in total 94 proteins. Also, the testing set are parsed into segments of length $\ell = 9$. The CASP7 test proteins do not share a high sequence identity with the proteins in PDB released before March 26, 2006, which contain the proteins in the structure space and training set. Six test proteins, used in previous studies [69, 93, 141] are employed to compare the quality of the decoys, assembled from FRazor's fragments, with that of ROSETTA's fragments. These six test proteins are: Protein A (PDB code 1FC2), Homeodomain (1ENH), Protein G (2GB1), Cro repressor (2CRO), Protein L7/L12 (1CTF), and Calbindin (4ICB).

Table 3.2: Proteins for the structure space and training set.

A. Structure space:									
1ci4a	1zm8a	1j79a	1rlja	1zhva	1wlya	2a14a	2gc9a	1lg7a	1wkoa
1jfla	1t9ha	1lm5a	1kxoa	1xfia	1rqpa	1m15a	1z96a	1mla	1ail
1yksa	1q25a	1mj5a	2erba	2bsya	1lst	1g8aa	1wzca	1y9wa	1xkpc
1v4va	1se8a	1p9ha	1r17a	1qfta	1aol	1ju3a	1rsga	1atg	1s5aa
B. Training set:									
1olra	2byca	1yb5a	1pbwa	1v0ea	1orva	1jb7b	2ftra	1fj2a	1fp2a
2foma	1xtta	1suua	1xuua	1w2wb	1viaa	1r9wa	1fj2a	1dmga	2ah5a
1tc5a	2az4a	1mzwb	1ef1c	1uvqc	1ikta	1xfsa	1zava	1vk5a	1oyga

The first four letters are the PDB code, and the 5th letter is the chain id for each entry. The 5th letter is missing if the protein only has a single chain.

FRazor vs. City Block Metric

An interesting question is whether structural information, such as secondary structure, solvent accessibility, and contact capacity, facilitate the prediction of structural fragments. In this experiment, this question is explored by comparing FRazor with the CBM model [141], where the sequence profile is used. The experimental results are listed in Table 3.3, where the fragment candidate list size is set to be 25, the number of templates is 40; that is, the 40 proteins in Table 3.2 Part (A), and the fragment length is 9.

Observe Table 3.3. With the threshold value 0.5Å, the position coverage increases from 10.0% to 37.6%, and from 26.6% to 38.7% for the β-sheets and loops, respectively. With the threshold value 1Å, the position coverage increases from 56.4% to 89.6%, and 55.5% to 78.1% for β-sheets and loops, respectively. For the threshold 1.5Å, significant improvement is observed for the β-sheets and loops as well. Overall, the position coverage scores are 88.2% and 96.7% for the threshold values 1Å and 1.5Å , respectively, and the two values for CBM are 72.2% and 89.9%, respectively. Although the improvement for the α-helix looks small, because there is not much left to improve upon, 20% improvement is still observed over the remaining gaps for 0.5Å and 1Å.

In Table 3.4, the threshold value is 1Å ,and the results are compared by varying the candidate list size. The position coverage is displayed. The average percentage of improvement for the β-sheets is more than 30% with the same candidate list size. The average percentage of improvement for the loops is more than 20% for all the cases. From the table, it is evident that the position coverage is increased from 56.4% to 79.1%, and from 55.5% to 67.9% for the β-sheets and loops, respectively, whereas the fragment candidate size is reduced from 25 to 10, simultaneously. By using five as the candidate list size, FRazor's performance is better than that of CBM with 40 as the fragment candidate list size for the β-sheets and loops. When

Table 3.3: Position Coverage for the CBM vs. FRazor's Score Function.

	α-Helix		β-Sheet		Loop		Overall	
θ (Å)	CBM	FRazor	CBM	FRazor	CBM	FRazor	CBM	FRazor
0.5	94.2	95.1	10.0	37.6	26.6	38.7	49.4	55.1
1	98.2	98.6	56.4	89.6	55.5	78.1	72.2	88.2
1.5	99.7	99.7	89.3	98.2	81.3	93.3	89.9	96.7
2	100	100	99.7	99.8	96.9	98.9	98.6	99.4
2.5	100	100	99.9	99.9	99.7	99.7	99.8	99.8
3	100	100	100	100	99.9	100	99.9	100
3.5	100	100	100	100	100	100	100	100

Position coverage(%) is displayed. CBM is the City Block Metric. The first column θ (Å) is the native threshold. The fragment candidate list size (k) is 25. The fragment length is 9.

15 is chosen as the candidate list size, FRazor's performance is better than CBM with 40 as the candidate list size.

Table 3.5 depicts the results of fragment coverage and local fit criteria. In Table 3.5, the threshold value 1Å is chosen and the results from varying the candidate list size are compared. This table demonstrates that FRazor with a candidate list size 10 has a higher fragment coverage than CBM with a candidate list size 40 with scores of 43.3% and 40.8%, respectively.

For all these evaluation criteria, it is safe to draw a conclusion that FRazor is able to identify compact candidate lists for sequence segments. In addition, experiments are conducted where the fragment length and candidate list size are varied. These experimental results suggest that FRazor is stable and robust, and consistent improvement is observed.

Selecting Fragments from a Library

Sequence specific fragment candidate lists are able to model a protein more accurately than an independent fragment library. In this section, it is shown that FRazor yields more accurate fragment candidates list than an independent library by comparing them to the fragment libraries from [93]. From another aspect that each structural fragment can be mapped to an entry in a fragment library, FRazor is able to select a subset of fragments from a library for a sequence segment. The libraries from [93] with the fragment length 7 are used, and the library sizes are 50, 100, 150, 200, and 250. For a fair comparison, the performances of these libraries on the test data are evaluated. The library size is represented by L.

Table 3.6 shows the results of the Kolodny's fragment libraries, and FRazor's customized lists. By using a candidate list size of 25, the fragment coverage score

Table 3.4: Position coverage for the threshold value as 1Å.

	α-Helix		β-Sheet		Loop		Overall	
k	CBM	FRazor	CBM	FRazor	CBM	FRazor	CBM	FRazor
5	90.5	96.6	34.2	65.6	40.3	59.8	60.7	75.1
10	97.2	97.5	42.4	79.1	46.1	67.9	65.1	81.5
15	97.8	99.3	49.5	82.1	50.6	70.5	68.6	85.0
20	98.1	98.0	53.6	85.1	53.5	73.0	70.8	86.4
25	98.2	98.6	56.4	89.6	55.5	78.1	72.2	86.4
30	98.3	98.7	59.9	90.8	57.4	79.6	73.6	88.2
35	98.5	98.8	61.5	92.0	58.5	81.1	74.5	90.0
40	98.7	99.0	63.5	92.9	59.5	82.3	75.4	90.8

Position coverage score(%) is displayed. CBM is the City Block Metric. The first column is the fragment candidate list size. The fragment length is nine. The position coverage (%) is reported for the three cases. The threshold value is 1Å.

Table 3.5: Fragment coverage and local fit score for threshold value as 1Å.

	Fragment Coverage(%)		Local Fit Score(Å)	
k	CBM	FRazor	CBM	FRazor
5	29.2	37.9	1.860	1.542
10	33.1	43.3	1.592	1.338
15	35.5	46.8	1.468	1.240
20	37.0	49.6	1.393	1.176
25	38.2	51.5	1.342	1.133
30	39.3	53.2	1.301	1.097
35	40.1	54.6	1.272	1.072
40	40.8	55.6	1.247	1.050

CBM is the City Block Metric. The first column is the fragment candidate list size. Column 2 and Column 3 are the fragment coverage scores for CBM and FSS, respectively. Column 4 and Column 5 are the local fit scores for CBM and FRazor, respectively. The fragment length is 9. The threshold value is 1Å.

is better than that of the library with 200 fragments. The local fit score for 100 fragments is comparable with a fragment library of 250 fragments.

Application to Protein Structure Prediction

FRazor is also compared to ROSETTA's fragment generation model. This final test is to examine the quality of the decoys folded from the fragments generated by FRazor. In this test, ROSETTA's fragment generation method is replaced by FRazor, and the resultant decoys are compared to the decoys obtained using the

Table 3.6: Customized fragment lists vs. independent fragment libraries

	Fragment Coverage (%)		Local Fit Score (Å)	
L or k	KFL	FRazor	KFL	FRazor
25	–	45.3	–	0.763
50	36.2	40.5	0.754	0.667
100	40.7	55.7	0.673	0.589
150	43.3	58.6	0.633	0.554
200	44.0	60.4	0.603	0.531
250	46.3	61.8	0.585	0.515

KFL refers to for Kolodny's fragment libraries. The first column is the fragment candidate list size for FRazor, and is the library size for Kolodny's Libraries. Fragment Coverage (%) of a threshold 0.5Å is shown for Kolodny's fragment libraries in column 2 and for FRazor's distance function in column 3, respectively. The local fit score (Å) is shown for Kolodny's fragment libraries in column 4 and for FRazor's distance function in column 5, respectively.

original ROSETTA. To fairly evaluate FRazor, ROSETTA's energy function and its default setting are employed in FRazor. ROSETTA's fragment generation code is obtained from the ROSETTA package (version 2.0.1). For both FRazor and ROSETTA's fragment generation module, their structural fragments are selected from the same set of 40 proteins, which is included in ROSETTA's fragment generation module. It should be noted that these proteins are different from the 40 proteins in Table 3.2 Part (A). The same 30 proteins in Table 3.2(B) are used to train FRazor.

The six proteins that were used in previous studies [69, 93, 141] are used to evaluate FRazor. 1000 decoys are assembled for each protein by using the structural fragments, generated by FRazor and ROSETTA, and then compared in terms of the percentage of *good decoys*[1] and the average RMSD of all the 1000 decoys. As seen in Table 3.7, the use of FRazor in the place of ROSETTA's fragment generation method resulted in 1.8%-26% more good decoys. In addition, the average RMSD of the decoys generated by FRazor is much smaller for four of the six test proteins. For the other two test proteins, both FRazor and ROSETTA have similar average RMSDs.

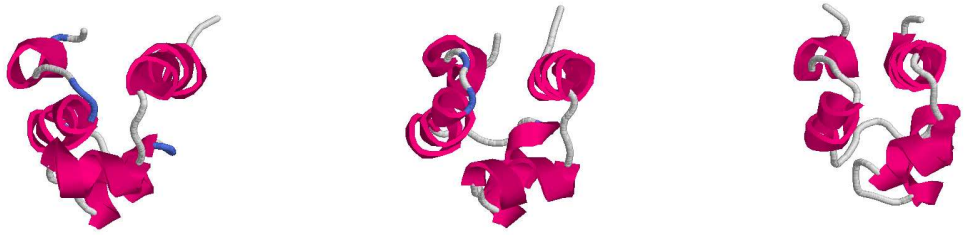
The best decoys generated when FRazor is in use also tend to have smaller RMSDs. For example, the best decoy generated with FRazor for the Cro repressor protein (PDB code 2CRO) has a much lower RMSD to its native structure than that generated with ROSETTA's original method. The first structure shown in Figure 3.1 is the best decoy for the Cro repressor protein generated by ROSETTA (RMSD 3.02Å to native), the second is the best decoy generated by FRazor (RMSD 2.57Å to native), and the third structure is the native. In addition to the Cro repressor protein, the best decoys for respectively Homeodomain (PDB code 1ENH) and

[1] A decoy is good if its RMSD to the native structure is less than 6Å.

Table 3.7: Decoy quality comparison between ROSETTA and FRazor

Target Protein			ROSETTA			FRazor		
PDB code	L	α, β	<6.0Å(%)	Best	Avg.	<6.0Å(%)	Best	Avg.
1FC2	43	2,0	20.5	2.59	7.3	38.6	2.60	6.4
1ENH	54	2,0	39.5	3.06	7.3	53.8	2.61	6.4
2GB1	56	1,4	89.8	1.88	4.3	90.6	2.04	4.4
2CRO	65	5,0	40.6	3.02	6.7	67.2	2.57	5.8
1CTF	68	3,3	9.2	3.42	9.1	11.0	3.14	8.4
4ICB	76	4,0	2.8	4.74	9.4	2.6	4.81	9.6

(Col. 1-3) Name and PDB code, length, and number of α-helices and β-strand of the target proteins.
(Col. 4-6) Percentage of good decoys (RMSD<6.0Å), RMSD of the best decoys, the average RMSDs of all decoys by ROSETTA
(Col. 7-9) Percentage of good decoys (RMSD<6.0Å), RMSD of the best decoys, the average RMSDs of all decoys by FRazor



(a) ROSETTA (3.0Å) (b) FRazor (2.6Å). (c) Native Structure

Figure 3.1: Two best decoys generated by ROSETTA and FRazor for the Cro repressor protein (PDB code 2CRO).

Protein L7/L12 (PDB code 1CTF) generated with FRazor, also demonstrated significantly lower RMSDs than the best decoys generated with ROSETTA's method. For the other three proteins: 1FC2, 2GB1 and 4ICB, the best decoys generated with ROSETTA's method are slightly better than those generated with FRazor.

3.5 Discussion

This chapter ends with some reflections on the results obtained.

3.5.1 Theoretical Issues of Independent Fragment Library

The PTAS, proposed in Section 3.3, depends on Lemma 1. For this reason, the technique can not be extended to the problem under distance measures where Lemma 1 cannot be applied, for example, the RMSD measure. However, should Lemma 1 apply to a distance measure, it should be easy to adapt the proposed PTAS to solve the problem for that distance measure.

It is possible to formulate variations of the k-consensus structural fragments problem, for example the following.

k-Closest Structural Fragments Problem under MSD	
Input:	n structural fragments $f_1, \dots f_n$, and a non-zero natural number $k < n$.
Output:	k structural fragments $g_1, \dots g_k$, minimizing the threshold $\max_{1 \leq i \leq n} \min_{1 \leq j \leq k} \mathrm{MSD}(f_i, g_j)$.

The cost function of the k-consensus structural fragments problem resembles that of the k-means problem, while the cost function of the k-closest structural fragments resembles that of the (absolute) k-center problem. One interesting problem for future study is whether this problem has a PTAS or not. It is not clear how one can generalize the technique employed in this chapter to the k-closest structural fragment problem under the MSD distance measure.

3.5.2 Is the Structural Fragment Space Continuous?

Clustering algorithms have revealed that the number of possible structural fragments is limited for a short sequence segment of three to seven residues. Take, for example, the extreme case of a sequence segment of only two amino acids. In this case, a library of size one would suffice, since two points can only form a line segment. For three residues, it is not difficult to create a library of size four, based on only avoiding steric clashes. This can be considered a discretization of the conformation space for three consecutive C_α atoms. If a similar discretization for more residues can be performed, then the conformation of the small structural fragments is continuous. This could also imply that the reason that clustering approaches work is due to discretization, not due to the fact that the number of small structure fragments have certain favorite conformations. The problem is formulated as follows.

Is Structural Fragment Conformation Space Continuous?	
Input:	A small integer l, $3 \leq l \leq 7$ and a threshold θ
Output:	A structural fragment library $F = f_1, ..., f_k$ such that: (1) Each fragment f_i contains l 3D points, (2) for any structural fragment f' of l residues in database: $\exists i$, such that $d(f', f_i) \leq \theta$, (3) F is obtained by protein geometric constraints only, and (4) the objective is minimize k.

Here, the term "F is obtained by protein geometric constraints only" means that no information from the known database is used when the structural fragment library is constructed.

3.5.3 Position Specific Structural Fragment Library

With regard to the method for finding structural fragment library proposed in this chapter, in spite of the favorable comparisons of the method against other methods in our experiments, there is still much space for improvement.

The scoring function, used to map a sequence segment to a structural fragment, currently consists of a mutation score, secondary structure score, contact capacity score, and environment fitness score. To improve the performance, it seems reasonable to use more scoring items. A promising way to accomplish this is to combine the scores from other threading results, like in [171, 173]. Currently all the scoring items depend on only a single position, which implies that the residues in a protein sequence are assumed to be independent. However, some residues are obviously correlated, and better performance might be possible if the correlation information is encoded into the scoring function. The challenge to do so is to deal with the sparsity in training data since there are many more parameters to be trained. Perhaps, some regularization technique can be developed to resolve this issue.

Using the method, significant improvement has been observed in the accuracy of the β-sheet and loop positions, which suggests that the loop regions could be predicted even more accurately. To do so, the program can be modified to assign weights to the positions of a structure automatically, an improvement which might be useful for identifying structure motifs. Here, a position with a small weight would imply that the position is unstable.

Chapter 4

Structure Sampling

4.1 Backbone Structure Prediction

The traditional algorithms for (backbone) structure prediction can be categorized into three general classes: homology modeling, threading, and *ab initio*.

Homology modeling is based on the observation that evolutionarily related proteins tend to share similar structures. Such methods attempt to detect evolutionary relationships by aligning the target protein sequence with the proteins in a database, without using their structural information, and then derive coordinates of each atom according to this sequence-sequence alignment. However, homology modeling methods work only for those proteins with a high sequence identity with some proteins in the database. This restricts their applicability.

A threading method evaluates how well the target sequence fits a known structure by a sequence-structure alignment. By using the structure information in addition to the sequence information, threading techniques can be applied to proteins with low sequence identity.

In contrast, an *ab initio* method is not based on comparison with known proteins. Typically, these methods work by constructing a protein structure that is minimal with respect to some score function, which usually estimates the conformational energy of the proteins. The advantage of *ab initio* methods is that the possible structure spaces are not confined to the known structures in a structure database. Theoretically, any structural conformations can be generated and tested by an *ab initio* method.

4.2 A Principle of Parsimony-based Framework

Instead of considering the three methods separately, the framework proposed here includes features from all the methods, but differs from them in basic principle. More precisely, the objective of the new framework is not to identify easy targets

accurately as in PSI-BLAST (a homology modeling method), or to identify harder targets as in RAPTOR [169] (a threading method), or to address *ab initio* prediction specifically, as in ROSETTA [141]. A typical *ab initio* method consists of various stages such as Monte Carlo fragment assembly, clustering, selection, and refinement, where usually, different methods are employed. In the new framework, the use of different methods in different stages is avoided as much as possible.

The framework is designed on the principle of parsimony, a principle which has found uses in many areas [15, 107]. The aim is towards a structure prediction method which takes an input and outputs the final structure in a way that is as simple as possible. This method is to include homology modeling, threading, fragment assembly (all its stages), loop modeling, refinement, side chain packing, and consensus; that is, the method in mind is to be simple, robust, and effective.

This chapter presents the initial efforts towards the design of such a framework, as well as an initial implementation of the framework called FALCON. Experimental results are also presented to demonstrate the effectiveness of FALCON.

Some of the ideas of the proposal are derived from three lines of research: fragment assembly, HMM sampling, and Ramachandran basins.

The most successful approach to *ab initio* structure prediction is to use short structural fragments to model the local interactions among the amino acids of a segment, and utilize the non-local interactions to arrange these short structural fragments to form native-like structures [141]. Despite the importance of non-local interactions in directing the search for native-like protein structures, the relationship between the local structures and the interactions among amino acids within a local structure remains active issues of research. An accurate prediction of the local structural bias for a sequence segment is critically important to protein structure prediction.

According to the Levinthal paradox [103], the number of possible conformations of a protein chain is exponential in the protein sequence length due to the high degrees of freedom of the unfolded polypeptide chain. As a consequence, a brute force enumeration of all the possible conformations for a given sequence is both computationally and physically infeasible. However, the local structural bias information restricts the possible conformations of each sequence segment, and therefore, narrows down the conformation space of the entire polypeptide chain significantly.

A structural motif is a straightforward description of a local structural bias, and the idea can be traced back to [122]. Here, a protein fold is modeled as an assembly of smaller building blocks from the regular secondary structure elements. In the past years, active research [26, 27, 29, 58, 71, 108, 131] has been conducted to define local structural motifs, and to analyze their structural characterization as well as sequence preferences. The sequence preferences can be used to predict structural motifs for new sequences. Another approach is to search for recurrent sequence patterns first, and then study the structural motif shared by these recurrent sequence patterns. This approach can identify new structural motifs, since the important structural properties need not be specified in advance. HMMSTR [28], an HMM of

a structural motif space, is an attempt to describe the overlaps of structural motifs, and the transition probability between motifs. HMMSTR can be considered as a probabilistic version of the structural motif library.

Structural motifs serve as the foundation to obtain better predictions. For example, ROSETTA [141] generates 9-mer structural fragments from known protein structures as building blocks, whereas TASSER [171, 173] generates fragments of various lengths from the threading results. In spite of the progress in fragment assembly methods, these methods are handicapped by their inherently discrete nature; that is, the structural motif library is discrete, whereas the conformation space of a protein is continuous. Therefore, it is impossible to cover the entire conformation space with a limited number of structural motifs. This drawback limits the accuracy of protein structure prediction [63, 78].

An alternative way to describe the local structural bias is the Ramachandran basin [128]. It refers to a specific region of the Ramachandran plot, imposed by local interactions among amino acids. The Ramachandran basin provides a convenient way to present the preference of a specific torsion angle. Colubri *et al.* [42] have employed the Ramachandran basin technique to investigate the levels of representation required to predict the protein structure. Specifically, the authors tested the ability to recover the native structure from a given Ramachandran basin assignment for each amino acid. In this method, the Ramachandran plot is divided into five previously determined Ramachandran basins. By decomposing the Ramachandran plot into four or more basins, Shortle [135] has calculated the propensities of amino acids mapped to each basin. Shortle has argued that these propensities are the results of local side-chain-backbone interactions, and can restrict the denatured conformation ensemble to a relatively small subset of native-like conformations. Also, Gong *et al.* [63] have investigated the protein structure reconstructing problem from a coarse-grained estimation of the native torsion angle. The only difference in these works lies in the definition of the Ramachandran basin. The authors have partitioned both the ϕ and ψ angle intervals into six ranges, each of 60°, thus partitioning the Ramachandran map into 36 basins uniformly.

These three studies demonstrate that the knowledge of the torsion angles helps in the construction of small-size proteins. However, these three works partition the Ramachandran plot into basins in random manners, without statistical explanations to describe the torsion angle distributions of each basin. Furthermore, to give each residue a coarse Ramachandran basin assignment, the native structure should be known in advance, which makes these frameworks infeasible for real-life protein structure prediction.

As yet another precursor to the work in this thesis, Hamelryck *et al.* [69] have applied FB5, a directional distribution, to parameterize the local structural bias. By using the tool, they investigated the local bias in (θ, τ) space, rather than in (ϕ, ψ) space. Here, in (θ, τ) space, a virtual bond with bond length $3.8\mathring{A}$ is created between each pair of adjacent C_αs. The θ angle is the bond angle between three adjacent C_αs and the τ is the dihedral angle for any four adjacent C_αs.

In this method, the local structural bias for each amino acid is trained via an HMM called FB5-HMM. In contrast, Xu *et al.* [176] proposed the CRFSampler, a protein structure sampling framework based on another probabilistic graph model, Conditional Random Fields [18], with improved results. The success of this method suggests an advantage to the use of continuous torsion angle distributions over discrete structural motifs. By using the torsion angle distribution technique, it is possible to generate conformations with local structures not occurring in the structural fragment library. In addition, experimental results have demonstrated that the derived local biases can help to generate native-like conformations, and support the view that relatively few conformations are compatible with the local structural biases.

In spite of the sound theoretical basis, the approach adopted by FB5-HMM suffers from several problems. First, FB5-HMM reports the optimal number of local biases as 75 by training on a large set of representative protein structures. In other words, the (θ, τ) map which is used to model the local bias is partitioned into 75 basins. This partition scheme implies that a protein sequence of length n has a conformation space of size $O(75^n)$, which is astronomically larger than the estimation of $O(1.6^n)$ by [142]. In addition, it is challenging to select suitable models from these 75 local biases for a particular residue. Secondly, the FB5-HMM-derived distributions are general, while a residue might have specific preferences of distributions, hence it is possible that none of the 75 local biases turn out to be appropriate. Thirdly, FB5-HMM is incapable of capturing the relationships among the residues of a local segment. As a consequence, though equipped with an elegant statistical model, FB5-HMM displays lower prediction accuracies than ROSETTA [70].

4.3 New Framework

Here, a simple and unified framework for protein structure prediction is proposed. The plan is to probabilistically sample protein structure conformations compatible with local structural biases for a given protein. The architecture of the model is as follows.

1. For residue i, several *Cosine* models [116] are used to describe the local bias of its torsion angle pair (ϕ_i, ψ_i).

2. A position specific (HMM) is used to capture the dependencies among the local biases of the adjacent residues, according to selected fragments [108,141]. This HMM is referred to as the Fragment-HMM.

3. The Fragment-HMM is used to sample a sequence of torsion angle pairs for the given protein sequence. An energy function is used to evaluate the generated decoys, and to direct the sampling process to better decoys.

4. The generated decoys are fed back to produce more accurate estimations of local structural biases, a more accurate Fragment-HMM, and thus, better decoys. This step is executed iteratively to increase the quality of the final decoys until convergence.

The novel model has advantages over existing works as follows.

- The Fragment-HMM model combines a very successful fragment assembly method [141] with the elegant ideas explored in FB5-HMM [70]. Rather than using the fragments as building blocks, the fragments are employed to produce local bias information. The directional distribution are used to model the local biases, and an HMM is used to explore the dependency of the adjacent residues. Unlike FB5-HMM, the proposed Fragment-HMM is position specific.

- The design of Fragment-HMM inherently suggests the refinement process in Step 4. It is readily seen that this also applies to obtaining fragments from template structures, for example, templates from threading. Thus, this naturally enables homology modeling, threading, refinement (requiring more hidden nodes to model the side chains), loop modeling, and consensus.

- Step 4 is akin to the primal and dual optimization process in linear programming. Here, the primal process aims to minimize energy by discriminating the decoys with an energy function; the dual process is done via sampling Fragment-HMM to improve the estimation of the torsion angles. Step 4 differs from similar iterative methods in the traditional fragment assembly methods that end with a population of decoys, where some decoys may be possibly good and some possibly bad. Step 4 here would not stop at such a point, but would iterate until convergence is achieved.

- In the new framework, the search space is narrowed step by step, which makes it different from a Monte Carlo (MC) technique-based fragment-assembly-based protein structure prediction. The MC-based methods tend to be inefficient, since they do not typically make use of the search space's characteristics to reduce its size. For example, for a protein of length n, its search space size would be of $O(200^{n-l})$, if each sequence segment consists of 200 candidate structural motifs, and l is the fragment length. This search space remains unchanged for the entire MC-based method's search process. In contrast, Fragment-HMM narrows the search space after each iteration, as the local structural biases are estimated more and more accurately.

This framework for protein structure prediction is implemented in C++, in a program called FALCON (**F**ragment-HMM **A**pproximating **L**ocal Bias and **Con**sensus).

4.4 Methods

4.4.1 Torsion Angle Pair Sequences

In this chapter, it is assumed that that torsion angle pair sequences can accurately parameterize the 3D backbone structure.

4.4.2 Representing the Local Biases of Torsion Angle Pairs

The local structural bias for residue i is represented by the joint distribution of its torsion angle pair (ϕ_i, ψ_i). The *cosine* model, a bivariate von Mises distribution over angular or directional space is adopted [116, 145]. The probability density function of *cosine* model is specified by five parameters κ_1, κ_2, κ_3, μ, and ν as follows

$$f(\phi, \psi) = c(\kappa_1, \kappa_2, \kappa_3) e^{\kappa_1 \cos(\phi - \mu) + \kappa_2 \cos(\psi - \nu) + \kappa_3 \cos(\phi - \mu - \psi + \nu)},$$

where μ is the mean value of ϕ, ν is the mean value of ψ, and $c(\kappa_1, \kappa_2, \kappa_3)$ is a normalization constant expressed as

$$c(\kappa_1, \kappa_2, \kappa_3)^{-1} = (2\pi)^2 \left\{ I_0(\kappa_1) I_0(\kappa_2) I_0(\kappa_3) + 2 \sum_{p=1}^{\infty} I_p(\kappa_1) I_p(\kappa_2) I_p(\kappa_3) \right\},$$

in which $I_r(\kappa)$ is the modified Bessel function of the first kind and order r [3].

An alternative bivariate circular distribution is the *sine* model [145]. Mardia *et al.* [116] have argued that *cosine* model outperforms the *sine* model due to the ability to fit more closely a larger set of distributions.

Given a set of torsion angle pairs $A = \{(\phi, \psi)\}$, a set of M *cosine* models is used to parameterize these data. The cosine models are combined to form a mixture model, which is formulated as

$$F(\phi, \psi) = \sum_{j=1}^{M} w_j f_j(\phi, \psi), \tag{4.1}$$

where f_j, $1 \leq j \leq M$, denotes a *cosine* model with parameters, $\theta_j = (\kappa_1^j, \kappa_2^j, \kappa_3^j, \mu^j, \nu^j)$, and w_j is the weight of model j with $\sum_j w_j = 1$. An Expectation-Maximization (EM) algorithm is employed to derive the most likely estimation of the parameters of the mixture model [116].

The number of *cosine* models to fit these data is unknown in advance. It is vital to choose a suitable M. Here, Rissanen's minimum description length (MDL)

principle is applied [105, 107] to determine the best value for M; that is, M is chosesn to minimize the following.

$$\mathrm{MDL}(A) = -2 \ln L(A, M) + 5 * \ln(|A|), \tag{4.2}$$

where L is the likelihood that the M mixture models explain A, and five is the number of parameters in each model.

4.4.3 Fragment-HMM: Position Specific Hidden Markov Model

An HMM is used to capture the local dependencies among the adjacent residues. Unlike FB5-HMM, the newly developed HMM is position specific; that is, each residue is associated with a specific subset of hidden nodes, and the subset for all the residues is mutually disjointed.

Model Topology

An HMM is a directed graph, where the vertices denote the hidden nodes, and the directed edges are used to capture transition and emission probabilities. For each residue i, a set of possible hidden nodes, denoted as H_i, are obtained. Given two adjacent hidden node sets H_i and H_{i+1}, a directed edge $\langle h, h' \rangle$ is created for each pair of hidden nodes $h \in H_i$ and $h' \in H_{i+1}$. Each possible hidden node (denoted as h) has two types of emissions: a secondary structure type (denoted as S), and a torsion angle pair, $T = (\phi, \psi)$.

For the i-th amino acid, the position specific HMM describes the following joint probability:

$$\mathrm{Pr.}(\mathbf{S}, \mathbf{T}) = \sum_{h \in H_i} \mathrm{Pr.}(\mathbf{T}|\mathbf{S}, h)\mathrm{Pr.}(\mathbf{S}|h)\mathrm{Pr.}(h),$$

where $\mathbf{S}$ is the secondary structure type for the i-th amino acid, and $\mathbf{T}$ is its torsion angle pair.

Figure 4.1 shows an example of Fragment-HMM for five residues. Each residue is associated with a hidden node subset. As illustrated, the hidden node subset, H_1 for residue one, has two hidden nodes, whereas H_2 for residue two has three possible hidden nodes. Each hidden node is associated with its own *cosine* model.

Creating Hidden Nodes

The novel HMM is both position and sequence specific. There is no assumption that the training data is available for the target sequences to be predicted. Therefore, the classical Baum-Welch [16] method cannot be applied to estimate the parameters. Here, the hidden nodes are built and the parameters are estimated with a position specific fragment library, and hence, the name Fragment-HMM.

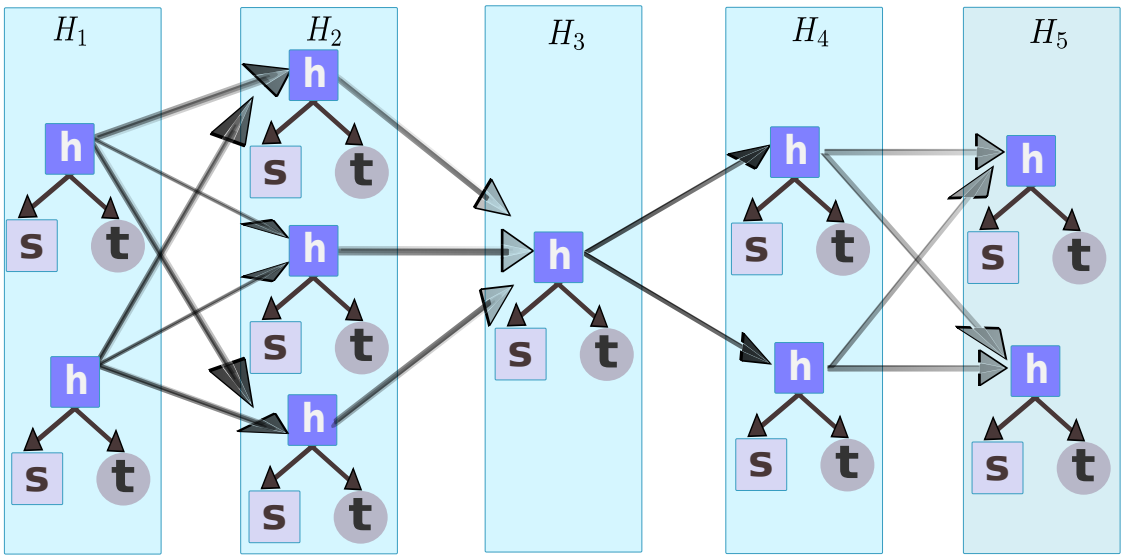


Figure 4.1: Fragment-HMM: a position specific hidden Markov model.

The construction process is broken down into three steps. First, the target sequence is parsed into segments with a sliding window of length ℓ and step size one. There are $n-\ell+1$ segments. These sequence segments are indexed by 1, 2, ..., $n-\ell+1$. For sequence segment i, a subset of structural fragments is predicted via ROSETTA or FRazor. A structural fragment for segment i consists of a predicted torsion angle pair and a secondary structure type for each residue from i to $i+\ell-1$. The set of structural fragments is denoted as $\mathcal{F}$. Secondly, the predicted torsion angle pairs for residue i are retrieved from the fragments in $\mathcal{F}$, and the EM method is used to generate a set of *cosine* models. Lastly, for each *cosine* model, a hidden node is created.

The *cosine* model, specified by hidden node h as m_h is identified. The density of m_h with parameters (ϕ, ψ) is written as $f_h(\phi, \psi)$.

Estimating Transition Probabilities

Here, the parameter estimation method is described without considering the secondary structures for clarity. The secondary structure information is easily integrated into the new framework.

The transition probabilities are estimated using the fragment library $\mathcal{F}$. Given a fragment $q \in \mathcal{F}$ and a hidden node $h \in H_i$, the probability that h emits the torsion angle pair, predicted by q for residue i, is defined as follows.

1. If q contains a predicted torsion angle pair (ϕ, ψ) for residue i, the value of the probability density function, f_h, with parameters, (ϕ, ψ), is employed.

2. Otherwise, the probability is 0.

The aforementioned probability is denoted as $g_h(q)$.

The joint probability $\Pr.(h' \in H_{i+1}, h \in H_i|q)$ for edge $\langle h, h' \rangle$, given fragment q, is specified as follows. If a structural fragment q does not contain the predicted torsion angles for both residue i and residue $i+1$, $\Pr.(h \in H_i, h' \in H_{i+1}|q) = 0$; otherwise, $\Pr.(h \in H_i, h' \in H_{i+1}|q)$ is defined as

$$\Pr.(h \in H_i, h' \in H_{i+1}|q) = \frac{g_h(q) g_{h'}(q)}{\sum_{h \in H_i, h' \in H_{i+1}} g_h(q) g_{h'}(q)}.$$

The probability is normalized to ensure

$$\sum_{h \in H_{i+1}, h' \in H_i} \Pr.(h \in H_i, h' \in H_{i+1}|q) = 1.$$

Then, the joint probability $\Pr.(h \in H_i, h' \in H_{i+1})$ is calculated by

$$\Pr.(h \in H_i, h' \in H_{i+1}) = \sum_{q \in \mathcal{F}} \Pr.(h \in H_i, h' \in H_{i+1}|q) \Pr.(q),$$

where $\Pr.(q)$ can be estimated as the inverse of the number of fragments in $\mathcal{F}$ which contain the predicted torsion angle pairs for both residue i and residue $i+1$.

Now, the transition probability $\Pr.(h' \in H_{i+1}|h \in H_i)$ is computed, according to

$$\frac{\Pr.(h' \in H_{i+1}, h \in H_i)}{\sum_{h' \in H_{i+1}} \Pr.(h \in H_i, h' \in H_{i+1})}.$$

The distribution of hidden nodes $h \in H_i$, $1 \leq i \leq n-1$ is expressed by:

$$\Pr.(h \in H_i) = \sum_{h' \in H_{i+1}} \Pr.(h \in H_i, h' \in H_{i+1}).$$

A Fragment-HMM of order one has been specified here for simplicity. A Fragment-HMM with a higher order can be defined accordingly. Order-7 and order-2 Fragment-HMMs are used in FALCON.

4.4.4 Sampling Protein Structure Conformation

The sampling of backbone conformations is performed using the derived position specific Fragment-HMM, as follows.

- *Sampling of hidden nodes*: A sequence of hidden nodes is first sampled. To do so, a hidden node h for residue 1 is first picked from the set H_1 according to the probability $\Pr.(h)$, $h \in H_1$. Subsequent nodes are then sampled as follows. Given that the hidden node h is sampled for residue i, a hidden node h' is sampled for residue $i+1$ according to the transition probability $\Pr.(h' \in H_{i+1}|h \in H_i)$.

- *Sampling of torsion angle pairs*: A sequence of torsion angle pairs is sampled, one pair per residue, according to the *cosine* model, specified by the respective hidden node. A backbone is constructed according to these torsion angles with ideal bond lengths and bond angles. Coupling the angle sampling process, a sequence of secondary structure types is also sampled. These secondary structure types are useful for the energy function to evaluate the sampled structure.

4.4.5 Conformation Optimization

The conformation optimization process is performed as follows. In order to reduce biases in the comparison of the new framework to ROSETTA, the energy function of ROSETTA 2.1.0 (released Sept. 2006) is used.

Initially, an entire sequence of angle pairs is sampled, and a new 3D backbone structure is constructed from these angles. Then, a subsequence of torsion angle pairs are sampled again for a given backbone structure and a new 3D backbone structure is built. If the new structure has an energy equal or better than that of the previous structure, the new structure is accepted. Otherwise, it is accepted with a certain probability by the Metropolis criteria. The process is repeated until the energy is converged, or the maximum number of iterations is reached.

4.4.6 Iteratively Improving the Fragment-HMM

The above procedure eventually leads to the generation of a set of decoys, as guided by the given energy function. Given that the energy function biases towards the native-like structures, using these decoys as a position specific fragment library would lead to the pruning of less feasible *cosine* models, as well as the forming of better *cosine* models. Then, these new *cosine* models can be integrated to build even more accurate Fragment-HMM with refined transition and emission probabilities. In turn, the more accurate *cosine* models and HMM would result in more native-like structures.

Hence, iteratively, more and more accurate *cosine* models can be obtained, given that the energy function favors the native-like structures.

4.5 Results

4.5.1 Data Set

Again, the six proteins that were used in the previous studies [70, 93, 141] are selected (see Table 3.7). Furthermore, FALCON is tested on eight larger proteins

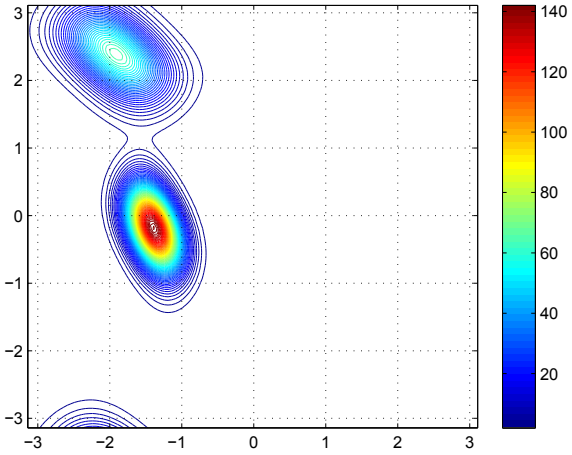


Figure 4.2: Cosine models for residue 13 of protein 1FC2.
The x-axis is the ϕ angle and the y-axis is the ψ angle.

with more than 100 residues. These proteins are selected from the CASP7 free-modeling targets, and are listed in Table 4.6.

The position specific fragment library for each protein is obtained from the recently released ROSETTA version 2.1.0. Its structural fragments are selected from a set of 1,020 protein chains, which are included in ROSETTA's fragment generation module. Also, ROSETTA's energy function and its default settings are used.

Two hundred structural fragments are predicted for each 9-mer sequence segment in the query sequence. In this process, homologues of the query protein are identified by running NCBI-Blast [11], and then removing these homologues in the following fragment predicting step. Thus the possible overlap of the training and testing sets are avoided.

4.5.2 Torsion Angle Distributions

The most likely torsion angle distributions are derived for each residue of the six proteins. A typical and concrete example for residue 13 of protein 1FC2 is plotted in Figure 4.2. Figure 4.2 contains two *cosine* models, which are centered at $(-1.39, -0.18)$ and $(-1.90, 2.39)$. This indicates that this residue has two possible local bias preferences: one corresponds to a β secondary structure type, and the other corresponds to an α secondary structure type. This is one of the primary differences between the Fragment-HMM and FB5-HMM. In FB5-HMM (and the CRFSampler), the number of distributions and the corresponding parameters are uniform for all the residues.

Table 4.1 lists the average number of *cosine* models per residue for the six proteins generated by the fragment library. These proteins average 1.66 *cosine* models per residue. Most residues have no more than two *cosine* models. This observation

Table 4.1: Number of cosine models per residue.

Target Protein	# Residue				
Name, PDB code	1	2	3	4	Ave.
Protein A, 1FC2	12	25	3	2	1.66
Homeodomain, 1ENH	24	24	6	0	1.21
Protein G, 2GB1	28	21	7	0	1.63
Cro repressor, 2CRO	52	12	1	0	1.22
Protein L7/L12, 1CTF	50	14	3	1	1.34
Calbidin, 4ICB	47	23	3	3	1.50

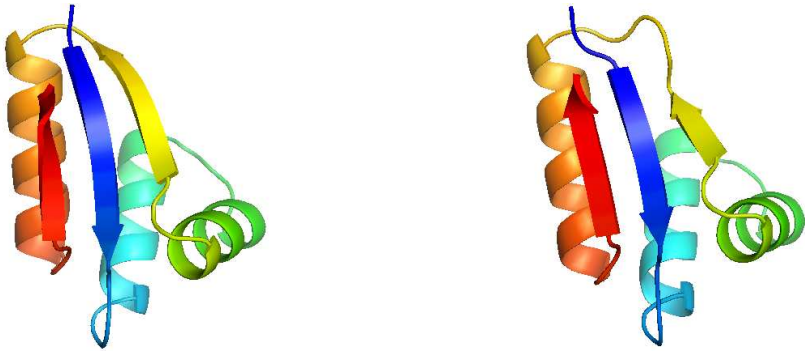
Column 2-5 are the number of residues with 1,2,3, and 4 *cosine* models, respectively. Column 6 is the average number of *cosine* models per residue.

confirms the fact that for a protein, relatively few conformations are compatible with the local biases of all the residues. Interestingly, the number of possible conformation clusters, C, is estimated to be $C = 1.6^n$ [142], which is consistent with the estimations given in [49,149], coinciding with the study here. There are several different estimations: according to the Levinthal paradox [103]. The conformational space has a size of at least $C = 3^n$ [178]; the number of possible conformation clusters can be estimated to be $C = 4^n$ via decomposing the Ramachandran map into 4 basins [128, 135]. Hamelryck *et al.* [70] assigned 75 possible states for each amino acid by decomposing the (θ, τ) plane, which implies a conformation space of size $C = 75^n$. Compared with these estimations, 1.6^n is drastically smaller. This observation suggests that

- local structural biases can be accurately described and captured; and
- the conformation space to be searched is greatly reduced, and thus, it might be possible to sample a native-like structure from a conformation space, where the conformations are compatible with the derived local biases.

4.5.3 Local Bias Representation: Fragment-HMM versus Structural Fragments

Using HMM to represent local structural biases, FB5-HMM failed to demonstrate any advantage over ROSETTA, which represents local structural biases with structural fragments. It is hence interesting to investigate, if the representation of local structural biases using Fragment-HMM with *cosine* models offers any advantage. To answer this, experiments are conducted to compare FALCON (without Step 4) to ROSETTA (version 2.1.0), in terms of the percentage of good decoys (below 6Å to the native structure) they obtain, as well as the RMSD values of their reported best decoys. To remove influences due to differences in energy function,



(a) The native structure of protein 1CTF. (b) The predicted structure of protein 1CTF.

Figure 4.3: Native structure and the best decoy predicted by FALCON (The RMSD is 0.557Å).

ROSETTA's energy function is used for both programs. The input fragment libraries are generated by ROSETTA and are identical for both programs.

In this experiment, 1,000 decoys are generated for each protein by each of ROSETTA and FALCON. 6Å is used as the cutoff value for the good decoys, and the same criteria is used in [70]. Since the decoys for ROSETTA and FALCON are generated independently, the percentage of good decoys is not expected to fluctuate too much when more decoys are generated.

In Table 4.2, FALCON generates significantly more good decoys than ROSETTA. FALCON improves the percentage of good decoys for 1FC2, 2GB1, 2CRO, 1CTF, and 4ICB; that is, five out of six proteins. Especially for 2GB1, 1CTF, and 4ICB, the improvements are from 53.7%, 14.3%, 19.9% to 93.4%, 25.6%, and 46.3%, respectively. The quality of the best decoys for these five proteins: 1FC2, 2GB1, 2CRO, 1CTF, and 4ICB are improved as well. A structure with RMSD of only 0.557Å to the native structure for 1CTF is displayed in Figure 4.3.

Although Fragment-HMM inherited ideas from FB5-HMM [70], it has incorporated enough improvements to become a significantly stronger model than FB5-HM. FB5-HMM describes the local biases by using 75 basins in the (θ, τ) plane, whereas FALCON uses an average of only 1.6 basins on average. Under admittedly different conditions, for the same six proteins in Table 4.2 and in that order, FB5-HMM [70] reports the best decoy accuracies as 2.6Å, 3.8Å, 5.9Å, 4.1Å, 4.1Å, and 4.5Å, respectively, and good decoy percentages (<6Å): 17.1%, 12.1%, 0.001%, 1.09%, 0.35%,

Table 4.2: Decoy quality of ROSETTA and FALCON.

Target Protein	ROSETTA		FALCON	
	Best	<6.0Å(%)	Best	<6.0Å(%)
Protein A, 1FC2	2.82	80.2	2.64	94.3
Homeodomain, 1ENH	1.52	94.4	1.81	92.8
Protein G, 2GB1	2.21	53.7	2.18	93.4
Cro repressor, 2CRO	2.56	70.4	2.48	75.8
Protein L7/L12, 1CTF	1.44	14.3	0.56	25.6
Calbidin, 4ICB	3.87	19.9	2.93	46.3

Column 2-3: RMSD of the best decoy ($\mathring{A}$) and percentage of the good decoys (RMSD$< 6\mathring{A}$) for ROSETTA. Column 4-5: are the corresponding values for FALCON.

and 0.38%, respectively, from 100,000 decoys.

4.5.4 FALCON: Zero in on the Native Structure

In the above experiment, Step 4 has been disabled in FALCON. The effectiveness of the iterative procedure is now investigated with the following experiment.

In this experiment, six iterations are executed for each protein. 1,000 decoys are generated at each iteration per protein. The first iteration takes as the input the position specific fragment libraries from ROSETTA. The $(i + 1)$-th iteration takes the set of decoys generated by the i-th iteration as input.

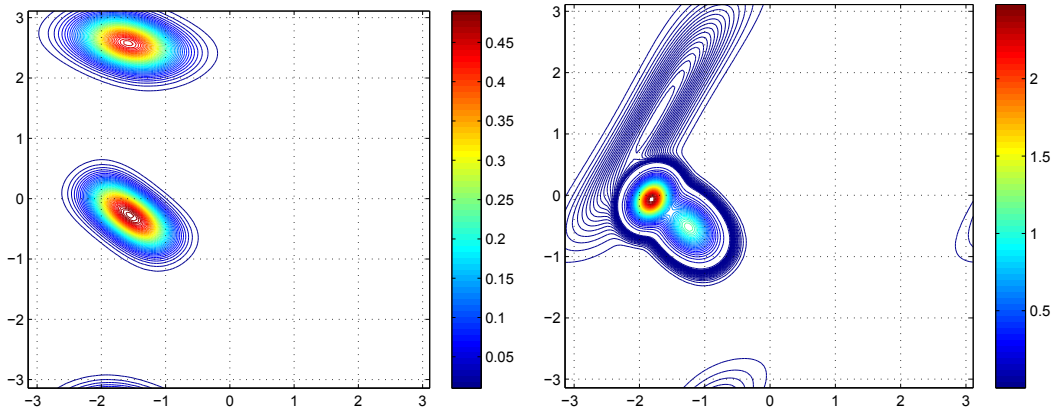
Table 4.3: RMSD distribution over iterations for protein 2CRO.

	#Iterations					
RMSD (Å)	1	2	3	4	5	6
$[0, 3)$	0.1	0	0.1	0.1	0	0
$[3, 4)$	22.8	47.2	75.3	87.9	94.7	94.9
$[4, 5)$	41.5	45.4	24.5	12.0	5.3	5.1
$[5, 6)$	11.4	4.7	0.1	0	0	0
$[6, 7)$	8.5	0.8	0	0	0	0
$[7, \infty)$	15.7	1.5	0	0	0	0

Col. 2-7: Percentages of decoys with the RMSD values in the corresponding intervals.

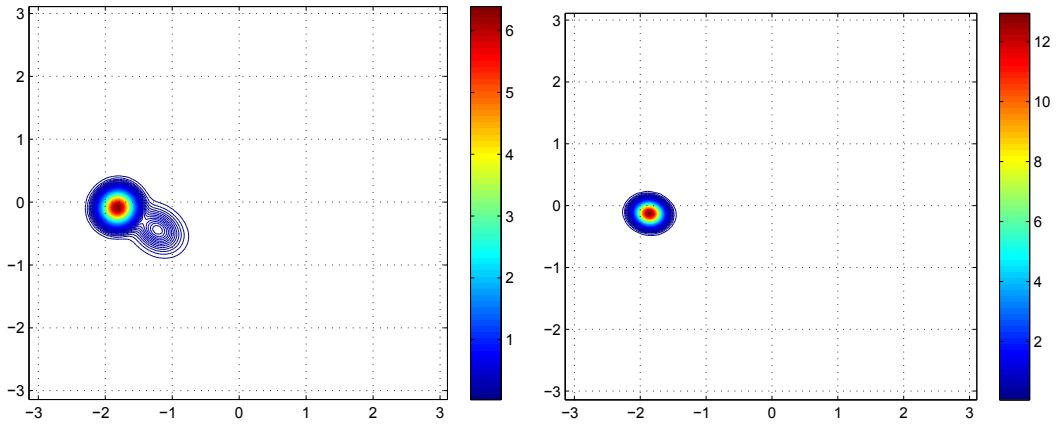
Table 4.3 displays the RMSD values of the decoys at each iteration for protein 2CRO. The RMSDs are observed to converge. After five iterations, the RMSD values of 94.9% of the decoys converge to the range [3Å, 4Å]. Both the best and the worst decoys disappear gradually over the iterations. However, the best decoys

decrease far slower than the worst decoys. The decoy RMSD distributions for the other proteins exhibit similar trends.



(a) Iteration #1: Two cosine models centered at $(-1.55, -0.28)$ and $(-1.58, 2.57)$.

(b) Iteration #2: Three cosine models centered at $(-1.25, -0.52)$, $(-1.75, 1.26)$, and $(-1.82, -0.07)$.



(c) Iteration #3: Two cosine models centered at $(-1.22, -0.44)$ and $(-1.82, 0.09)$.

(d) Iterations #4 and #5: One cosine model centered at $(-1.86, -0.13)$, and then to $(-1.86, -0.13)$ in Iteration #6.

Figure 4.4: Evolution of torsion angle pair distributions for residue 41 of protein 2CRO. The x-axis is the ϕ angle and the y-axis is the ψ angle.

Figure 4.4 shows the evolution of the torsion angle pair distributions for residue 41 of protein 2CRO. Initially, the torsion angle pairs, acquired from structural fragments, display two clusters: one lies in the β-strand area; and the other lies in the α-helix area, both initially incorrect. Interestingly, the subsequent iterations tend to correct the torsion angle distributions step by step.

At the second iteration, the initial two clusters are diminishing, and a new cluster, centered at $(-1.82, -0.07)$, emerges. At the third iteration, the β-strand distribution disappears completely, and the new cluster becomes dominant. The α-helix distribution disappears at the fourth iteration. In the fifth and sixth iterations, the new cluster becomes denser and denser. Finally, a distribution centered

Table 4.4: Percentage of good decoys with RMSD below 6Å after each iteration.

Target Protein	# Iterations					
	1	2	3	4	5	6
Protein A, 1FC2	94.3	98.5	100	100	100	100
Homeodomain, 1ENH	92.8	95.0	96.9	100	100	100
Protein G, 2GB1	93.4	96.4	100	100	100	100
Cro repressor, 2CRO	75.8	97.3	100	100	100	100
Protein L7/L12, 1CTF	25.6	68.8	97.0	100	100	100
Calbidin, 4ICB	46.3	90.5	99.3	100	100	100

at $(-1.86, -0.13)$ is obtained after six iterations. It is observed that there is a small gap between the center of this distribution and the native torsion angle pair $(-1.44, -0.63)$. This gap is inevitable since the standard bond lengths and bond angles are adopted in the proposed structure generating model [78].

Percentage of Good Decoys

Table 4.4 displays the percentages of the good decoys, that increase steadily at each iteration. All six proteins reach 100% of the good decoys after four iterations. In particular, the percentage of good decoys for 1CTF and 4ICB is boosted to 100% from 25.6% and 46.3%, respectively.

Quality of the Final Decoys

In FALCON, the Fragment-HMM is not only used to sample decoys but also to rank a given decoy. Each decoy from the fifth iteration is ranked according to the probability that the Fragment-HMM generates the decoy, and outputs the decoy with the highest probability as FALCON's prediction. Table 4.5 summarizes the comparison between FALCON and ROSETTA. ROSETTA's results are obtained by the clustering program of ROSETTA's package with the default configuration.

As summarized in Table 4.5, for five of the six benchmark proteins: 1FC2, 1ENH, 2CRO, 1CTF, and 4ICB, FALCON's final predictions are better than that of ROSETTA's, under the RMSD metric. For protein 2GB1, ROSETTA exhibits a better prediction than FALCON.

Further experiments are conducted on eight larger proteins with more than 100 residues. These proteins are selected from CASP7 free modeling targets (the 7-th Critical Assessment of Techniques for Protein Structure Prediction). As shown in Table 4.6, for five of the eight proteins; that is, T0283, T0350, T0354, T0361, and T0373, FALCON performs better than ROSETTA under the RMSD metric. In CASP8, FALCON ranked third in the hard target category.

Table 4.5: Quality of the final decoys of ROSETTA and FALCON for the six benchmark proteins.

Target Protein	ROSETTA	FALCON
Protein A, 1FC2	3.660	3.652
Homeodomain, 1ENH	2.717	2.464
Protein G, 2GB1	2.755	3.323
Cro repressor, 2CRO	3.997	3.477
Protein L7/L12, 1CTF	8.327	3.035
Calbidin, 4ICB	4.866	4.770

Column 2-3: RMSD ($\mathring{A}$) of the finally chosen decoys of ROSETTA and FALCON.

Table 4.6: Quality of the final decoys of ROSETTA and FALCON for eight larger proteins from CASP7 free modeling targets.

Target Protein	PDB Entry	Length	ROSETTA	FALCON
T0283	2HH6	112	11.544	11.083
T0300	2H3R	102	7.557	9.282
T0307	2H5N	133	14.822	16.343
T0350	2HC5	117	10.635	7.406
T0354	2ID1	130	11.254	8.085
T0361	2HKT	169	20.009	12.225
T0373	2HR3	147	19.097	14.224

Column 4-5: RMSD ($\mathring{A}$) of the final decoys of ROSETTA and FALCON.

4.6 Extending FALCON to Accept NMR Data

To make FALCON more useful, it is adapted to accept partial chemical shift information and contact information, as well as to tolerate errors. The modified program forms part of the pipeline of an automated system for the NMR structure determination, AMR. AMR consists of two other components: PICKY and iPass. Here, a brief description of the system is given.

4.6.1 Methods

Each sequence segment is searched in a structural database. The structural database is parsed from the 5,564 protein structures used by Shen *et al.* [134].

Three types of scores are identified between a structural fragment and a sequence segment: the chemical shift score, homology score, and mutation score.

Chemical shifts of the structural database are predicted by a modified SPARTA program [133], and (partial) chemical shifts of the target sequence are obtained by iPass. Secondary chemical shifts [44] are used. Given a sequence segment and a structural fragments, both ℓ amino acids, a chemical shift score is computed between an amino acid of the structural fragment and the corresponding amino acid of the sequence segment. At most ℓ chemical shift scores exist for each sequence segment and structural fragment pair. The chemical shift score between an amino acid of a sequence segment and an amino acid of the structural fragment is the RMSD of the corresponding chemical shifts of the atoms in the amino acids. One of three possible states is assigned for each pair of amino acids: *match, mismatch*, and *unknown*. If all the chemical shifts of atoms in an amino acid for the sequence segment are missing, the state is *unknown*. Otherwise, two amino acids are a *match*, if the chemical shift score between them is below a certain threshold, and the secondary chemical shift difference between each corresponding pair of atoms is below certain threshold. The other cases are called a *mismatch*. Structural candidates with more *matches* are preferred. If there is a tie, structural fragments with smaller chemical shifts and homology scores are used. Given a structure and a sequence fragment, details of the score function between them are as follows. Here, the homology score is as given in [44], where a substitution matrix is used.

- If a sufficient number of matches between them have been founded (i.e. above 66%), the score to measure the distance between the two fragments is a combination of the homology score and the chemical shift score.
- If most of the chemical shifts are missing and a substantial amount of chemical shift states are *unknown* (i.e below 33%), the mutation score is used to measure the similarity between two fragments. Parameters are trained according to the approach described in FRazor.
- If the number of *matches* is between 33% and 66%, a hybrid score of the chemical shift and mutation score is employed.

4.6.2 NMR Results

Although using the chemical shifts alone might be sufficient for some proteins, it is insufficient for most proteins. Previous work, first by [32, 64] and followed by [134], have demonstrated that it is possible to reliably obtain high resolution protein structures for some small proteins by using the chemical shift information (obtained from a manual full sequential assignment) alone.

FALCON calls RAPTOR [169] to do the threading. If the score is high, it starts the FALCON-NMR refinement process by using N-NOESY and chemical shift information to select the best decoys. (This step was omitted in the experiments for TM1112 and VRAR.)

Four proteins TM1112, CASKIN, VRAR and HACS are tested by both approaches: the *ab initio* approach and homologous model refinement approach.

Table 4.7: AMR final structures.

Protein	Length	RMSD
TM1112	89	1.25Å
CASKIN	67	1.89Å
VRAR	72	1.49Å
HACS	74	1.93Å

For TM1112 and VRAR, the *ab initio* approach is applied. The homologous proteins are removed from the FALCON-NMR structural database. FALCON-NMR generates decoys. They are selected by a combination score of the chemical shifts and N-NOESY contact constraints. The top decoys are fed back to FALCON-NMR for another iteration. Figure 4.5 shows the superimposed final decoy by FALCON-NMR and the native structure for TM1112. Figure 4.6 illustrates the superposition of the final decoy by FALCON-NMR, and the native structure for VRAR.

CASKIN and HACS consists of many high confidence homologous models. The direct application of RAPTOR finds the structural homologs: 2de0x, 2fpea, 1i1ja, and 1spka. By feeding these to FALCON-NMR for a further refinement round, and selecting using chemical shifts and contacts, the top decoys are 1.89Å RMSD and 1.93Å RMSD away from the NMR models, provided by Donaldson's lab, respectively. The superimpositions of the top ranked models and the superimposed structures are shown in Figure 4.7 and Figure 4.8, respectively.

4.7 Summary and Discussion

The following summarizes the results in this chapter. Based on the belief that simple frameworks and models are better than a complicated approach, in this

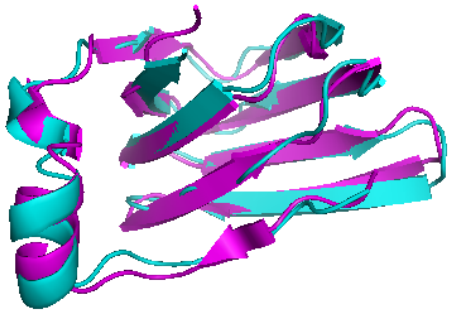


Figure 4.5: TM1112 structure by AMR, magenta, superimposed on the NMR structure, cyan, at 1.25Å RMSD.

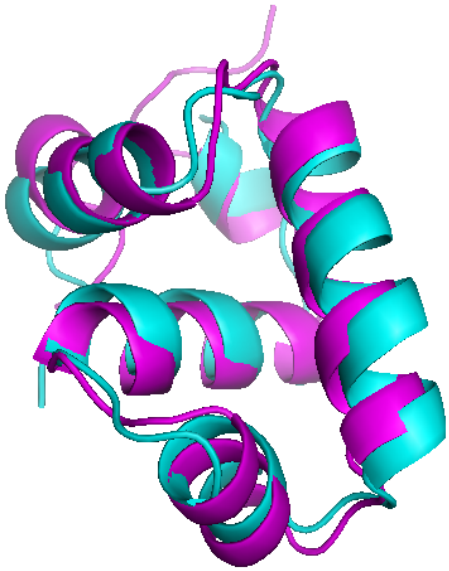


Figure 4.6: VRAR structure by AMR, magenta, superimposed on an NMR structure, cyan, at 1.48Å RMSD.

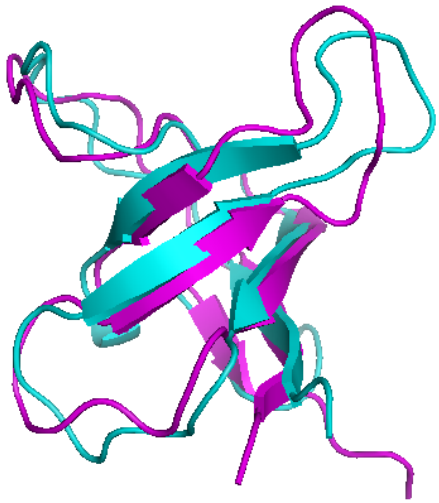


Figure 4.7: CASKIN structure by AMR, magenta, superimposed on an NMR structure, cyan, at 1.89Å RMSD.

Figure 4.8: HACS structure by AMR, magenta, superimposed on an NMR structure, cyan, at 1.93Å RMSD.

chapter a simple framework for structure sampling is proposed. It unifies the features in several approaches. The more notable ideas that the framework is based on are the fragment assembly method and HMM sampling. The proposed Fragment-HMM overcomes the difficulties of stiff structural fragments in sequence assembly approach, and the high dimensionality problem by the simple HMM approach. Experimentations using an implementation of the novel framework shows it to compare favorably with ROSETTA. With an iteration technique enabled by Fragment-HMM, the procedures of fragment assembly, clustering, and final decoy selection are unified into the framework naturally. In addition, the framework conveniently embodies other approaches such as homology modeling, threading, loop modeling, refinement, and consensus.

Finally, some remarks on the results in this chapter. In the proposed method, ideally, the quality of the decoys should converge to its native structure over iterations. However, it is noticed that, for example, the RMSD values of the decoys for protein 2CRO converge to 3Å-4Å. This is mainly due to the lack of an accurate energy function at the backbone level to direct the search process, and an all-atom energy function for a refinement process. Another reason for it would be the use of idealized bond angles and bond lengths. To solve the latter problem, it would be beneficial for the sampling program to sample different bond angles and bond lengths.

Chapter 5

Side Chain Packing

Side chain conformation prediction is important for protein structure prediction and protein structure design. The feasibility of side chain conformation prediction relies on the fact that the possible side chain conformations are limited, and there are only a few frequent patterns [50, 83, 87, 118, 165]. Following this idea, frequent side chain conformations are extracted from the known protein data bank [17] to generate rotamer libraries [50, 87, 112, 115, 118, 126, 151, 165]. The improvements of these libraries, by combining the information of the backbone dihedral angles, have resulted in backbone dependent rotamer libraries [50–52, 87].

Two families of algorithms have been designed. One family consists of exact algorithms which guarantee an optimal conformation, but at the expense of exponential time complexities. The techniques frequently employed by these algorithms include dead-end elimination [45, 47, 48, 62, 65, 89, 98, 99, 111, 124, 156], graph theory [14, 30, 167], and mathematical programming [34, 55, 91]. In contrast, the techniques in the other family are faster through the utilization of heuristics such as Monte Carlo [75,109,154], cyclical search [51,165], and meta heuristics [81,100,151].

Most side chain prediction methods employ a two-step strategy:

- first, rotamer candidates are selected from a rotamer library for each residue; and
- secondly, a searching method, along with an energy function, is used to find the optimal combination of side chain conformations for all residues.

An energy function, either statistically or physically based, serves as a core to guide side chain packing [36, 109, 119, 130, 147, 165]. However, there is a technical dilemma in designing a perfect energy function: an accurate energy function is difficult to design, and is inefficient in practice. Conversely, a simple energy function results in significant deviations in side chain packing. Therefore, though equipped with powerful methods to find global minimum, the solutions to side chain packing problem till now are not satisfactory due to the energy functions.

Inspired by the threading approaches to protein structure prediction, substructures are applied to capture the subtle energy terms implicitly. In threading approaches, the interactions among neighboring residues are captured by structures rather than by accurate energy functions. The focus is shifted to decide whether the target sequence can adopt a structure, from deciding whether a given protein sequence can be folded into some structures guided by energy functions. An alternative approach to overcome the difficulty in designing an energy function is to improve the rotamer candidate extraction. More specifically, the subtle terms of the energy function can be implicitly embedded by rotamer candidate selection. Thus, an accurate rotamer candidate selection approach can relieve or eliminate the requirements of accurate energy functions.

This strategy also follows the main stream trend in previous studies on rotamer libraries. Initially, backbone-independent libraries were used, along with complicated energy functions. Later, these libraries were improved by introducing backbone dihedral angles and simplified energy functions, giving rise to backbone-dependent rotamer libraries [30]. At this moment, the rapid accumulation of known structures makes it feasible to introduce more constraints to extract rotamers candidates. Thus, such libraries are built statistically, based on the ϕ/ψ dihedral angle and amino acid type information. In summary, with the increase of known protein structures, it is reasonable to use more constraints to build rotamer libraries and this should benefit the accuracy of side chain packing.

The intuition of our methods can be described as follows. Given two independent residues of the same amino acid type, if the conformations and the amino acid types of residues surrounding these two residues are the same, then the side chain conformations of these two residues should be similar, according to the assumption that a protein structure is stable at the lowest energy state. The amino acid types and conformations of the surrounding residues of residue R is referred to as *R's neighborhood*. Theoretically, given R's neighborhood, the side chain conformation of R can be determined. However, the intent in this chapter is not to discover how a side chain conformation is inferred from its neighborhood; but how to simplify neighborhood representation and retain sufficient information to code the side chain conformation. The motivation is to design a simple but informative neighborhood representation, such that the side chain conformation can be inferred by searching in a known database for residues with similar neighborhoods.

Moreover, the use of hexagon substructures to describe the neighborhoods of the residues is explored. Typically, a simple neighborhood cannot capture the underlying factors of side chain conformations, and a complicated neighborhood might not have matches in the known structures. As an initial attempt, hexagon substructures are proposed to capture neighborhood information. The hexagon structure can utilize the backbone information of the neighbors: if two residues of the same type shares similar hexagon structure, their side chain conformations should be similar. By using hexagon substructures, some issues in the energy function are implicitly coded, and further elimination of the ambiguities for the side chain conformations are possible. The initial experimental results indicate

that this use of hexagon substructures is reliable.

The side chain packing problem is NP-hard in the general case [7, 125], and remains NP-hard even under geometric constraints [167]. By using the hexagon substructures, the number of rotamer candidates for each residue is reduced to a small constant. However, the problem remains NP-complete even if a residue assumes, at most, three rotamer conformations.

5.1 Method

The development of the proposed method can be summarized as follows. For each residue in the given backbone structure, a hexagon substructure is built to describe its neighborhood information. Then, this hexagon structure is searched in a predefined hexagon structure database. The side chain conformations of the top K hexagon substructures from the database are extracted as predictions. The details are presented in the subsequent subsections.

5.1.1 Rotamer Database

The novel rotamer (or hexagon) database is preprocessed using a similar procedure as that in [30]. In particular, PDB20 with a resolution < 1.7Å is used. There are 3,428 chains. Suspicious residues are filtered out by previously established tools from [162, 163].

5.1.2 Hexagon Substructure

Hexagon substructures are designed to provide a concise way to describe the neighborhood information of a residue, including its nearest neighbors, and the number of residues within a given threshold distance.

The hexagon substructures are designed as follows. First, a reference system is specified by the N, C_α, and C_β atoms of each residue. It is assumed that N is on the x-y-plane (on the half plane, where x is positive), C_α is the origin, and C_β is along the y-axis in the positive direction. The entire 3D space is partitioned along the y-axis into six equal sub-spaces by three planes, and the angle between any two planes is 120°. Note that the first subspace ranges from $-30°$ to 30° with respect to the positive half of the x-y-plane, and the N atom is not on any of the subspaces' boundaries.

With this definition, it is easy to describe the relative position of the residue's neighbors in a simple but informative way. It should be noted that the reference systems are directly comparable across residues, since the coordinates of N, C_α, and C_β can be aligned approximately. In addition, the coordinates in various reference

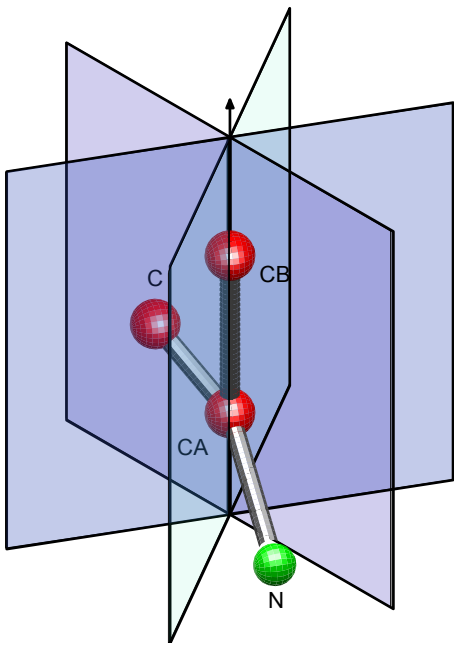


Figure 5.1: Hexagon substructure.

systems of the same amino types are directly comparable, regardless of the rotation and translation.

In each subspace, the nearest neighbors and contact number are computed as follows.

- *Nearest Neighbors*: In each subspace of residue R, the nearest nonlocal C_α to C_β of R is located. Here, a residue is *nonlocal* to R, if and only if it is separated by three residues from R in the target sequence. These six residues are *spatial neighbors* of residue R, denoted as H_R^1, H_R^2, ..., H_R^6. Also, H_R^i is used as the C_α coordinate of residue H_R^i, when the context is clear.

- *Contact Vector*: In each subspace of residue R, the *contact number*, that is, the number of C_α atoms within a certain distance, $10\mathring{A}$ in this chapter, is calculated. The six *contact numbers*, C_R^1, C_R^2, ..., C_R^6, form a *contact vector* for residue R.

The idea behind the hexagon substructures is to describe how many contacts a residue has in each subspace, and what kinds of amino acids are nearest neighbors. In addition, a hexagon substructure captures some distant contact information. By doing so, a substructure can implicitly capture long distance contact energy terms.

Additional notations are defined before describing the usage of hexagon substructures. Given a residue R, its hexagon structure is H_R. The ϕ and ψ angles of R are represented as ϕ_R and ψ_R, respectively. Two angles are said to be similar, if

and only if the difference between them is below a certain threshold. The subsequent residues of R in sequence are denoted as R^{+1}, R^{+2}, and so on. Similarly, the preceding residues of R are R^{-1}, R^{-2}, and so on. R is written as R^0 for notation simplicity. R^{-2}, R^{-1}, R^1 and R^2 are the *sequential neighbors* of R.

The objective is to investigate whether two residues with similar hexagon substructures share similar side chain conformations. The similarity between two hexagon substructures is measured by two types of distances: *geometry distance* and *sequence distance*.

5.1.3 Geometry Distance

The geometry similarity between two hexagon substructures consists of three components: the torsion angle difference, contact vector difference, and contact distance.

Dihedral Angle Difference

It has been reported that side chain conformations are highly related to backbone dihedral angles [51]. The typical example in Figure 5.2 portrays the statistical data for amino acid Phenylalanine from the training data set. In particular, given two occurrences of amino acid Phenylalanine, the backbone dihedral angle difference is measured on the plane formed by the x-axis and z-axis, while their χ_1 angles are measured with respect to the y-axis. It is observed in Figure 5.2 that the more similar the two residues' backbone dihedral angles are, the more similar their side chain conformations are. However, it should be noted that it is not necessary for two residues of identical dihedral angles to have the same side chain conformation. Also, that side chain conformation is more sensitive with respect to the changes of the ϕ angles than that of the ψ angles. The curves for the other types of amino acids are similar.

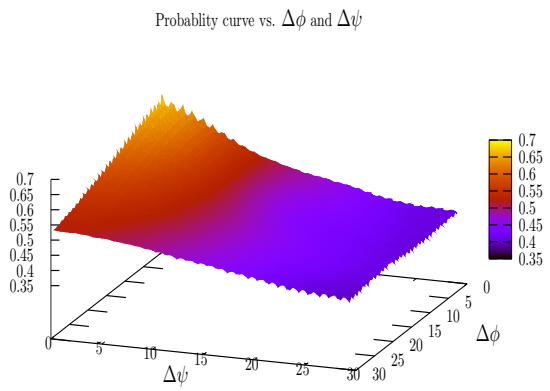


Figure 5.2: Similarity of χ_1 angles vs. the difference between ϕ and ψ for the amino acid Phenylalanine.

To quantitatively describe the relationship between χ_1 and the backbone dihedral angles, a 2D elliptical Gaussian function, shown in Equation 5.1, is applied to parameterize the curves, where λ is the normalization factor,

$$N(\Delta\phi, \Delta\psi) = \lambda \exp^{a\Delta\phi^2 + b\Delta\psi^2 + c\Delta\phi\Delta\psi + d\Delta\phi + e\Delta\psi}. \tag{5.1}$$

The parameters are determined by using the regression techniques on a training data set.

$G_T(R_1, R_2) = N(|\phi_{R_1} - \phi_{R_2}|, |\psi_{R_1} - \psi_{R_2}|)$ describes the similarity χ_1 angles for two given residues.

Besides the backbone dihedral angles of R itself, the dihedral angles information of the sequential neighbors of R is also utilized. The reason for this is that the sequential neighbors' backbone dihedral angles also affect the side chain conformation of R. The similarity of sequential neighbors' backbone dihedral angles are represented as follow. Given two residues and their sequential neighbors: $R_1^{-2}, R_1^{-1}, R_1, R_1^{+1}, R_1^{+2}$ and $R_2^{-2}, R_2^{-1}, R_2, R_2^{+1}, R_2^{+2}$, the sum of the dihedral angles difference, between the 5-mers is used as a score, and is defined as follows:

$$G_L(\Delta\phi, \Delta\psi) = \sum_{i \in -2,-1,1,2} (\phi_{R_1^i} - \phi_{R_2^i})^2 + (\psi_{R_1^i} - \psi_{R_2^i})^2. \tag{5.2}$$

The amino acid types of R_1 and R_2 are assumed to be the same.

Contact Number Difference

The contact number difference between two residues R_1 and R_2 is expressed as

$$G_C(R_1, R_2) = \sum_{i=1}^{6} |C_{R_1}^i - C_{R_2}^i|. \tag{5.3}$$

Two residues of the same amino acid type are compared only if both of the dihedral angle differences are within 10°. Otherwise, a large constant is assigned.

Spatial Neighbor Distance

As we pointed out in Section 5.1.2, the coordinates of the spatial neighbors of two residues are directly comparable. Therefore, RMSD is used as the metric here. The neighborhood distance between the two residues, R_1 and R_2 is computed by

$$G_S(R_1, R_2) = \sqrt{\frac{1}{6} \sum\nolimits_{i=1}^{6} (H_{R_1}^i - H_{R_2}^i)^2}. \tag{5.4}$$

Table 5.1: Parameters the for joint probability function for 18 amino acids.

AA	λ	$a \times 10^4$	$b \times 10^4$	$c \times 10^4$	$d \times 10^4$	$e \times 10^4$
CYS	0.648	-20.6	-116	-2.25	-2.47	2.61
ASP	0.667	-2.77	-40.5	0.411	-3.07	0.519
GLU	0.523	0.495	-131	-0.943	-1.66	6.05
PHE	0.687	-50.8	-270	-0.773	2.21	3.33
HIS	0.571	-9.75	-216	-2.06	-0.44	5.84
ILE	0.908	50	-103	-2.61	-11.8	5.37
LYS	0.525	41.8	-160	-1.23	0.0195	4.07
LEU	0.565	54.2	-63.2	-0.0727	0.984	-0.932
MET	0.574	0.437	-83	-0.913	0.774	1.65
ASN	0.678	-71.9	-73.1	0.317	-2	2.73
PRO	0.829	-331	-17.3	-3.24	0.305	-0.989
GLN	0.522	4.5	-81.8	-0.437	-1.12	3.84
ARG	0.527	0.724	-134	-0.506	0.204	2.81
SER	0.519	91.8	-134	-4.15	-6.33	6.53
THR	0.871	-14.5	-165	-0.808	-10.4	2.39
VAL	0.907	7.49	-94.3	-1.11	-12.9	4.2
TRP	0.571	51.9	-318	-2.14	0.822	5.39
TYR	0.687	-48.7	-304	-1.05	2.82	3.7

5.1.4 Sequence Comparison

In addition to the geometry comparison, the sequence similarities are also calculated. The sequence similarity consists of two components: a local sequence similarity and a spatial sequence similarity.

Local Sequence Similarity

The local sequence substitution matrices are built for the sequential neighbors of the residues. The substitution matrices are built in the same way as BLOSUM [74] matrices except for the way to "align" the sequence fragment. Specifically, a matrix is computed for each amino acid type at positions $i = -2, -1, +1, +2$, and there are 4×18 local mutation matrices, M_A^i, $-2 \leq i \leq 2, i \neq 0$.

To compute the local mutation matrices, sequences need to be aligned. Two 5-mers R_1^{-2}, ..., R_1^{+2} and R_2^{-2},... ,R_2^{+2}, are *aligned* if the following conditions are satisfied

- R_1 and R_2 are amino acids of the same type,
- $d(\phi_{R_1{}^i}, \phi_{R_2{}^i}) \leq \theta^i$ and $d(\psi_{R_1{}^i}, \psi_{R_2{}^i}) \leq \theta^i$, $-2 \leq i \leq 2$.

In this chapter, $\theta^0 = 15^\circ$ and $\theta^i = 60^\circ$ for $i \neq 0$. Two aligned 5-mers are matched, if $d(\chi_{R_1}, \chi_{R_2}) \leq \theta_\chi$.

Probability $p^i_{A,j,k}$ indicates the number of normalized mutations from amino acid j to k at position i, when R_1 and R_2 are matched for amino acid type A. Similarly, $p^i_{A,j}$ is the frequency of amino acid type j at position i, when R_1 and R_2 are matched for amino acid type A. The entry of the mutation matrix, $M_A^i[j,k]$, is

$$M_A^i[j,k] = \log \frac{p^i_{A,j,k}}{p^i_{A,j} \times p^i_{A,k}}. \tag{5.5}$$

Given the matrices, the similarity of the local sequence is computed as follows:

$$S_L(R_1, R_2) = \sum_{-2 \leq i \leq 2, i \neq 0} M_A^i[A^i_{R_1}, A^i_{R_2}]. \tag{5.6}$$

Spatial Neighborhood Sequence Similarity

Besides the local sequence similarity, the neighborhood sequence similarity is computed. By using similar methods, one matrix is computed for each amino acid type in each of the six subspaces. In total, there are 6×18 matrices, L^i_{aa}, $1 \leq i \leq 6$, aa is any amino acid except for Gly and Ala.

To compute the neighborhood mutation matrices, it is necessary to align two residues and their neighborhoods. As discussed in Section 5.1.2, a reference system is defined for each residue, R, according to its backbone atoms N, C_α, and C_β.

Two residues R_1 and R_2, along with their neighbors are aligned, if

- the amino acid types of R_1 and R_2 are the same,
- $d(H^i_{R_1}, H^i_{R_2}) \leq \theta_N$, $1 \leq i \leq 6$,
- $d(\phi_{R_1}, \phi_{R_2}) \leq \theta^0$ and $d(\psi_{R_1}, \psi_{R_2}) \leq \theta^0$.

Here, $\theta^0 = 15^\circ$ and $\theta_N = 2\mathring{A}$ are chosen. Two aligned residues and their neighbors are *matched*, if $d(\chi_{R_1}, \chi_{R_2}) \leq \theta_\chi$.

Probability $p^i_{A,j,k}$ is the frequency of mutations from j to k in subspace i when R_1 and R_2 are matched for amino acid type A, $1 \leq i \leq 6$. Probability $p^i_{A,j}$ denotes the frequency of amino acid type j at subspace i if R_1 and R_2 are matched for amino acid type A. The entry of mutation matrix $L^i_A[j,k]$ is defined as

$$L^i_A[j,k] = \log \frac{p^i_{A,j,k}}{p^i_{A,j} \times p^i_{A,k}}. \tag{5.7}$$

Given the matrices, the spatial sequence similarity is computed as follows.

$$S_N(R_1, R_2) = \sum_{1 \leq i \leq 6} L^i_A[A_{H^i_{R_1}}, A_{H^i_{R_1}}] \tag{5.8}$$

5.1.5 Identify Rotamers Candidates

Given two residues R_1 and R_2, the similarity score is computed as follows.

$$f(R_1, R_2) = \sum_{i \in \{T,N,C\}} w_i G_i(R_1, R_2) + \sum_{i \in \{L,N\}} w_i S_i(R_1, R_2). \tag{5.9}$$

Given a backbone structure, the neighborhood information and local information are extracted for each residue; and the rotamer database is searched by using the scoring function in Equation 5.9. The top 25 residues are returned as candidates for the side chain conformation. Finally, the 25 rotamers of each residue are clustered, and the presentative rotamer from each cluster is reported as final solutions.

5.2 Results

The same testing data set as in [30] is used. A prediction of the χ angle is considered correct, if the predicted value is within 40° of the actual angle. If not mentioned explicitly, the accuracy of the χ_1 angles is used.

Table 5.2: Effectiveness of different distance measures

AA	None	Geo. Dist				Seq. Dist	
		G_T	G_C	G_S	G_L	S_N	S_L
CYS	71.1	72.0	72.8	73.3	76.0	77.2	75.7
ASP	74.7	74.9	76.0	76.8	80.0	78.8	80.0
GLU	62.7	63.3	64.6	65.3	66.0	67.9	64.9
PHE	75.2	72.3	77.2	78.5	78.5	82.5	78.8
HIS	67.6	68.9	70.8	71.3	73.2	74.9	72.4
ILE	90.1	90.3	90.9	90.8	91.5	91.5	90.8
LYS	68.0	67.1	70.5	70.5	71.0	72.7	70.0
LEU	72.7	73.2	76.6	77.8	75.0	81.8	76.4
MET	67.6	69.0	68.9	70.2	71.7	72.6	69.6
ASN	70.2	70.9	71.8	72.6	76.5	74.9	76.0
PRO	94.9	95.4	96.0	95.6	96.8	96.1	96.5
GLN	64.5	65.2	66.9	67.6	69.1	71.8	68.4
ARG	65.7	66.8	67.5	68.4	69.2	71.2	68.3
SER	64.4	65.2	66.5	66.0	68.8	69.0	69.4
THR	87.3	88.5	88.9	88.8	90.1	89.9	89.2
VAL	89.2	90.0	90.2	90.0	90.6	90.8	90.0
TRP	69.8	68.9	73.0	73.1	75.1	76.5	74.2
TYR	74.9	75.3	76.3	77.9	78.1	80.5	78.0
ALL	72.9	74.3	75.8	76.3	77.1	78.6	76.9

5.2.1 Effectiveness of features

Table 5.2 displays the effectiveness of the proposed features. Both ϕ/ψ differences are restricted to within 10° when two residues are compared. As a control, the accuracy of majority voting technique is listed in the second column. The remaining columns display the accuracies for each feature independently. As seen in the table, all the features are effective with the local sequence similarity being the most effective feature.

Figure 5.3 displays the relationship between the χ_1 angles and contact vector of the subspaces for residue type Aspartate. In this example, the backbone dihedral angle variations are within, at most, 3°. This figure conveys that as the contact distance increases, the similarity of χ_1 decreases. The χ_1 angles are similar for most of the residues with the same backbone dihedral angles and contact distances. In addition, the accuracy increases, when more subspaces are adopted. It is noteworthy that similar trends can be observed for the other amino acid types.

In addition, the interest is in whether or not employing more subspaces improves side chain conformation prediction. The relationship between χ_1 and the number

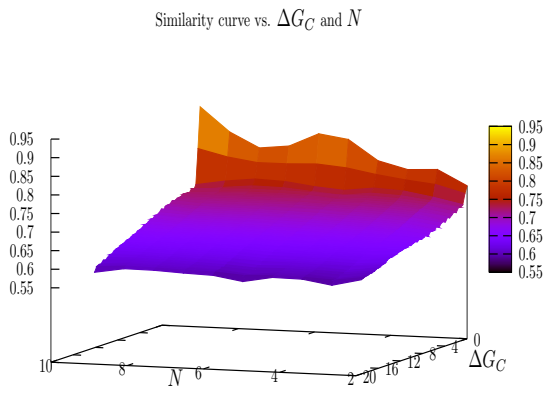


Figure 5.3: Similarity of χ_1 angles vs. the difference between ϕ and ψ for amino acid type Aspartate.

of subspaces is illustrated in Figure 5.3. It is evident that when more subspaces are used, the accuracy of the χ_1 prediction is improved. However, with the increasing number of subspaces, the number of residues with the same contact vectors decreases as well.

Figure 5.4 plots the relationship between the χ_1 angle accuracy with the spatial distance and the number of subspaces. This figure portrays that the χ_1 accuracy increases as the number of subspaces and the spatial distance increase.

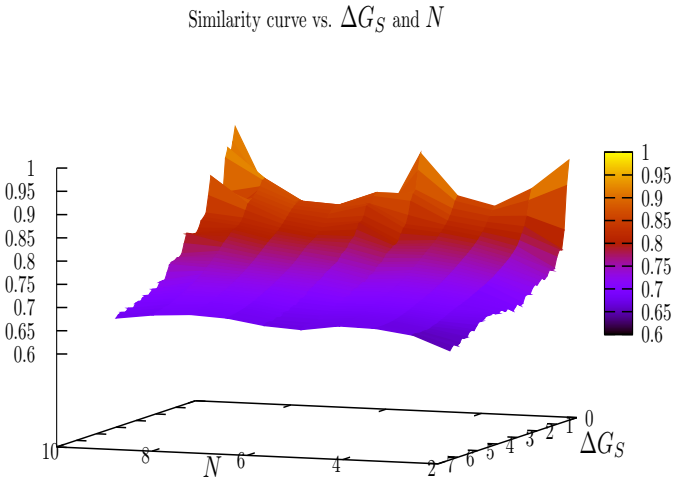


Figure 5.4: Similarity of the χ_1 angles vs. the difference between ϕ and ψ for amino acid type Phenylalanine.

5.2.2 Rotamer selection

Figure 5.4 displays the accuracy of the χ_1 angles and the average number of the rotamer candidates for each residue type. Interestingly, the accuracies of χ_1 angles

Table 5.3: Contact distance vs. number of subspaces.

	Number of Subspaces							
AA	2	3	4	5	6	7	8	9
CYS	86.0	96.7	99.1	99.8	99.8	99.8	100	99.9
ASP	76.1	82.6	86.6	88.3	89.6	91.5	91.5	94.2
GLU	56.0	62.1	68.4	70.3	71.9	75.1	74.4	82.3
PHE	77.6	90.7	96.5	98.8	99.3	99.5	99.7	99.8
HIS	85.2	97.0	98.8	99.6	99.7	99.8	99.8	99.9
ILE	95.6	97.4	98.8	99.6	99.8	99.9	99.9	99.9
LYS	59.1	66.9	71.3	76.2	74.1	78.6	81.1	80.6
LEU	62.0	74.3	84.6	93.4	92.9	95.6	95.9	97.7
MET	73.7	90.2	96.2	98.7	98.6	99.2	99.2	99.2
ASN	84.5	93.0	95.6	97.3	97.4	97.9	98.1	98.8
PRO	81.2	86.1	88.9	91.3	91.8	93.1	92.7	93.6
GLN	63.5	77.0	82.9	88.2	86.8	90.9	91.5	92.9
ARG	61.1	73.1	78.7	83.5	83.0	87.1	89.2	88.8
SER	68.3	82.4	88.2	91.3	92.1	93.4	94.0	96.4
THR	91.8	96.2	98.0	98.8	99.3	99.4	99.3	99.6
VAL	93.6	96.3	98.4	99.3	99.7	99.8	99.8	99.9
TRP	82.1	95.1	98.0	99.3	99.5	99.8	99.8	99.8
TYR	80.2	92.7	97.0	98.8	99.1	99.3	99.5	99.4

Table 5.4: Rotamer selection.

Residue Type	χ_1 Accuracy			Average # of Rotamers
	BBDEP	SCWQL3.0	HEXAGON	
CYS	80.5	88.2	86.7	2.15
ASP	71.7	80.8	83.9	2.08
GLU	61.7	71.3	70.8	2.33
PHE	75.4	93.7	89.6	1.93
HIS	67.7	85.3	86.4	2.14
ILE	87.7	91.8	93.5	1.58
LYS	66.5	74.0	75.4	2.20
LEU	72.9	89.9	84.8	1.86
MET	64.9	80.4	78.9	2.24
ASN	68.1	78.7	80.2	2.20
PRO	82.6	84.4	97.9	1.39
GLN	66.1	74.6	76.2	2.26
ARG	62.6	76.9	77.6	2.27
SER	62.6	66.4	73.3	2.42
THR	85.0	88.0	90.4	1.76
VAL	85.9	89.9	90.2	1.68
TRP	68.4	88.4	90.3	2.08
TYR	74.1	92.3	87.2	2.01
ALL	73.0	82.6	83.8	2.00

are even higher than the commonly used tool SCWQL3.0. The number of average rotamers for each residue is two.

Table 5.5 displays the predicted confidence in relation to the actual accuracies. Column one and four display the predicted confidence; column two and five display the actual accuracy above the predicted confidence; and column three and six display the percentage of residues which are predicted above the confidence. Approximately half of the residues have accuracies 93.8%, with a claimed confidence above 0.85 from the program.

5.2.3 Accuracy of Prediction

The previous experimental results suggest that it is plausible to use hexagon substructures for side chain prediction. A simple consensus is performed for rotamers reported from HEXAGON and rotamers from SCWRL for the native structural side chains. For the non-native case, rotamers from hexagon substructure strategy are used. The comparisons are listed in Table 5.6. The hexagon strategy has a higher accuracy for all the cases, and this suggests that capturing the energy terms implicitly facilitates better predictions of the side chain conformations.

Table 5.5: Predicted confidence vs. actual accuracy

	≥	real	%	≥	real	%
0.95	97.5	23.2		0.45	84.2	93.6
0.9	96.4	34.6		0.40	83.8	95.1
0.85	93.8	50.5		0.35	83.8	95.1
0.8	92.4	57.4		0.30	83.8	95.3
0.75	90.9	64.3		0.25	83.8	95.4
0.7	90.9	64.3		0.20	83.7	95.6
0.65	89.4	70.7		0.15	83.7	96.1
0.6	86.2	84.2		0.10	83.6	97.0
0.55	84.8	90.7		0.05	83.6	1
0.50	84.2	93.6		0	83.6	1

Table 5.6: Accuracy of χ_1 and χ_2 angels by a simple consensus method

Residue	Native		Non-Native	
	χ_1	χ_{1+2}	χ_1	χ_{1+2}
SCWRL3.0	82.6	73.7	53.0	33.4
HEXAGON	84.8	74.1	53.8	37.3

5.3 Complexity Results

It has been proven that the side chain packing problem is NP-complete [7, 125]. However, the proof does not reflect any geometric constraints, that is, constraints which imply that there are limits to the number of rotamers that any residue can be in contact with. When geometric constraints are imposed, a PTAS exists for the problem [167] and a new NP-complete proof is given. However, the proof placed a limit of m as the number of residues that a residue can be in contact with, without any bound on m.

It would be useful to know if the problem is efficiently solvable when m is a very small number, because it is possible to reduce the number of rotamer candidates for each residue to a very small constant, either by the dead end elimination preprocessor, or by the hexagon structures proposed in this thesis. But, the case remains NP-hard, even when each residue has, at most, three rotamers.

5.3.1 Reduction

The one-in-three planar 3SAT problem is reduced to this problem. An instance of planar 3SAT is shown in Figure 5.5. Given a three conjunction norm form (3-CNF) of a formula Φ, a bipartite graph, $G_\Phi(V_C, V_X, E)$, is established. Vertex v_x is defined for variable x, and vertex v_c is defined for each clause c. An edge is created between two vertices v_x and v_c, $v_x \in V_X$, $v_c \in V_C$ if and only if x appears in the clause c. The set of edges is represented by E. Φ is planar, if and only if G_Φ is planar. Planar 3SAT is NP-complete [110]. Knuth and Raghunathan [92] suggested an NP-complete case where G_Φ is further restricted, as demonstrated in Figure 5.5: all the variables reside on a straight line, and the three-legged clauses are arranged below and above them. The legs are in a rectilinear fashion. In [121], it has been shown that PLANAR 1-in-3 SAT is NP-complete even for the embedding in Figure 5.5.

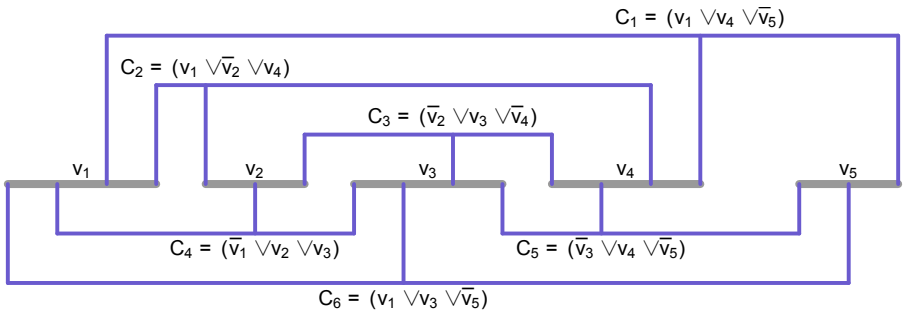


Figure 5.5: Instance of planar 3SAT.

It is clear that a planar 3SAT can be embedded on a grid. The planar 3SAT drawing is modified, as shown in Fig. 5.6. Two consecutive literals of the same

variable are two units apart. Any two horizonal lines of any two legs are at least two units apart.

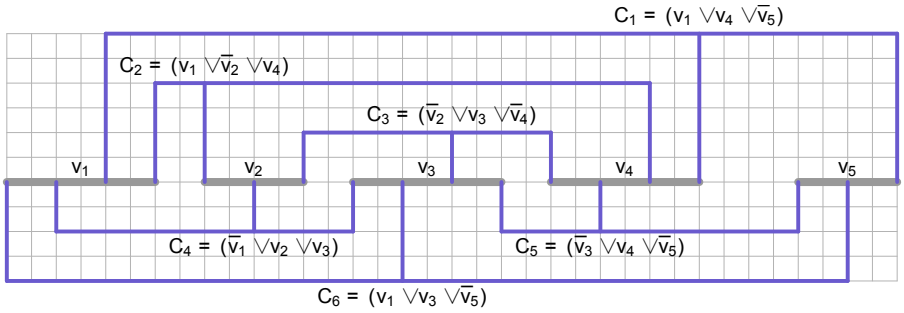


Figure 5.6: Embedding planar 3SAT on a grid

The intersection point of the three legs is called a pivot.

A residue is created for each grid point, which resides on the legs and on the line segments for the variables. In the construction, three types of residues $\{A^l, A^c, A^v\}$ are used. All the residues on a leg, except those for the pivots are of type A^l; and all the residues for pivots are of type A^p. All the residues on the line segment of variables, except those for leg terminals, are of type A^v.

Two types of rotamers $\{t_t^l, t_f^l\}$ are defined for residue type A_l. Three types of rotamers $\{t_1^c, t_2^c, t_3^c\}$ are associated with residue type A^c. Two types of rotamers $\{t_t^v, t_f^v\}$ are defined for residue type A^v.

An interaction edge is created between any two residues, if the distance between them is no more than one. It is assumed that there are N residues $r_1, ..., r_N$.

An assignment of side chain is denoted as $\mathcal{A}$. To reduce the 3SAT problem to the side chain packing problem, an energy function with all the possible assignments as the domain is designed. The energy score evaluates to zero on any optimal assignment, and positive values on all other assignments.

If two residues do not share an interaction edge, the score between them is zero. For two residues r_i and r_j that share an interaction edge, the scores are summarized in Table 5.7. For two residues within the same leg which interact, the score between them is zero if they adopt the same type of rotamers, and one otherwise. Ideally, if variable x is assigned to a true value, all the residues in the legs, corresponding to non-negated literals x, adopt rotamers A_t^l, and all the residues in the legs, corresponding to negated literals $\bar{x}$, adopt rotamers A_f^l. Similarly, if variable x is false, all the residues in the legs, corresponding to non-negated literals x, adopt rotamers A_f^l, and all the residues in the legs, corresponding to negated literals, $\bar{x}$ adopt rotamers A_t^l.

A pivot has three rotamers. If the left leg is for a true value, and the other two legs are for false values, then the pivot residue has a rotamer, t_1^p. Similarly, the scores are defined for the other two cases: where the middle leg is true, and where

Table 5.7: Energy values for the side chain packing reduction

r_i			$\mathcal{A}(r_i)$, r_i of type A^l			
			Non-negated Leg		Negated Leg	
	Rotamer		t_t^l	t_f^l	t_t^l	t_f^l
A^c	left	t_1^c	0	1	0	1
		t_2^c or t_3^c	1	0	1	0
	middle	t_2^c	0	1	0	1
		t_1^c or t_3^c	1	0	1	0
	right	t_3^c	0	1	0	1
		t_1^c or t_2^c	1	0	1	0
A^l		t_t^l	0	1	1	0
		t_f^l	1	0	0	1
A^v		t_t^v	1	0	0	1
		t_f^v	0	1	1	0

the right leg is true. The details are given in Table 5.7. This ensures that when the score is zero, only one of the literals, x and $\bar{x}$, can be true.

Two residues r_i and r_j of types A^l and A^v interact only at the line segments of the same variable. If variable x adopts a true value, the residues of type A^v for x adopts rotamer t_t^v. If variable x adopts a false value, the residues of type A^v for x adopts rotamer t_f^v, ideally. The scores are defined accordingly.

The energy function is designed to restrict all the residues on the same leg to adopt the same type of rotamer, by taking on non-zero values when this is not fulfilled. The rotamers of the pivots ensure that only one of the legs adopts rotamer type A_t^l, and the other two legs adopt rotamer type A_f^l, to ensure the total energy is zero.

If the total energy is zero, all the residues for the same variable of type A^v on the line segment must adopt the same type of rotamer. If not, there exists a residue on a leg's terminal, whose two neighbor residues on the line segment have different types of rotamers, and the total energy would be non-zero. If the residue variable adopts rotamers of type t_t^v, the corresponding variable is assigned a true value; otherwise, a false value.

An example of such assignments is provided in Figure 5.7. The arrow which points to the pivot indicates that the residue adopts rotamer A_f^l and the arrow which points away from the pivot indicates that the literal adopts rotamer A_f^l. If the arrows on a leg point to the pivot, it indicates that the corresponding literal is assigned a true value. Otherwise, the literal is assigned a false value. The leg that the arrow of a pivot points to indicates that the corresponding literal of this leg is true, and the corresponding two literals are false. If the arrow of the line segments points to the left, the variable is assigned true. Otherwise, the arrows indicate that

the variable is assigned false.

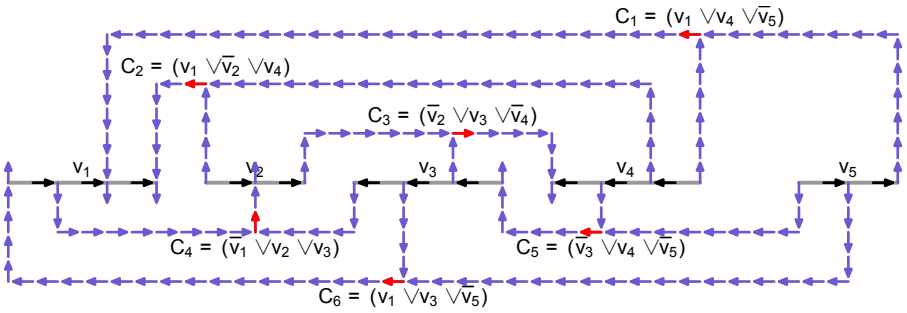


Figure 5.7: Assignment of planar 3SAT

The formal proof of the correctness of the reduction is clear. Details are omitted here.

Theorem 5. *Geometric side chain packing problem is NP-complete, even for the case that the energy items are binary and each residue have at most three rotamer candidates.*

5.4 Summary and Discussion

In this chapter it is found that hexagon substructures are useful for rotamer candidate extraction. It is possible to reduce the number of rotamer candidates to a small constant with such substructures. Moreover, they are useful for both native and non-native cases.

In the current definition of hexagon substructure, the side chain conformation of neighbors are not considered. For about 50% of the side chain conformations, their accuracy is over 93%. This information can be utilized for further side chain packing.

The accuracy for the non-native structures is even lower than the accuracy of directly applying a backbone rotamer library to the native structures. This implies that the side chain conformations are very sensitive to the backbone dihedral angles. In other words, a small change in the backbone dihedral angles can introduce a large change in side chain conformation. This drawback can undermine the usage of rotamer libraries for protein structure predictions, but side chain conformation prediction methods should be able tolerate such changes in the backbone.

In contrast, in the near-native structures, although the backbone dihedral angles are modified, the hexagon substructures are still similar. Therefore, by using the hexagon substructures to capture the rotamer conformations, higher accuracies should be achievable for near-native backbone substructures.

Chapter 6

Decoy Delection

The FALCON system in Chapter 4 rebuilds the HMM model using the decoys generated in the previous iteration. Given that very numerous decoys are outputted in an iteration, it is necessary to select only a subset of decoys with better qualities. The decoy selection problem has been intensively studied, and the common approach is to select decoys that have more similar decoys in the set. In this chapter, this common approach to decoy selection is improved in two aspects: a faster implementation and a justification for its use.

6.1 Decoy Selection Methods

Most methods for *ab initio* protein structure prediction methods adopt a "generating-selecting" paradigm. More precisely, by using a fragment-assembly [141] or other techniques, a set of decoys are generated for the given protein. Then, a selection procedure is employed to select the most native-like structure from these decoys. A reliable discrimination of native-like structures from a set of candidate decoys is crucial to protein structure prediction.

A variety of methods have been developed to perform decoy selection. These methods can be grouped roughly into two families: single-model methods, and model-ensemble methods.

Typically, a single-model method calculates the scores for each model, independently, by using physics-based energy functions [25, 76, 93, 177] or knowledge-based scoring functions [68, 84, 117, 132, 141, 148, 166]. Usually, a physics-based function uses features such as electrostatic, van der Waals force, hydrogen bond, solvation accessibility, contact potential, and hydrophobic folding energy. Unlike physics-based functions, a knowledge-based function takes into account the statistics on the interactions between the residues or atoms from known protein structures. Compared with physics-based functions, knowledge-based functions are generally simpler and easier to use, and their performances are comparable to those of physics-based functions, according to an examination of CASP results. To assign optimal weights

for the adopted features, both of these two types of functions employ machine learning techniques, say, support vector regression [123, 166]. Other methods utilize residue environmental information [113], non-bonded atomic interaction patterns [41], intra-molecular pairwise interactions [146], atom-atom contacts [157], torsion angle potential [150], and stereo-chemistry information [79, 97]. One weakness of the single-model approach is that a slight change in the backbone can result in a significant change in the scores.

In contrast, the ensemble-based methods rely heavily on the hypothesis that if a structure sampling method generates two protein structures that are close to each other, both of these structures are likely to be native-like. Such ensemble-based methods [21,136,139,159] take advantage of the relationships among decoys, rather than measuring each decoy individually.

One ensemble-based method is to assume that the decoys consist of a dominating cluster with a native-like structure inside, and the native-like structure has more neighbors than other decoys. Based on this assumption, it is reasonable to treat the decoy with the largest number of neighbors as the most native-like decoy in the decoy set. These techniques are referred to as *maximum neighbor* methods. Thus, after the clustering techniques are employed to identify the largest single cluster, the cluster center is reported as the best decoy. An alternative technique is to calculate the density around each decoy in the decoy set [159]; that is, the sum of RMSDs between a decoy and all the other decoys, and to output the decoy with the highest density as a solution.

The work in this chapter is concerned with the maximum neighbor method. It is improved in two aspects: new strategies to speed up a common implementation, and an attempt at giving a rationale for its use.

At present, there are a few tools to implement the maximum neighbor method [106, 141, 175]. In the popular protein structure prediction systems I-TASSER [164, 175] and ROSETTA [140], the decoys are selected by the following procedure: Starting with the set of generated decoys, a threshold d is decided. Then, from the set, the decoy with the maximum number of neighboring decoys within RMSD d is found, and is reported as the highest ranking decoy. (The ties are broken arbitrarily.) This decoy and all of its neighbors (the first cluster) are then removed from the set, after which the decoy, with the most neighbors within RMSD d, is again found. This decoy is reported as the second highest ranking decoy, and together with all its neighbors (the second cluster) are removed from the set. Similarly, the third highest ranking decoy is then found, and so on.

All the implementations of this procedure, at present, evaluate the pairwise RMSD (or approximate values) of the decoys, resulting in runtimes which are, at best, quadratic in the number of decoys. As the number of decoys grows to tens of thousands, this method becomes infeasible, necessitating the development of faster methods.

6.2 Faster Clustering

Three strategies are developed to speed up the clustering. In the first strategy, auxiliary groups of proximate decoys are created. This facilitates the decision of whether a group of decoys is (or is not) within the threshold distance from a given decoy, through the use of triangular inequality. The second strategy is to use efficiently computable lower and upper bounds of the RMSD to preliminary screen out unlikely candidates. Thirdly, outlier decoys can be detected and removed prior to the clustering. These strategies are implemented in an open source tool called Calibur.

6.2.1 Strategy 1: Auxiliary Grouping of Decoys

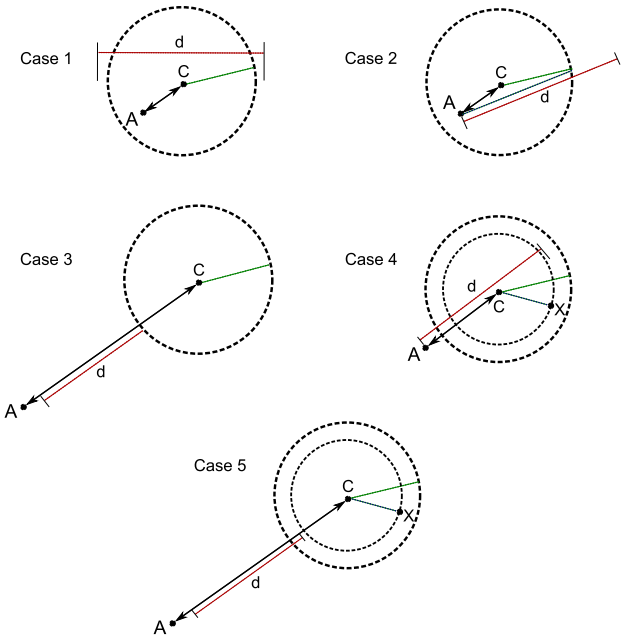


Figure 6.1: Using auxiliary grouping of decoys.

To avoid the pairwise RMSD computation, proximate decoys can be considered collectively; that is, decoys in close proximity can be grouped and represented with a single decoy C, such that if C is within a certain distance from a decoy X, the entire group is within threshold distance d from X. Similarly, no decoys in the group are within distance d from X if X is further than some distance from C.

The desired grouping is one where each decoy belongs to exactly one group, and each decoy is at most RMSD r from the group's *center* (i.e., the representative decoy). This can be accomplished as follows. First, a distance r less than d is decided, and an arbitrary decoy is set as the center. (For the purpose of Case 1 below, $r \leq \frac{d}{2}$ is used.) Repeatedly for each unclassified decoy, a search is performed

on all current centers, for one which the decoy is within distance r from. If and when such a center C is found, the decoy is grouped with C and its distance to C is recorded. Otherwise, the decoy is declared as a new center.

To locate the decoys in a group that are within distance d from decoy A, the following five cases are examine (C denotes the group's center and X denotes an arbitrary decoy in the group).

Case 1: A is in the group of C (including where A is the group's center), given that $r \leq \frac{d}{2}$.

Case 2: $\text{RMSD}(A, C) + r \leq d$.

Case 3: $\text{RMSD}(A, C) > d + r$.

Case 4: $\text{RMSD}(A, C) + \text{RMSD}(C, X) \leq d$

Case 5: $|\text{RMSD}(A, C) - \text{RMSD}(C, X)| > d$

These cases are depicted in Figure 6.1. Since RMSD is a metric [22], triangular inequality applies. Hence, in Cases 1 and 2, all the decoys grouped with C must be within distance d from A. In Case 3, the converse is true.

In Cases 4 and 5, the distances from the group's center to each member of the group has already been computed during the auxiliary grouping. Again, triangular inequality implies that in Case 4, decoy X is within distance d from A, whereas in Case 5, the converse is true. The RMSD between X and A is computed, if and only if none of the cases apply.

6.2.2 Strategy 2: Lower and Upper Bounds of RMSD

Given decoys X and Y, a lower bound of $\text{RMSD}(X, Y)$ can be used to detect if $\text{RMSD}(X, Y)$ is larger than the given threshold d. Likewise, an upper bound can be used to detect the case where $\text{RMSD}(X, Y)$ is smaller than d. The strategy is to use multiple such efficiently computable bounds as preliminary checks to reduce the much more expensive RMSD computations. These checks can be applied to the conditions in Cases 2 and 3 of Strategy 1, as well.

First, given three decoys O, X, and Y, by triangular inequality, the following holds.

$$\text{RMSD}(X, O) + \text{RMSD}(Y, O) > \text{RMSD}(X, Y) \text{ and}$$
$$|\text{RMSD}(X, O) - \text{RMSD}(Y, O)| \leq \text{RMSD}(X, Y).$$

Consequently, an upper and a lower bound of RMSD(X,Y) can be efficiently computed, through an arbitrarily chosen *reference decoy* O and pre-computed $\text{RMSD}(X, O)$ values for each decoy X. In practice, one can use c reference decoys O_1, O_2, ..., O_c to obtain c upper bounds and c lower bounds. $c = 6$ is used in this thesis.

The Euclidean distance between two decoys, after they are re-orientated to minimize their RMSDs to a fixed arbitrary decoy, yields another upper bound to their RMSD [106]. This upper bound distance is referred to as rRMSD.

Another lower bound is obtained as follows. Denote the centroid of a protein structure S_x as c_x. The *signature* Sig_x for a protein structure, $S_x = (s_{x,1}, s_{x,2}, ..., s_{x,n})$, is defined as

$$Sig_x = \langle v_{x,1}, v_{x,2}, \ldots, v_{x,n} \rangle, \tag{6.1}$$

where $v_{x,i} = ||s_{x,i} - c_x||$, $1 \leq i \leq n$. The distance between the two signatures, Sig_1 and Sig_2, called the *signature distance*, is defined as:

$$dist(Sig_1, Sig_2) = \frac{1}{\sqrt{n}} \sqrt{\sum_{j=1}^{n} (v_{1,j} - v_{2,j})^2}. \tag{6.2}$$

The signature distance between two protein structures is a lower bound of their RMSD, expressed as follows:

Lemma 6. *RMSD*$(S_1, S_2) \geq dist(Sig_1, Sig_2)$

Proof. The optimal rotation and translation found in computing the RMSD of two structures, S_1 and S_2, are denoted as R and T. The distance between $s_{1,k}$ and $s_{2,k}$ in the optimal superposition is $r_k = ||Rs_{1,k} - s_{2,k} - T||^2$. The line segments $u_{1,k}$ and $u_{2,k}$ are defined as $u_{1,k} = \langle s_{1,k}, c_1 \rangle$ and $u_{2,k} = \langle s_{2,k}, c_2 \rangle$, $1 \leq k \leq n$. The lengths of $u_{1,k}$ and $u_{2,k}$ are denoted as $v_{1,k}$ and $v_{2,k}$, respectively.

It is known that the superposition in computing the RMSD of any two structures results in the centroids of the structures to coincide [13].

The angle between $u_{1,k}$ and $u_{2,k}$ is denoted as θ. By trigonometry, $r_k = v_{1,k}^2 + v_{2,k}^2 - 2v_{1,k}v_{2,k}\cos\theta \geq (v_{1,k} - v_{2,k})^2$. Hence, RMSD$(S_1, S_2) = \frac{1}{\sqrt{n}}\sqrt{(r_1 + \ldots + r_n)} \geq \frac{1}{\sqrt{n}}\sqrt{((v_{1,1} - v_{2,1})^2 + \ldots + (v_{1,n} - v_{2,n})^2)} \geq dist(Sig_1, Sig_2)$. □

6.2.3 Strategy 3: Filtering Outlier Decoys

Another possible enhancement of the performance is to discard decoys with low similarity to the other decoys in the set, prior to the clustering. Here, an efficient technique is proposed to quickly identify such decoys. The aim is to retain all of the high ranking decoys, and the decoys which are within distance d from them. These are the "good" decoys. It is reasonably to assume that every high ranking decoy is within distance d from 10% of all the decoys. For a random sample of n decoys, the probability for a good decoy to be within distance $2d$ from, at least, one of the sampled decoys is $1 - 0.9^n$, which is > 0.9999, when $n = 100$. Hence, decoys, which are not within $2d$ from any one of 100 randomly sampled decoys, are likely bad, and are removed from the set.

6.2.4 Overall Program Design

A program based on the three strategies is designed. On a given set S of n input decoys, the program does the following.

Step 1: Discover a suitable threshold distance d for clustering S.

Step 2: Filter the outlier decoys by using 100 randomly selected decoys, as in Strategy 3.

Step 3: Create auxiliary groups for the decoys as required by Strategy 1. Compute the signature (Sig_x), and the distances (RMSD(X, O) for each decoy X and reference decoy O; rRMSD(X, Y) for each decoy X, Y) as required by Strategy 2.

Step 4: Find, for each decoy A, a neighbor set N_A which contains all the decoys in S within distance d from A (A inclusive), by using Strategy 1 and 2.

Step 5: Start with an empty sequence *Output*. Repeatedly find $A \in S$ with the largest N_A, appending A to *Output* while removing N_A from S and all the neighbor sets.

Step 6: Output the decoys in *Output*. (For brevity the program is set to output only the first three decoys.)

The threshold selection in Step 1 is addressed in the next subsection.

Steps 2 and 3 are performed straightforwardly.

Step 4 is implemented with the use of Strategy 1 and 2; that is, for each auxiliary group, the elements' membership in N_X is decided for each decoy, $X \in S$ with the aid of the strategies.

Step 5 is performed by repeating the following until S is empty: Find the decoy $X \in S$ with the largest N_X (breaking ties arbitrarily), and append the decoy to *Output*. Then, remove N_X from S, and for each $Y \in N_X$, remove Y from N_Z for each $Z \in N_Y$.

Selection of a Suitable Threshold

Two decoys are considered to be significantly related, if and only if their RMSD is *relatively small* among all pairwise RMSDs of the decoys. Hence, the strategy adopted here for threshold finding is to identify the value d such that only x percent of pairwise RMSD distances are below d, for some suitable x. Given x, a straightforward way to determine such a d exactly is to compute all $n \times n$ RMSDs, and find the $\lceil (0.01xn^2) \rceil$-th shortest distance. Alternatively, a reasonable approximation to

the x-percentile value can be obtained efficiently by using only a relatively small random sample of the decoys. In the tests, approximately ten samplings, each of 100 decoys, sufficed to determine this value quickly and consistently. The new program uses this method by default with x set to $100n^{-1/4}$. The expression $100n^{-1/4}$ is heuristic. It's aim is to reduce the percentile when more decoys are available, in order to speed up the clustering (e.g., $x = 10$ when $n = 10000$, $x = 5$ when $n = 160000$).

A similar strategy is to use the most frequently occurring RMSD among decoys, say f, as a reference to decide threshold distance d. (If the pairwise distances are distributed normally, f should correspond to the 50th percentile.) As a selectable option, the program includes a simple method, based on this, in which $d = cf + b$, where c is set to $\frac{2}{3}$, and b is set to the minimum pairwise distance, discovered through random sampling.

Memory Usage

In Steps 1, 2, 3, and 5, the memory required is linear in n. For Step 4, in the theoretical worst case, $|N_X| = n$ for each X, results in an $O(n^2)$ memory usage. However, such a case is unlikely to occur in the program's intended use. In practical use, $|N_X|$ is seldom above $0.2n$, and is small for most Xs. In the case that the number of neighbor sets falls off geometrically with the size, the memory required to store all the neighbor sets would be linear in n. In the tests, the actual growth in memory usage is closer to $O(n)$ than $O(n^2)$.

If one is interested in only the highest ranked decoy from the clustering, it is unnecessary to construct the neighbor sets, since the sizes of the neighbor sets suffice to determine such a decoy. In this case, the total memory usage is linear to n. This mode of operation is an included option.

6.3 Evaluation of Calibur

The program is implemented in C++ and called Calibur. Calibur accepts, as input, a list of names of PDB files (each for a decoy) and an optional threshold d. No pre-processing is required of the PDB files. If no threshold is given, Calibur automatically finds a suitable threshold for the input decoys. The method which Calibur uses for threshold discovery can be altered through command-line arguments. A list of all the implemented methods is shown, when Calibur is called without any input arguments.

6.3.1 Effectiveness of Strategies

The effectiveness of each strategy is evaluated with decoys predicted by FALCON on the proteins TM1112 from the Arrowsmith Lab at University of Toronto (herein

the set is referred to as TM1112) and SH3 from Donaldson's Lab at York University (herein referred to as CASKIN). Each of these two sets contains 9,999 decoys.

Auxiliary Grouping, Lower and Upper Bounds

Each case contributed in reducing the runtime, although the amounts differed at different thresholds (see Figures 6.2 and 6.3). At low thresholds, the chances of decoys being further than the threshold distance are high. Hence, Case 3 and the lower bounds are more effective. For a similar reason, the effects of Case 1 and the upper bounds become elevated at larger thresholds.

In Calibur, the order in which the cases are performed, as well as the range of thresholds is optimized according to these observations.

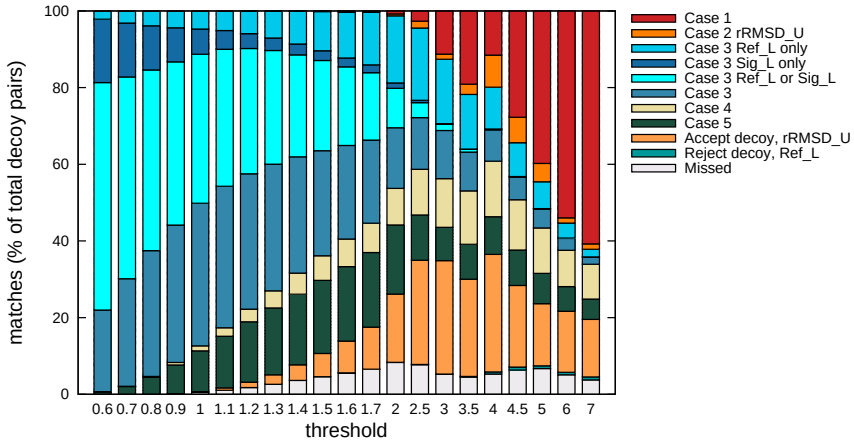


Figure 6.2: Number of RMSD computations avoided.

The number of RMSD computations avoided (percentage over 9999×9998 computations) due to each of the cases considered, at different threshold values. For Case 2, the upperbounds are used for condition evaluation prior to the actual RMSD. The contribution from the upperbounds via the reference decoys can be completely accounted for by rRMSD, and the RMSD evaluations contributed insignificantly. Only the contribution from rRMSD (label "Case 2 rRMSD_U") is shown. For Case 3, the lowerbounds are used for condition evaluation prior to the actual RMSD. Although the contributions from both kinds of lowerbounds overlap (label "Case 3 Ref_L or Sig_L"), there were contributions, entirely due to the signatures (label "Case 3 Sig_L only"), as well as the reference decoys (label "Case 3 Ref_L only"). The contribution from evaluating the actual RMSD are highly significant as well (label "Case 3"). In evaluating individual decoys, the upperbound obtained from rRMSD was highly effective at high thresholds (label "Accept decoy, rRMSD_U"). The lowerbounds from the reference decoys demonstrated noticeable effects (label "Reject decoy, Ref_L"). Other contributions are insignificant.

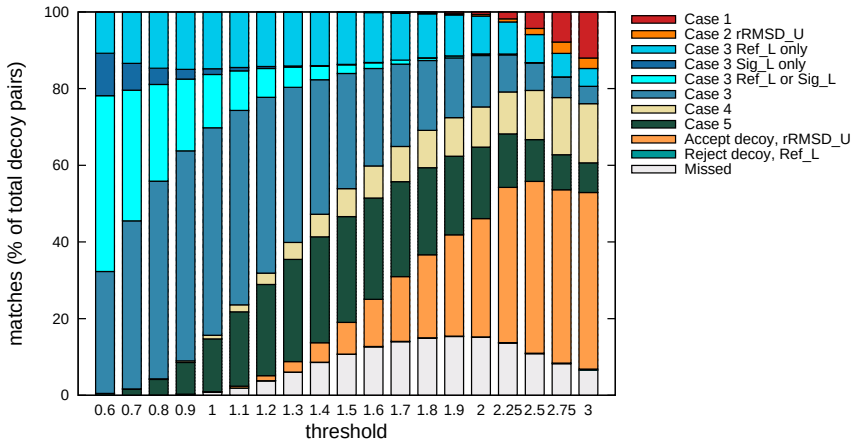


Figure 6.3: Same as Figure 6.2, but on the CASKIN data set.

Filtering

On the data sets, TM1112 and CASKIN, filtering does not affect the clusters formed by the highest ranking decoys. Their rankings remain the same. This is true even in the cases where more than 70% of decoys are filtered prior to the clustering. Figure 6.4 and 6.5 show, for TM1112 and CASKIN respectively, the number of decoys filtered (for a total of 9999 decoys) at various threshold values.

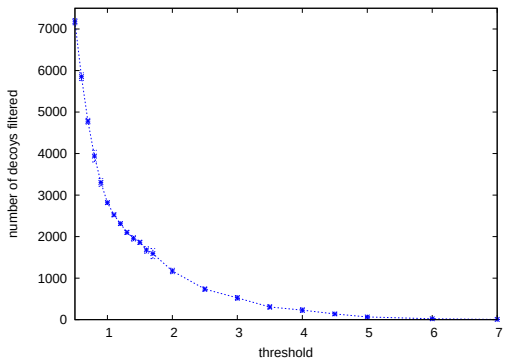


Figure 6.4: The number of decoys filtered from the set TM1112 by using 100 randomly selected decoys at different thresholds. Each value is an average of ten numbers from ten different trials by using the same threshold. The error bars show the standard deviations.

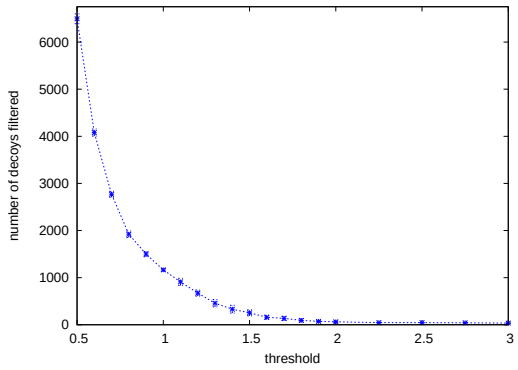


Figure 6.5: Same as Figure 4, but with the CASKIN data set.

Strategies' Effects on Calibur's Performance

To evaluate the strategies' effects on Calibur at various thresholds, the runtimes, when the strategies are used ("Calibur") and when they are not used ("*pairwise*") are compared. For reference purposes, the runtime for ROSETTA's pairwise evaluation, based on the clustering program ("cluster_info_silent") is also shown. All the tests are run on a 3GHz Intel Core 2 Duo PC with 2.98GB RAM running CentOS 5.3. All three tools are compiled by using GCC 4.1.2 with the optimization `-O`. The same codes is used for computing RMSD.

All the tools are given input such that the output is exactly the same. As a result, only their runtimes are compared. For *pairwise* and cluster_info_silent, the CPU time is taken to be the total time needed for finding neighbors, and the recursive search for the largest clusters. For Calibur, the CPU time is the sum of the times taken for the signature computation, decoys re-orientation, filtering, auxiliary grouping, finding neighbors, and the recursive search for the largest clusters. Figure 6.6 illustrates the results on the data set, TM1112. The largest sizes of the clusters at the thresholds 1, 2, 3, 4, 5, 6, and 7 are respectively 1,796, 6,017, 7,744, 8,186, 8,671, 9,120, and 9,368.

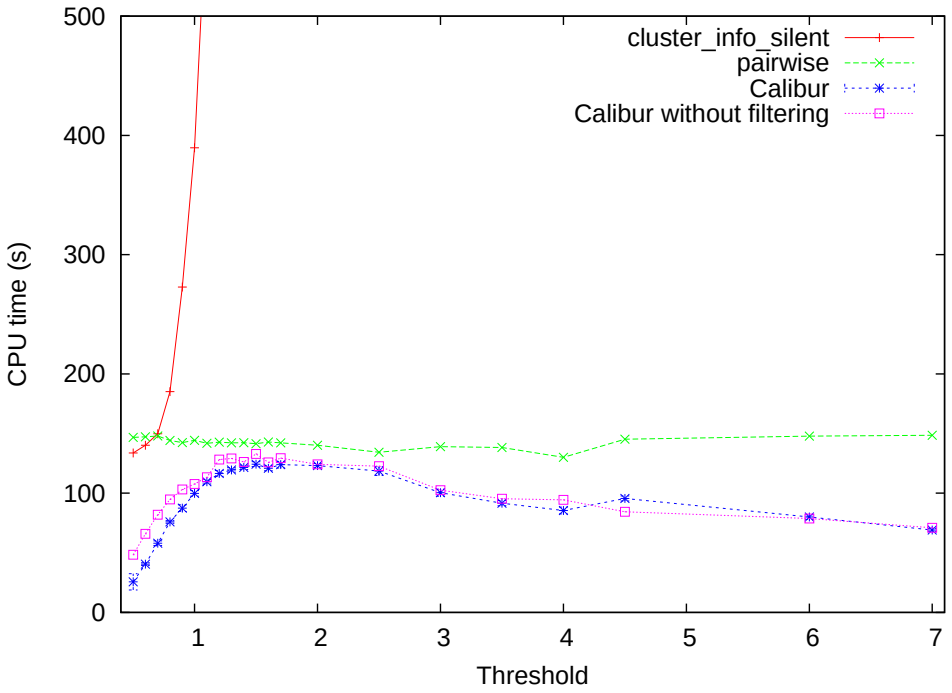


Figure 6.6: CPU times used to obtain clusters at different thresholds on the TM1112 data set of 9,999 decoys, by (1) cluster_info_silent (label "cluster_info_silent"), (2) Calibur without using any of the strategies (label "*pairwise*"), (3) Calibur (label "Calibur") (To account for variations caused by the filtering, each point is the average of 10 trials), (4) Calibur with the filtering mechanism disabled (label "Calibur without filtering")

6.3.2 Calibur's Performance on a Large Data Set

Calibur's performance in handling large numbers of decoys is evaluated using a set of 29,770 decoys for the TM1112 protein generated by FALCON. For each threshold in 0.5, 1, 1.5, 2, 3, 4, and 5, ten trial runs are performed over a UNIX cluster. (More precisely, an HP XC cluster with 378 nodes, each with 8×Xeon 3.0 GHz CPUs and 16GB memory, running RHEL 5.1.) All the runs result in the same decoy clusters. Table 1 lists the average CPU times (in sec).

Table 6.1: CPU times.

Threshold	0.5 (27)	1.0 (1966)	1.5 (5531)	2.0 (8560)	3.0 (14397)	4.0 (17915)	5.0 (19905)
Calibur	74 ± 7	506 ± 14	1047 ± 27	1482 ± 42	2369 ± 154	3109 ± 266	3616 ± 290
no filtering	225 ± 11	717 ± 22	1250 ± 35	1629 ± 42	2495 ± 180	3166 ± 233	3501 ± 272
pairwise	2628 ± 72	2624 ± 69	2651 ± 66	2741 ± 83	3293 ± 205	4014 ± 130	4425 ± 324

Numbers in brackets are the sizes of the largest clusters at the corresponding threshold.
CPU times of (1) Calibur, (2) Calibur with the filtering mechanism disabled ("no filtering"), and (3) *pairwise*, on a set of 29,770 decoys for the TM1112 protein, shown with ± standard deviation.

In practice, the largest clusters contain around 10% of the decoys. In the present case, the largest clusters found at 1.5 threshold distance already contain more than 18% of all the decoys. At this point, the corresponding CPU time, required by Calibur is about one third of the time required when the strategies are not used.

As a further reference on Calibur's performance in high load use, Calibur completed within 15,000 seconds CPU time for 100,000 decoys under its default settings.

6.3.3 Evaluation of Calibur's Output Decoys

To evaluate the decoys produced by Calibur, they are compared with those by using SPICKER [175], the clustering tool used in the leading *ab initio* protein structure prediction system I-TASSER [164]. The decoy sets, natives, and SPICKER's results found on I-TASSER's website [2] (downloaded on the July 24, 2009) are used. The data consists of decoys for 56 targets.

In order to compare Calibur with SPICKER in terms of both output and speed, Calibur is run under the same conditions as SPICKER. Both programs are compiled with optimization `-O3`, and up to 13,000 decoys are sampled from each decoy set (using the same selection procedure as in SPICKER's source codes). All the tests are performed on the same UNIX cluster. Calibur is set to use its default method for automatic threshold distance discovery. Table 6.2 indicates the TM-scores and RMSDs (to native) for the first decoys reported in each case.

The decoys of both tools are comparable. The total TM-score for SPICKER's decoys is 32.936, while that for Calibur's is 33.191. In 20 cases, SPICKER's decoys have better TM-scores. In contrast, Calibur's decoys have better TM-score in 33 cases. With respect to RMSD, SPICKER's decoys have smaller values in 23 cases,

Table 6.2: TM-scores (to native), RMSDs (to native), and CPU times (sec) for SPICKER and Calibur.

Target	Sample size	TM-score		RMSD		CPU Time		Calibur reported decoy
		SPICKER	Calibur	SPICKER	Calibur	SPICKER	Calibur	
1abv_	12,500	0.2903	0.2868	13.941	13.05	321.61	55.23	d12468.pdb
1af7__	12,499	0.4654	0.5125	4.728	4.117	242.64	67.82	d11483.pdb
1ah9_	13,000	0.5082	0.6313	4.701	3.341	346.17	199.14	d23158.pdb
1aoy_	13,000	0.6598	0.6598	4.761	4.761	365.77	295.87	d16717.pdb
1b4bA	12,500	0.4099	0.484	6.578	5.571	326.79	121.13	d808.pdb
1b72A	12,499	0.6651	0.6722	3.195	3.233	285.41	83.01	d1842.pdb
1bm8_	13,000	0.4317	0.3627	7.384	6.842	344.02	330.32	d16762.pdb
1bq9A	13,000	0.3781	0.3696	7.838	8.14	238.26	124.5	d6185.pdb
1cewI	13,000	0.7167	0.7336	3.874	3.809	449.38	304.34	d13869.pdb
1cqkA	13,000	0.821	0.8523	1.946	1.687	446.8	349.79	d14302.pdb
1csp_	12,500	0.7269	0.719	2.369	2.384	353.13	251.95	d1430.pdb
1cy5A	13,000	0.8739	0.8779	1.573	1.66	440.48	336.39	d23161.pdb
1dcjA_	13,000	0.3542	0.3751	10.798	12.145	335.77	143.65	d12374.pdb
1di2A_	13,000	0.7989	0.7683	2.336	2.62	374.5	223.57	d12120.pdb
1dtjA_	13,000	0.7821	0.796	2.18	2.115	389.66	335.89	d6451.pdb
1egxA	13,000	0.774	0.7731	2.598	2.617	471.79	398.52	d3845.pdb
1fadA	12599	0.5784	0.58	3.611	3.62	406.35	318.15	d1888.pdb
1fo5A	13,000	0.5197	0.5442	3.97	3.879	324.25	339.78	d279.pdb
1g1cA	13,000	0.7794	0.7812	2.648	2.499	375.03	412.5	d19130.pdb
1gjxA	12,500	0.4338	0.3926	7.372	8.226	220.33	81.27	d5311.pdb
1gnuA	13,000	0.5202	0.5446	8.82	9	414.94	181.74	d17004.pdb
1gpt_	13,000	0.5207	0.4979	5.467	6.324	327.16	170.6	d9347.pdb
1gyvA	11,508	0.7609	0.7678	3.444	3.435	375.84	305.36	d6819.pdb
1hbkA	13,000	0.6633	0.6633	3.482	3.482	356.79	216.9	d13973.pdb
1itpA	12,500	0.3268	0.3076	11.699	10.809	204.5	79.74	d7727.pdb
1jnuA	13,000	0.7242	0.7506	2.83	2.681	386.13	322.46	d7000.pdb
1kjs_	13,000	0.3789	0.383	8.466	8.436	390.63	133.71	d14800.pdb
1kviA	13,000	0.7059	0.7067	2.153	2.1	371.62	234.93	d16371.pdb
1mkyA3	6,119	0.4307	0.4139	5.095	5.175	86.93	60.31	d4435.pdb
1mla_2	12,500	0.6219	0.6349	3.036	2.824	338.14	221.02	d11922.pdb
1mn8A	12,500	0.3579	0.348	7.096	7.452	243.34	145.14	d5490.pdb
1n0uA4	12,499	0.4645	0.456	4.46	4.622	324.49	216.03	d8172.pdb
1ne3A	12,500	0.4899	0.4315	4.542	5.915	208.35	195.67	d8859.pdb
1no5A	12,500	0.4456	0.4251	10.832	10.695	389.69	153.34	d6145.pdb
1npsA	13,000	0.7686	0.7671	2.287	2.233	345.46	245.05	d6340.pdb
1o2fB_	12,500	0.3738	0.3932	8.345	6.098	219.45	78.71	d1553.pdb
1of9A	13,000	0.5391	0.5422	3.616	3.635	341.31	297.09	d1791.pdb
1ogwA_	13,000	0.6606	0.7011	2.867	2.672	366.24	273.2	d7796.pdb
1orgA	13,000	0.7666	0.7566	2.583	2.659	412.02	379.68	d6724.pdb
1pgx_	13,000	0.5043	0.5295	3.506	3.26	338.33	143.1	d10731.pdb
1r69_	13,000	0.7381	0.7538	2.027	1.971	271.85	197.38	d16473.pdb
1sfp_	13,000	0.7462	0.7281	5.312	5.326	412.33	425.41	d334.pdb
1shfA	13,000	0.8123	0.8255	1.471	1.476	283.79	143.59	d14354.pdb
1sro_	13,000	0.6616	0.6258	3.571	3.756	364.43	291.37	d10853.pdb
1ten_	13,000	0.7927	0.8233	1.93	1.837	363.24	336.27	d473.pdb
1tfi_	13,000	0.4889	0.4991	4.746	5.136	273.66	245.07	d680.pdb
1thx_	13,000	0.7944	0.7966	2.422	2.262	396.21	374.17	d16965.pdb
1tif_	12,500	0.3222	0.3339	7.584	7.574	261.39	79.01	d310.pdb
1tig_	12,500	0.4685	0.5609	9.69	3.58	277.55	98.05	d1417.pdb
1vcc_	13,000	0.4026	0.376	6.507	8.128	244.25	142.76	d2437.pdb
256bA	13,000	0.76	0.76	3.448	3.448	418.19	404	d15667.pdb
2a0b_	13,000	0.792	0.8034	2.617	2.467	441.06	365.23	d10915.pdb
2cr7A	12,500	0.4879	0.3699	3.605	8.239	223.83	69.35	d9843.pdb
2f3nA	13,000	0.721	0.7263	1.946	1.938	295.13	235.45	d13274.pdb
2pcy_	13,000	0.6273	0.6402	4.71	4.658	373.94	302.21	d9770.pdb
2reb_2	12,500	0.3285	0.3372	7.003	5.928	259.69	76.22	d6304.pdb

CPU time is as reported by the UNIX server, and hence includes file processing time.

while Calibur's decoys have smaller values in 30 cases. The total time taken by

SPICKER to cluster all 56 sets of the decoys is 18660.04 seconds CPU time, and for Calibur 12612.14 seconds CPU time.

Using the entire set of data did not result in significant improvement in the decoys obtained. Table 3 shows Calibur's results on the seven larger data sets in the 56 test cases when the full data sets are used. Here, the TM-score for the decoys obtained is 4.756, slightly lower than the score 4.766 obtained when decoys are sampled. This lack of improvement is likely because the sampling size is more than a third of the full data and the sampled data sets are sufficiently representative.

Table 6.3: TM-scores and RMSDs to native for larger decoy sets. (cf Table 6.3.3).

Target	Size	TM-score	RMSD	CPU Time	Reported decoy
1ah9_	27498	0.645 (0.6313)	3.314 (4.701)	1125.38 (199.14)	d23165.pdb
1aoy_	32000	0.6598 (0.6598)	4.761 (4.761)	3144.66 (295.87)	d16717.pdb
1cy5A	32000	0.8701 (0.8779)	1.62 (1.573)	3585.62 (336.39)	d30926.pdb
1gpt_	32000	0.5113 (0.4979)	6.292 (5.467)	1384.36 (170.6)	d13341.pdb
1tfi_	32000	0.4983 (0.4991)	5.077 (4.746)	2111.49 (245.07)	d679.pdb
1thx_	32000	0.7966 (0.7966)	2.262 (2.422)	3939.86 (374.17)	d16965.pdb
2a0b_	32000	0.7745 (0.8034)	2.782 (2.617)	3804.93 (365.23)	d9949.pdb

The numbers in brackets are the values obtained from using the sampled sets of 13,000 decoys each.

6.4 Rationale for the Ensemble-based Methods

The Ensemble-based methods rely heavily on the hypothesis that if a structure sampling method generates two protein structures that are close to each other, both are likely to be native-like. Here, this hypothesis is investigated. A probabilistic foundation is given, and applied to develop a new method for decoy selection.

6.5 Hypothesis

Most MC based methods for *ab initio* protein structure prediction consist of two major components:

- a scoring or an energy function $\mathcal{E}$ to evaluate the current structural conformation, and
- a searching strategy to direct the search process to move from one conformation to subsequent conformations.

After making a series of searches directed by the energy function, the decoy with the lowest energy is ultimately generated as the predicted structure for the given protein sequence. A set of decoy structures $\mathcal{S}$ can be obtained by repeating the process.

In the following discussion, the native structure is denoted as s_0, and the number of residues in s_0 is denoted ℓ. $\mathcal{D}(s, s')$ is a distance function that measures the distance between two decoys s and s'. Ideally, $\mathcal{D}(s, s')$ quantifies the number of movements from s to s' in the search process. In particular, $\mathcal{D}(s, s_0)$ measures the distance between a decoy s and its corresponding native structure s_0; that is, the quality of decoy s. The assumption that most ensemble-based selection methods rely on can be stated as the following hypothesis.

Hypothesis: $\mathcal{D}(s, s_0) < \mathcal{D}(s, s')$ *holds with a high probability for any* $s, s' \in S$, *where* S *refers to a set of decoys, randomly generated by an MC search method.*

In most Monte Carlo-based protein structure prediction methods, it is assumed that the energy functions favor near-native decoys during search. In the process of this search, each decoy can be treated as the result of a random trial in a sample space with 3ℓ dimensions. Specifically, each protein structure can be considered as a 3ℓ dimensional point. It is unlikely for two randomly drawn high-dimensional points to be close; that is, the distance between two decoys s and s' tends to be random due to the property of high dimensional space. In contrast, if the energy function, used to direct the search, favors near-native movements, a reasonable conclusion is that s tends to be close to s_0 although the distance between s and s' is large. This hypothesis is in general true; that is, it ought to hold under reasonable distance functions and energy functions.

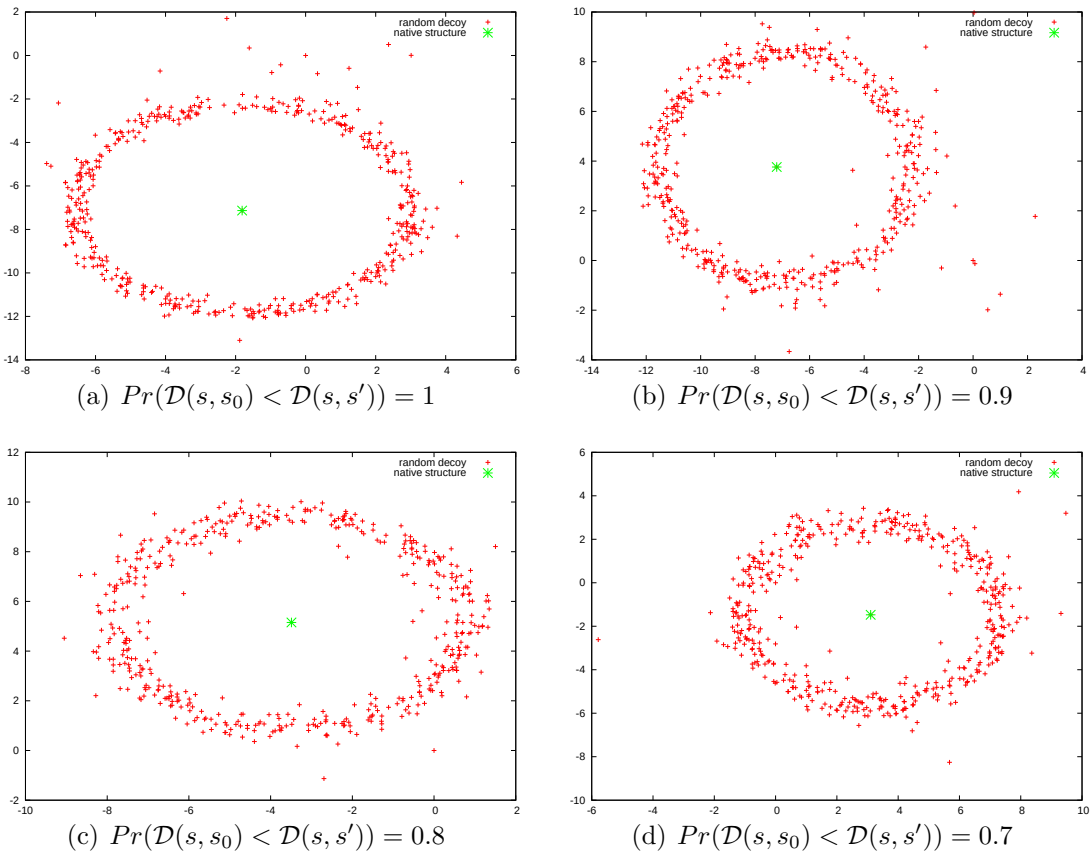
Given that the hypothesis holds for a given set of decoys and a given distance function, it is very likely that the native structure would be at the centroid of this set. If so, this would rationalize both the density score function technique and the maximum neighbor method. First, this is demonstrated by using simulation.

6.5.1 Verify the Hypothesis by Simulated Data

Simulations are employed to verify our assertion that when the hypothesis holds, the decoys would have the native as their centroid. In each simulation, 500 points are generated, where each point represents one decoy. Given native structure s_0 and point s', a new point, s is generated with the probability $Pr(\mathcal{D}(s, s_0) < \mathcal{D}(s, s'))$. In these simulations, the probability is set to be 1, 0.9, 0.8, and 0.7, respectively. The results are as shown in Fig 6.7. All the points are optimally mapped onto a 2D plane for an intuitive observation. The four experiments display similar results; that is, the point which represents the native structure consistently lies at the centroid of the cluster.

6.5.2 Verifying the Hypothesis on Real Data

The premise of the hypothesis is that the energy function favors decoys that are close to the native *by the distance function*. Given this, one would expect a function that is more favored by the energy function to perform better in decoy selection.



(a) $Pr(\mathcal{D}(s, s_0) < \mathcal{D}(s, s')) = 1$ (b) $Pr(\mathcal{D}(s, s_0) < \mathcal{D}(s, s')) = 0.9$

(c) $Pr(\mathcal{D}(s, s_0) < \mathcal{D}(s, s')) = 0.8$ (d) $Pr(\mathcal{D}(s, s_0) < \mathcal{D}(s, s')) = 0.7$

Figure 6.7: Embedding the decoys in the 2D plane.
The decoys are colored in green, and the native structure are colored in red.

This conjecture is new verified, using actual output from two MC protein prediction methods based on different energy functions.

Given a set S of decoys and a distance function d, we first compute $\overline{d}(s)$, the average distance between decoy s and the other decoys in S. The distance between a decoy s and the native structure is denoted $d(s, s_0)$. $\frac{1}{|S|}\sum_{s\in S}\frac{d(s,s_0)}{\overline{d}(s)}$ is used as an indicator of how close the decoys are to the native *by the measure* d. This gives an idea of how well d is favored by the energy function which is used to produce the decoys.

Four distance functions: RMSD, GDT, TMScore [174], and MaxSub [137] are employed. (These measures are originally proposed to measure the quality of predicted structures.) As data the 56 targets from SPICKER's website [175] and 11 data sets generated by FALCON are used as the data. For each of the data sets, we do the following. An indicator value is computed for each distance function. The functions are then ranked by their indicator values. For each of the function the maximum neighbor based method on the function is used to obtain a decoy. This gives a decoy from each function, which then are ranked by some distance measure Q. $X_{i,j}$ is to denote the number of times that the rank i function obtained a decoy

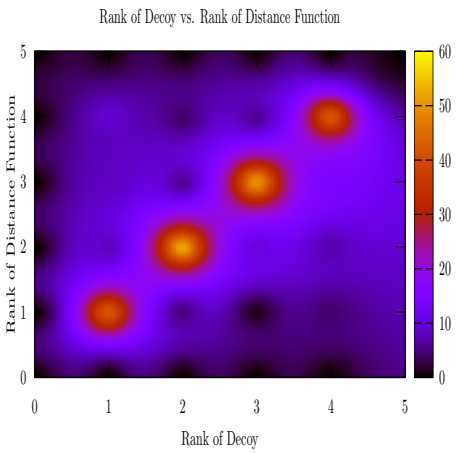


Figure 6.8: Rank of distance functions according to the indicator (x-axis) and the rank of selected decoy (y-axis).

of rank j.

In Figure 6.8, the values for $X_{i,j}$, $1 \leq i, j \leq 4$, using the GDT measure for Q is shown. The intensity of each grid i, j corresponds to the value of $X_{i,j}$. The ordering of the points along a diagonal line lends evidence to the conjecture. Similar results are obtained for other measures of Q.

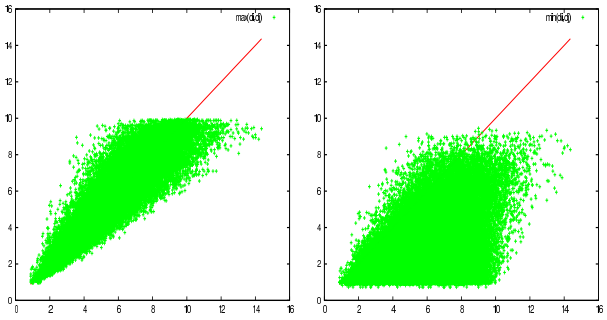
6.6 New Measure for Selecting Good Decoys

In this section, direct evidence for the hypothesis on the RMSD is first demonstrated, by observing the relationship between $\mathcal{D}(s, s_0)$ and $\mathcal{D}(s, s')$ where $\mathcal{D}$ is the RMSD. Then, it is observed that for a collection of decoy pairs of a fixed distance, the distances of the decoys to the native structure roughly conform to a Gaussian distribution. This observation allows us to devise a new measure for ranking decoys.

For native, the six widely used benchmark proteins (see Table 6.7) are used. For each protein, 10,000 decoys at various accuracy levels, are generated with the program FALCON; their RMSDs to the corresponding native structure are from 1 to 10 Angstroms.

Given two decoys, their distance and their respective distances to the native structure are computed. The smaller of the two distances to the native is referred to as the *lower* distance, the larger is referred to as the *upper* distance. The distance between decoys is plotted against the distance to the native structure. For clarity, two separate plots are plotted: one of the lower distance versus the distance to native, and one of the upper distance versus the distance to native.

Figure 6.9 shows the relationship between $\max(\mathcal{D}(s, s_0), \mathcal{D}(s', s_0))$ and $\mathcal{D}(s, s')$, and the relationship between $\min(\mathcal{D}(s, s_0), \mathcal{D}(s', s_0))$ and $\mathcal{D}(s, s')$. This figure reveals that: (i) the probability that $\min(\mathcal{D}(s, s_0), \mathcal{D}(s', s_0))$ is greater than $\mathcal{D}(s, s')$ is



(a) 1FC2. (b) 1ENH.

Figure 6.9: Relationship between $max(\mathcal{D}(s, s_0), \mathcal{D}(s', s_0))$ and $\mathcal{D}(s, s')$ (left-hand side), and the relationship between $min(\mathcal{D}(s, s_0), \mathcal{D}(s', s_0))$ and $\mathcal{D}(s, s')$ (right-hand side).

very low; (ii) $\max(\mathcal{D}(s, s_0), \mathcal{D}(s', s_0))$ is symmetric. Thus, we know that $\mathcal{D}(s, s_0) < \mathcal{D}(s, s')$ holds for high probability if the RMSD is used as an distance function for the decoys generated by FALCON.

Secondly, the distribution of $\mathcal{D}(s, s_0)$, when $\mathcal{D}(s, s')$ is fixed, is described. Figure 6.10 suggests that the conditional distribution $Pr(\mathcal{D}(s, s_0)|\mathcal{D}(s, s') = d_{ij})$ can be approximated by a Gaussian distribution; that is,

$$Pr(\mathcal{D}(s, s_0) = d|\mathcal{D}(s, s') = d_{ij}) = C\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(d-d_{ij})^2}{2\sigma^2}}$$

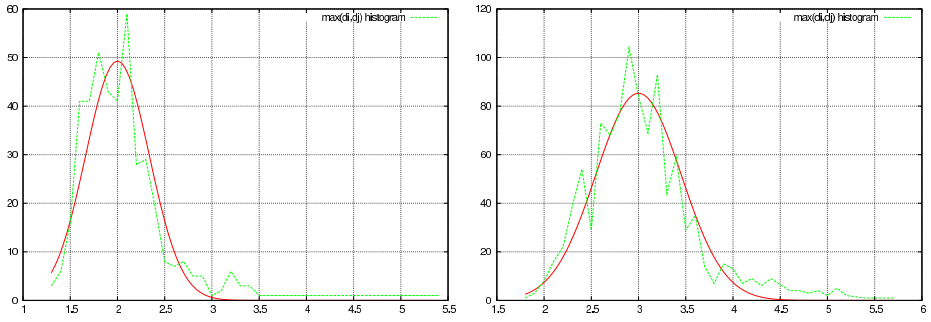
where C is the scale factor for normalization, and σ is the standard variation. Accordingly, the parameters are estimated.

Based on the aforementioned discussion, the conditional probability, $Pr(\mathcal{D}(s, s_0) = d|\mathcal{D}(s, s') = d_{ij})$, is computed; that is, the probability that $\mathcal{D}(s, s_0) = d$ for a pair of decoys s and s' with a distance of d_{ij}. Then the probability that a decoy has quality q is expressed as:

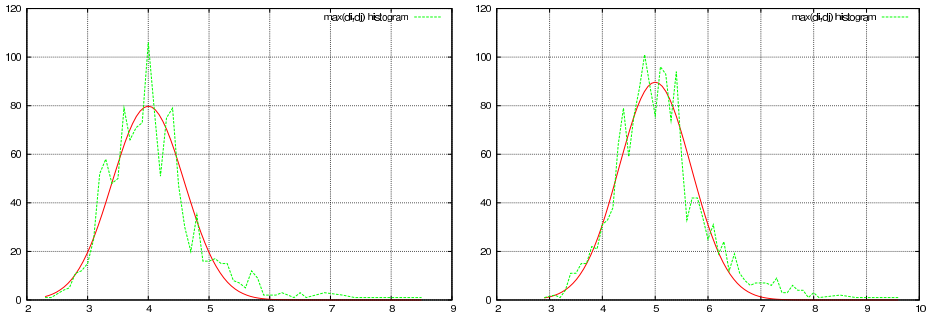
$$Pr(\mathcal{D}(s, s_0) = q) \propto \sum_{s' \in \mathcal{S}, s' \neq s} Pr(\mathcal{D}(s, s_0) = q|\mathcal{D}(s, s') = d) \times Pr(\mathcal{D}(s, s') = d).$$

Consequently, the decoy which maximizes the above equation can be identified, by assigning a set of values to q iteratively. However, things are more complicated in practice: the collection of decoys may contain some totally random structures due to imperfection in the energy function. These decoys are "random noises", which should be removed prior to the decoys selection process. To achieve this goal, decoys are extracted layer by layer until a core set remains. Only the decoys in the core set are ranked. This is the reason for the name ONION.

The ONION algorithm is described in Figure 6.4. In the algorithm, the neighborhood of s is defined as $\mathcal{N}(s, \theta) = \{s'|\mathcal{D}(s, s') \leq \theta, s' \in \mathcal{S}\}$, where $\mathcal{S}$ denotes



(a) 1FC2. (b) 1ENH.



(c) 2GB1. (d) 2CRO.

Figure 6.10: Distribution of $\mathcal{D}(s, s_0)$, when $\mathcal{D}(s, s')$ is fixed at 2, 3, 4, and 5 Angstroms. The Gaussian distributions approximating these distributions are also shown with σ estimated to be 0.15.

the given decoy set, and θ is a threshold. The neighborhood size is defined as $r(s, \theta) = \frac{|\mathcal{N}(s,\theta)|}{|\mathcal{S}|}$. ϵ and ϵ' are two small constants, and $0 < \alpha < 1$ and $0 < \beta < 1$ represent two parameters.

The idea of ONION follows. Given a set of decoys $\mathcal{S}$, the decoys that do not have enough neighbors by tuning the parameter β are removed. The reason is that, according to the analysis in the previous subsections, a decoy with few neighbors is most likely a bad (or random) decoy. The threshold indicating the neighborhood size is tuned down, if the portion of decoys removed from $\mathcal{S}$ is greater than α. This removing operation is repeated, until no further decoys extraction is possible. In addition, the decoys that are too similar (controlled by a small constant ϵ') are considered as a single decoy and only one copy of them is retained. Finally, the remaining decoys are ranked according to the discussion from the last subsection.

Onion Algorithm

Input: A set of decoys: $\mathcal{S}$

Output: Rank of of decoys

1. $\mathcal{S}' \leftarrow \emptyset$, $\theta' \leftarrow \theta$
2. **while** $|\mathcal{S}| - |\mathcal{S}'| > \alpha|\mathcal{S}|$
   $\mathcal{S} \leftarrow \mathcal{S}'$
   foreach $d \in \mathcal{S}$
   if $r(\mathcal{S}, d, \theta') < \beta$ **then**
   $\mathcal{S}' \leftarrow \mathcal{S}' - \{d\}$
   $\theta' \leftarrow \theta' - \epsilon$
3. $\mathcal{S}'' \leftarrow \emptyset$.
4. **foreach** $d \in \mathcal{S}$
   if $\forall d' \in \mathcal{S}''$ such that $\mathcal{F}(d, d') \geq \epsilon'$ **then**
   $\mathcal{S}'' \leftarrow \mathcal{S}'' \cup \{d\}$.
5. Rank decoys in $\mathcal{S}''$

Table 6.4: Proposed algorithm to rank a set of decoys

Table 6.5: Quality (RMSD) of the best decoy reported by ROSETTA clustering tool and ONION.

Target Protein	ROSETTA	ONION
Protein A, 1FC2	3.590	3.734
Homeodomain, 1ENH	1.989	2.020
Protein G, 2GB1	3.348	2.068
Cro repressor, 2CRO	3.577	3.490
Protein L7/L12, 1CTF	0.933	0.797
Calbidin, 4ICB	2.785	2.785

6.7 Decoy Selection

6.7.1 Selecting Good Decoys

To evaluate the Onion algorithm its output is compare to the clustering tool in ROSETTA. As input, 2,000 decoys are generated for each of the six benchmark proteins, used in earlier sections, with ROSETTA. Then, ROSETTA's selection and ONION are employed to select the best decoy from the decoys. The results are displayed in Table 6.5. Onion's performance is similar to ROSETTA's.

The top ten decoys reported by ROSETTA's selection program and ONION are compared. The average RMSDs are summarized in Table 6.6. For all six cases, ONION yields better decoys than ROSETTA.

Table 6.6: Quality (average RMSD) of the top ten decoys reported by ROSETTA clustering tool and ONION.

Target Protein	ROSETTA	ONION
Protein A, 1FC2	3.671	3.649
Homeodomain, 1ENH	2.217	2.101
Protein G, 2GB1	3.373	2.030
Cro repressor, 2CRO	3.711	3.660
Protein L7/L12, 1CTF	1.078	0.876
Calbidin, 4ICB	3.315	3.151

6.7.2 Refine Decoys from *ab initio* Methods

It is interesting to investigate how Onion may be integrated into the actual usage of *ab initio* protein structure prediction. Here, an integration of Onion with FALCON is attempted.

FALCON typically generates 1000 decoys at each iteration. Instead of using all the decoys as input for the next iteration, ONION is employed to select the top 20 decoys, and only these 20 decoys are used. In this experiment, the full atom energy function of ROSETTA is adopted in FALCON, and the HMM is extended to handle the side chains. In so doing, the experiment is set up to examine the validity of the hypothesis on ROSETTA's energy function.

The average pairwise distance among the 20 decoys is used to determine whether further iterations are necessary. Specifically, the iteration process is ceased if the gap between the average pairwise distances of two consecutive iterations is small (0.15 is used in the present study), or the average of pairwise distances between two consecutive iterations increases. Table 6.7 displays the average pairwise distance between the input decoys, demonstrating that the distances among top decoys are smaller and smaller as the iteration proceeds.

In the case of 1FC2, only two iterations are performed, since the distance at the second iteration is very close to that of the first iteration. Similarly, only three iterations are performed for 1ENH and four iterations for 2GB1. For 1CTF, the distance is increasing at the second iteration, hence the iteration ceased at that iteration. The third iterations for 2CRO and 4ICB are terminated similarly.

Table 6.8 displays the final reported decoys (in bold) and the selected decoys at each iteration. The quality of the decoys are shown to be optimized at the final iteration.

Figure 6.11 shows the superposition of the final decoy and its corresponding native structures. It is evident that, in all cases, good quality structures are produced, validating the effectiveness of the integrated system, as well as showing ROSETTA's energy function to conform to the hypothesis.

Table 6.7: Average pairwise distance of the top 20 decoys at each iteration.

Target Protein	# Iterations			
	1	2	3	4
Protein A, 1FC2	0.269	0.189	-	-
Homeodomain, 1ENH	0.632	0.303	0.174	-
Protein G, 2GB1	0.578	0.355	0.284	0.283
Cro repressor, 2CRO	1.398	0.463	0.590	-
Protein L7/L12, 1CTF	1.471	4.096	-	-
Calbidin, 4ICB	1.936	1.393	1.785	-

The value is the average pairwise distance in the top 20 decoys.

Table 6.8: Final decoys (in bold) and the quality (RMSD) of the best decoys at each iteration step.

Target Protein	# Iterations			
	1	2	3	4
Protein A, 1FC2	3.796	**3.705(1.75)**	-	-
Homeodomain, 1ENH	2.156	1.907	**2.028(0.82)**	-
Protein G, 2GB1	1.696	1.463	1.315	**1.330**
Cro repressor, 2CRO	3.115	**2.497**	-	-
Protein L7/L12, 1CTF	**0.937**	-	-	-
Calbidin, 4ICB	3.005	**2.588**	-	-

The number in bracket indicates the decoy quality, if the heading loops are excluded.

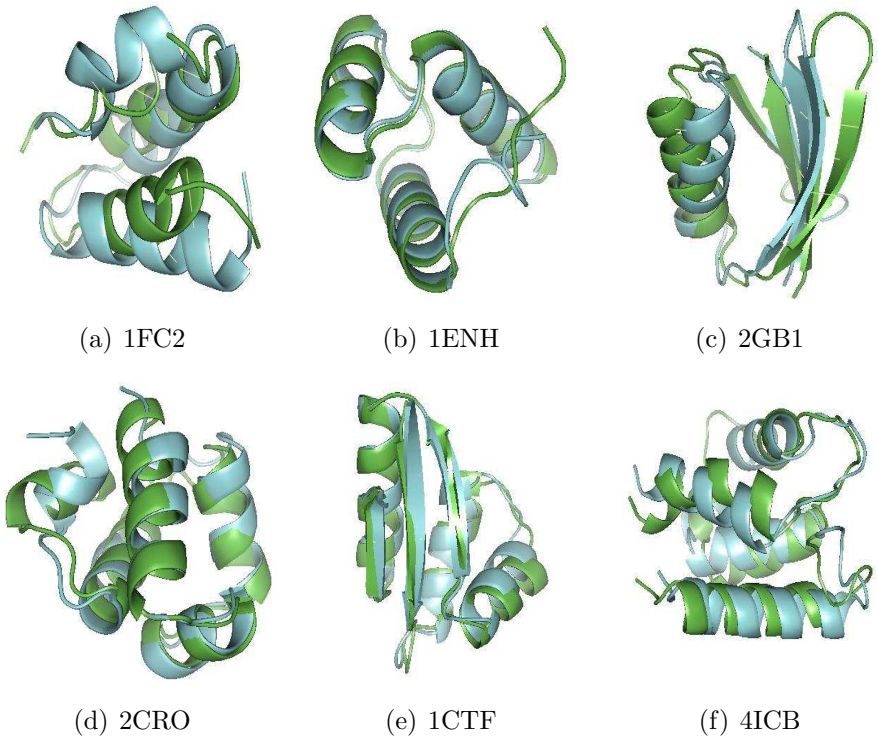
6.8 Summary and Discussion

The following remarks are made to conclude this chapter.

The decoy clustering program, Calibur, presented in the first half of this chapter, has the ability to cluster very large number of decoys. As methods in *ab initio* protein structure prediction advances, the number of decoys to be analyzed is expected to increase, and the disability to cluster decoys efficiently will, eventually, pose a hindrance to the analyses of various problems and subproblems in the prediction of protein structures. Hence Calibur will come in useful when this situation arises.

In the second half of this chapter, an underlying assumption in ensemble-based selecting methods is investigated. A probabilistic foundation is given for this assumption. Based on this foundation, a method is devised for decoy selection and structure refinement, with promising results. Even though it is assumed that there is only one cluster in the generated decoys, it is not difficult to extend the hypothesis to multiple clusters.

An MC method depends on two components: a search method and an energy



(a) 1FC2 (b) 1ENH (c) 2GB1

(d) 2CRO (e) 1CTF (f) 4ICB

Figure 6.11: Final reported decoys by ONION, superimposed with the native structures.

function. The FALCON-ONION iterative strategy provides a framework for the search process. To complement the framework with a good energy function is where future work should be conducted.

Chapter 7

Model Comparison

Each protein structure prediction method generates numerous models of the target structure. The quantitative evaluation of these models; that is, the measuring of a model's similarity to the native structure, is a difficult and fundamental question which has been intensively studied in structural bioinformatics, and is still under active research [137].

RMSD is popular as a measure for evaluating models [13]. However, it is prone to a few drawbacks. Such a measure is likely to underestimate the quality of a model, where most of the structure is accurately predicted, but the incorrectly predicted parts are far from their correct positions. The RMSD was initially proposed to handle data with a relatively small error due to noise, and cannot be appropriately employed to evaluate structures which differ by large distances. The interpretation of an RMSD value also differs for targets of different lengths. For example, the quality of a model of 10 residues with an RMSD of 3Å is considered bad, whereas the quality of a model of 100 residues with an RMSD of 3Å is considered accurate.

To eliminate these issues, measurements such as MaxSub [137], the Global Distance Test (GDT), Local/Global Alignment (LGA) [170] and TM-score [174] are proposed. A comprehensive review is given in the literature [96]. These methods are heuristic and employ RMSD minimization as a subroutine. The common schema can be summarized as follows. A set of residue pairs are taken as the starting point. Each residue pair contains a residue from the predicted model and the corresponding residue in the native structure. Then, the transformation that minimizes the RMSD between these residue pairs is calculated. By applying this transformation to the entire model, the matched residue pairs are computed as the residue pairs matched under the given threshold. This process is iterated, until no change of matched residue pairs is observed. Various resultant transformations are generated by using different starting points, and the one which maximizes the number of matched residue pairs is viewed as the final solution.

The RMSD minimization, which is employed to identify the candidate transformations, might not yield the optimal transformation for the defined measures. Two concrete examples are given in later sections, where the RMSD minimization

technique results in a wide gap to the optimal scores. This calls for the development of techniques to improve the computation of these RMSD-based measures. In particular, the GDT measure is addressed in this chapter.

In GDT, the average of the percentages of the matched residue pairs between the model and the native structure under the thresholds 1Å, 2Å, 4Å and 8Å is used as the score of the model. The step where the percentage at a given threshold is calculated is abstracted as the *largest "well-predicted" subset (LWPS)* problem; that is, to find the maximum matched residue pairs under a given distance threshold. The problem was conjectured to be NP-hard [96, 137]. The LWPS problem is actually polynomially solvable by using a computational geometry technique for solving d-LCP, the *largest common point sets under approximate congruence with a distance threshold d*. However, the high ordered polynomial runtime of the method limits its practical usage. In this chapter a $O(n^3 \log n/\epsilon^5)$ time algorithm is proposed to obtain $d/(1+\epsilon)$ distance approximation solutions to the LWPS problem, in order to compute GDT for general protein structures. For globular proteins, this result can be enhanced to a randomized $O(n \log^2 n)$ time algorithm with a probability of at least $1 - O(1/n)$. In addition, a $1/(1+\epsilon)$-approximation algorithm is proposed to compute the minimum distance to fit all the corresponding points of a model and its native structure in time $O(n(\log\log n + \log 1/\epsilon)/\epsilon^5)$.

7.1 Methods

7.1.1 Notations and Preliminaries

A protein structure, A, consists of an ordered set of n points in 3D space; that is, $A = (a_1, a_2, \ldots, a_n)$, $a_i \in \mathbb{R}^3$, where point a_i represents the $C\alpha$ atom coordinate of residue i. Similarly, the predicted model, B, of the protein also consists of an ordered set of n points; that is, $B = \{b_1, b_2, \ldots, b_n\}$, where b_i represents the predicted $C\alpha$ atom coordinate of residue i. In this study, the *θ-ball* of a point p is used to denote the ball of radius θ centered at p. Similarly, the *θ-sphere* of a point p can be defined. Given an index set, I, and a point set, $P = \{p_1, \ldots, p_n\}$, $P[I]$ denotes the subset, $\{p_i | i \in I\}$. Given threshold d and rigid transformation $\mathcal{T}$ (including a rotation and a translation), if $|a_i - \mathcal{T}(b_i)| \leq d$, a_i is said to *match* b_i, or b_i *fits into* a_i under $\mathcal{T}$. $M_{\mathcal{T}} = \{i | |a_i - \mathcal{T}(b_i)| \leq d\}$ is a *matching set* under distance threshold d, and d is referred to as the *bottleneck distance*.

The chemical characteristics of proteins supply some specific properties for protein structures. The properties in this chapter are listed as follows.

Property 1. A protein structure, A, is bounded within a ball with radius R_A. $R_A = O(n)$ for general proteins, and $R_A = cn^{1/3}$ for globular proteins (c is a constant) [94]. For notation simplicity, the leading constant of R_A is omitted for a globular protein structure.

Property 2. The distance between any two points (Cα atoms) in a protein structure cannot be too small due to steric clashes. More exactly, the distance between any two non-consecutive points is no less than $4\mathring{A}$ and the distance between any two consecutive points is about $3.8\mathring{A}$.

Due to these distance constraints, the maximum number of points which can be encapsulated in a given ball with radius r is proportional to the volume of the ball. When the context is clear, r^3 and the number of points that can be encapsulated in the ball are used exchangeably.

7.1.2 Problem Statement

The following formalizes the problems studied in this chapter

Largest Well-predicted Subset (LWPS) Problem Given protein structure A, model B and threshold d, the *largest well-predicted subset problem*, or $LWPS(A, B, d)$, is to identify a maximum match set, $M^d_{opt} \subseteq \{1, 2, \ldots, n\}$, and a corresponding rigid transformation, $\mathcal{I}_{opt}$, (a rotation and translation) [96, 137]. d is called the *bottleneck distance*. Denote $A_{opt} = A[M^d_{opt}]$ and $B_{opt} = B[M^d_{opt}]$.

Minimum Bottleneck Distance (MBD) Problem Given protein structure A and model B, find the smallest distance d_{opt} and a corresponding rigid transformation $\mathcal{I}_{opt}$ such that $\forall i, 1 \leq i \leq n$, $|a_i - \mathcal{I}_{opt}(b_i)| \leq d_{opt}$.

With a careful examination of the algorithm for the d-LCP problem [12, 38], it is obvious that the LWPS problem has a polynomial time solution in $O(n^7)$, which contradicts a previous claim in [137] that the LWPS problem is NP-complete.

Theorem 7. *[38] The largest well-predicted subset problem can be solved in $O(n^7)$ time under general transformations.*

This theorem has only theoretical significance due to the high ordered running time. It is still demanding to develop practical algorithms. One approach to NP-complete problems or problems with high time complexities is to utilize approximation algorithms. Such algorithms give approximate solutions with theoretically guaranteed accuracy, instead of exact solutions. Interestingly, for the LWPS problem, a small relaxation of the bottleneck distance threshold yields an efficient algorithm. Consequently, an algorithm is proposed which guarantees to identify at least ℓ' match pairs, where ℓ' is the maximum number of matched pairs

under the distance threshold, $d/(1+\epsilon)$. The relaxed version of the problems are formally described as follows.

Distance Approximation for $LWPS(A, B, d)$. Identify a rigid transformation $\mathcal{T}'$ and a matching set $M' \subseteq \{1, 2, \ldots, n\}$ such that $\forall i \in M', ||a_i - \mathcal{T}'(b_i)|| \leq d$ and $|M'| \geq |M_{opt}^{d/(1+\epsilon)}|$, ϵ is some small constant, and $\epsilon > 0$.

Bottleneck Distance Approximation. Discover a transformation, $\mathcal{T}$, such that $\forall b_i \in A$, $||a_i - \mathcal{T}(b_i)|| \leq (1+\epsilon)d_{opt}$, ϵ is some constant, $\epsilon > 0$.

7.1.3 Distance Approximation Algorithm for $LWPS(A, B, d)$

The crucial concept of the newly developed algorithm is the *radial axis*. Given point p and point set P, p' is a *radial point* in P with respect to p, iff p' is the furthest point in P from p. Points $p, p' \in P$ are called a *radial axis* of P, iff p' is a radial point with respect to p. Note that $\langle p, p' \rangle$ is a radial axis of P does not imply that $\langle p', p \rangle$ is a radial axis of P.

The traditional way to represent a rigid transformation by a translation and a rotation is not adopted here. Instead, transformation $\mathcal{T}$ for model B is represented by a radial axis alignment and a rotation around the axis.

- *Radial axis alignment*: A radial axis alignment is a rigid transformation T that transforms a given radial axis $\langle b_i, b_j \rangle$ in B to $\langle b_i, b_j \rangle$ positions under $\mathcal{T}$; that is $T(b_1) = \mathcal{T}(b_1)$ and $T(b_2) = \mathcal{T}(b_2)$. It is clear that the radial axis alignment is not unique.
- *Rotation around a radial axis*: A rotation R around the radial axis $\overrightarrow{T(b_1)T(b_2)}$ ensures that $\forall b \in B$, $R(T(b)) = \mathcal{T}(b)$.

The property of the radial axis is applied to exhaustively search the nearly-optimal transformations.

The overview of the novel algorithm is now discussed. Each ordered pair $\langle b_i, b_j \rangle$ of B is used as a radial axis candidate of B_{opt}. The transformation space is discretized to match $\langle b_i, b_j \rangle$ to $\langle a_i, a_j \rangle$. For each of such transformations, the model B is rotated around this axis, and the maximum match set is identified. The algorithm is shown in Figure 7.1.

In the following paragraphs, the three components of the new algorithm are described; that is,

- the existence of a nearly optimal transformation, given a radial axis alignment of B_{opt},
- the identification of a nearly optimal radial axis alignment, and
- the computation of the optimal rotation, given a radial axis alignment.

Table 7.1: Distance approximation algorithm for $LWPS(A, B, d)$

Input: Structure A, Model B, threshold d, and constant ϵ
Output: a transformation T, and matching set M

1. **foreach** index $i \in \{1, 2, ..., n\}$
   /* using $\langle b_i, b_j \rangle$ as radial axis candidate of B_{opt} */
2. **discretize** the d-ball C of a_i with a grid of side length $1/3\epsilon d$;
   foreach grid point b'_i of C
   discretize the sphere cap D with grids of side length $c_2\epsilon d$.
   D is defined by the portion of the $||b_j - b_i||$-sphere of b'_i
   encapsulated inside the d-ball of a_j.
   foreach grid b'_j point of D
   calculate a transformation T to map $\langle b_i, b_j \rangle$ to $\langle b'_i, b'_j \rangle$
3. **apply** T to B;
   /* using $\langle b_i, b_j \rangle$ as rotation axis */
   foreach $k \in \{1, 2, ..., n\}$
   determine the angle interval R_k that brings b_k into the d-ball of a_k;
   endfor
   use plane-sweep algorithm to find a angle γ
   covered by the maximum number of intervals
   endfor
   endfor
   endfor
   return the largest γ, the corresponding radial axis
   transformation, rotation angle and matching set.

Nearly Optimal Transformation Given a Nearly Optimal Radial Axis Alignment.

$\mathcal{T}$ denotes an optimal transformation of protein structure B, and $\langle b_1, b_2\rangle$ denotes the radial axis in B. If an approximation T' for $\mathcal{T}$ can be found such that $T'(b_1) \approx \mathcal{T}(b_1)$ and $T'(b_2) \approx \mathcal{T}(b_2)$, then there exists a rotation R around the axis along $\overrightarrow{T'(b_1)T'(b_2)}$, which transforms each point $b \in T'(B)$ to some point near $\mathcal{T}(b)$. The proof is as follows.

Lemma 8. *Given point set B, rigid transformations $\mathcal{T}$ and T', let $\langle b_1, b_2\rangle$ be a radial axis of B. If $|\mathcal{T}(b_1) - T'(b_1)| \leq \epsilon$ and $|\mathcal{T}(b_2) - T'(b_2)| \leq \epsilon$, then there exists a rotation R around the axis along $\overrightarrow{T'(b_1)T'(b_2)}$, such that $\forall p \in B$, $||R(T'(b)) - \mathcal{T}(b)|| \leq 3\epsilon$.*

Proof. Denote $b'_1 = T'(b_1)$ and $b'_2 = T'(b_2)$. With two points fixed, the only degree of freedom for a rigid transformation T' on B are the rotations around the axis along $\overrightarrow{b'_1 b'_2}$. Therefore, it suffices to show that there exists a transformation T'' which transforms $\mathcal{T}(B)$, such that $T''(\mathcal{T}(b_1))$ coincides with b'_1, $T''(\mathcal{T}(b_2))$ coincides with b'_2, and $\forall b \in B$, $||T''(\mathcal{T}(b)) - \mathcal{T}(b)|| \leq 3\epsilon$.

T'' is divided into two steps. First, $\mathcal{T}(B)$ is translated with translation t such that $\mathcal{T}(b_1)$ coincides with $T'(b_1)$. Second, $\mathcal{T}(B) - t$ is rotated with rotation axis as the line which passes though b'_1, and is orthogonal to the plane defined by points b'_1, $\mathcal{T}(b_2) - t$ and b'_2, with the rotation angle as the angle formed by $\mathcal{T}(b_2) - t$, b'_1 and b'_2, where b'_1 is the vertex. Denote this rotation as R'' and the rotation angle as α. It is verified that $b'_1 = T''(\mathcal{T}(b_1))$ and $b'_2 = T''(\mathcal{T}(b_2))$. With rotation R'', $\mathcal{T}(b_2) - t$ is moved to coincide with b'_2.

By translation t, $\forall b \in B$, $||\mathcal{T}(b) - (\mathcal{T}(b) - t)|| = ||t|| = ||\mathcal{T}(b_1) - b'_1|| \leq \epsilon$. Since $||\mathcal{T}(b_2) - T'(b_2)|| \leq \epsilon$, $||(\mathcal{T}(b_2) - t) - b'_2|| \leq 2\epsilon$. It can verified that the angle formed by points b'_1, $\mathcal{T}(b) - t$ and $R(\mathcal{T}(b) - t)$ with b'_1 as the vertex is at most α, and that $||b_1 - b|| \leq ||b_1 - b_2||$. With these two properties, $||(\mathcal{T}(b) - t) - R(\mathcal{T}(b) - t)|| \leq ||(\mathcal{T}(b_2) - t) - b'_2|| \leq 2\epsilon$. Therefore, by triangle inequality, $||T''(b) - \mathcal{T}(b)|| \leq 3\epsilon$. The statement holds. □

Finding a Nearly Optimal Radial Axis Alignment.

If a radial axis $\langle b_i, b_j\rangle$ of B_{opt} matches the pair $\langle a_i, a_j\rangle$ of A, then $||a_i - \mathcal{T}_{opt}(b_i)|| \leq d$ and $||a_j - \mathcal{T}_{opt}(b_j)|| \leq d$.

As portrayed in Figure 7.1, the d-ball of a_i is partitioned with 3D grids of side length $1/3\epsilon d$. The number of grid points to partition the d-ball of a_i is bounded by $O(d^3/(1/3\epsilon d)) = O(1/\epsilon^3)$. Here all the grid positions are tried for b_i.

Once b_i is fixed at a grid point, all the possible positions for b_j fitting into the d-ball a_j, form a sphere cap that is centered at b_i with radius $||b_j - b_i||$ and is contained in the d-ball of a_j. The spherical cap has an area of $O(d^2)$, and this area is to be roughly partitioned with grids of resolution size $1/\epsilon$. This is done by

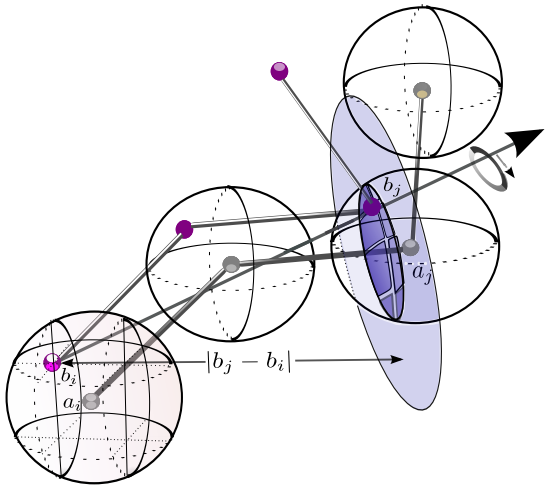


Figure 7.1: Approximating $\mathcal{T}_{opt}(b_i)$ and $\mathcal{T}_{opt}(b_j)$.

First, a radial axis is matched to the approximated positions (the grid points), and then rotate B is rotated around this radial axis to find a maximum match. Each of A and B consists of four points in this example. b_i is matched to a_i approximately. Then, the possible directions for $\langle b_i, b_j \rangle$ are discretized. Last, B is rotated around $\langle b_i, b_j \rangle$ to find a maximum match.

creating the smallest cube which encapsulates the sphere (the one the sphere cap belongs to) and create grids of side length of $O(1/\epsilon)$ on the six faces of the cube. Then, the grid on the cube is used to partition the sphere cap. This is a common trick used in computation geometry to round directions [5]. Note that there is no need to create the grid explicitly. It is easy to show that only $O(1/\epsilon^2)$ grid points are necessary to partition the sphere cap.

Combining the discretization with Lemma 10, to be shown later, the following results are produced.

Lemma 9. *If the radial axis $\langle b_i, b_j \rangle$ of B_{opt} is matched to a pair $\langle a_i, a_j \rangle$ of A, there are $O(1/\epsilon^5)$ possible choices to transform $\langle b_i, b_j \rangle$, such that at least one of the transformations results in an error at most ϵd for each $b \in B$ from their optimal positions.*

Optimal Rotation Given a Radial Axis Alignment.

Among all the rotations of the points of B around a given axis, a rotation angle $\theta \in [0, 2\pi)$ is to be identified such that the number of matched pairs is maximized. This problem can be efficiently solved as follows. First, the interval $[0, 2\pi)$ is represented by a unit circle. The angle that moves b_i into the d-ball of a_i, and the angle that moves b_i out of the the d-ball of a_i, together forms an arc on the circle. This results in $O(n)$ arcs which divide the circle into $O(n)$ circular intervals. To find a rotation angle which maximizes the number of matched pairs, it suffices to find a point

along the circle, which is covered by the most number of arcs. This can be solved efficiently using a plane-sweep approach [10, 39].

Lemma 10. *The LWPS(A, B, d) problem can be solved in time $O(n \log n)$, when the rotations are allowed only on a given rotation axis.*

Since it is not known which pair is a radial axis of B_{opt}, all the possible pairs of points are enumerated. There are $O(n^2)$ possible pairs. For any pair $\langle b_i, b_j \rangle$, there are $O(1/\epsilon^5)$ ways to match them to $\langle a_i, a_j \rangle$. For each of these matches, $O(n \log n)$ time is required to find the rotation angle which gives the optimal number of residue matches, according to Lemma 10. Therefore, the following result is obtained.

Theorem 11. *The $LWPS(A, B, d)$ can be solved in time $O(n^3 \log n/\epsilon^5)$ with a $d/(1+\epsilon)$ distance approximation algorithm.*

7.1.4 Randomized Algorithm for Globular Protein Structures

The distance approximation algorithm, proposed by Theorem 11, has a time complexity of $O(n^3 \log n)$, which is still inefficient. If a radial axis $\langle b_i, b_j \rangle$ of B_{opt} is known, then the problem can be solved in time $O(n \log n/\epsilon^5)$. This observation is the impetus to improve the algorithm by identifying a radial axis $\langle b_i, b_j \rangle$ or some pair good enough to approximate a radial pair. This section presents an efficient method to identify such a pair with a high probability for *meaningful* models of globular proteins.

A model is *meaningful* if the TM-score is greater than 0.4 [174]. Here, the TM-score is defined as

$$TM(A, B) = \frac{1}{n} \sum_{1 \le i \le n} \frac{1}{1 + (\frac{d_i}{d})^2} \tag{7.1}$$

where d_i is the Euclidean distance between a_i and b_i under some optimized transformation, and d has a similar meaning as in the present chapter, which is a predefined threshold.

The following assumption can be readily deduced for the meaning models.

Assupmption 1. *A meaningful prediction B of structure A has $|LWPS(A, B, d)| \ge \alpha n$, for some constant α.*

A pair of points b_i and b_j is called a *pseudo radial* pair if $|b_i - b_j| \ge (1/2\alpha n)^{1/3}$. Grids of side length $1/3(1/2\alpha)^{1/3}\epsilon d$ are created, and recall that in Section 7.1.1 for globular proteins, $R_B = n^{1/3}$. If pseudo radial axes are used as radial axes, the error introduced at each point in the matching set is less than:

$$3\frac{n^{1/3}}{(1/2\alpha n)^{1/3}} \times \frac{1}{3}(1/2\alpha)^{1/3}\epsilon d = \epsilon d.$$

Therefore, the following statement can be made.

Lemma 12. *Given a globular protein P, rigid transformations $\mathcal{T}$ and T', and a pseudo radial axis of P, $\langle p_1, p_2 \rangle$, if $|\mathcal{T}(p_1) - T'(p_1)| \leq \epsilon$ and $|\mathcal{T}(p_2) - T'(p_2)| \leq \epsilon$, then there exists a rotation R around the axis along $\overrightarrow{T'(p_1)T'(p_2)}$ such that $\forall p \in P$, $||R(T'(p)) - \mathcal{T}(p)|| \leq 3c\epsilon$, where c is some constant.*

The proof is omitted. Thus, any pseudo radial axis provides a $(1+\epsilon)$ distance approximation algorithm.

Theorem 13. *There exists a probabilistic $d/(1+\epsilon)$ distance approximation algorithm for LWPS for globular proteins of meaningful models with a probability at least $1 - O(1/n)$ in time $O(n \log^2 n/\epsilon^5)$.*

It remains to be shown that *enough* radial axes exist.

Lemma 14. *B_{opt} contains at least $1/2|B_{opt}|^2$ pairs, b_i and b_j, such that $|b_i - b_j| \geq (1/2\alpha n)^{1/3}$.*

Proof. The number of points, confined in the ball centered at p with radius $(1/2\alpha n)^{1/3}$, $p \in P$ is bounded by $1/2\alpha n$. This implies that there are at least $|B_{opt}| - 1/2\alpha n$ points in B_{opt} with a distance of at least $(1/2\alpha n)^{1/3}$. Thus, the statement holds. □

Since there are at least $1/2(\alpha n)^2$ pseudo radial axes, randomly sampling $\lceil 1/\alpha^2 \log n \rceil$ pairs from B yields a randomized distance approximate algorithm. Note that each pair has a probability $\frac{1/2(\alpha/n)^2}{1/2n(n-1)} = O(1)$ of being a pseudo radial axis. Given that there are $\lceil 1/\alpha^2 \log n \rceil$ pairs, the probability that none of them is a pseudo radial axis is $O(1/n)$. In addition, the calculation for each pair needs $O(n \log n/\epsilon^5)$ time. Thus, the total time complexity is $O(n \log^2 n/\epsilon^5)$.

7.1.5 Approximating the Bottleneck Distance

In some applications, it is necessary to compute the minimum distance d^* such that each point b_i in B can fit into the corresponding d^*-ball of a_i, $a_i \in A$. Akutsu's techniques [10] are to address this problem. However, these techniques exhibit high time complexities. This section presents an efficient method.

First, if there is some d' such that $d' \leq d_{opt} \leq 2d'$, then the following holds.

Lemma 15. *If $d' \leq d_{opt} < 2d'$, d_{opt} can be approximated with ratio $(1+\epsilon)$ in time $O(\frac{n \log 1/\epsilon}{\epsilon^5})$.*

Proof. The interval $[d', 2d']$ is subdivided into intervals of length $0.5\epsilon d'$ (assume 1 is divisible by 0.5ϵ). There are $2/\epsilon$ such intervals in total. For each interval $\lambda_i = [d'(1 + 0.5i\epsilon), d'(1 + 0.5(i+1)\epsilon)]$ $(0 \leq i \leq 2/\epsilon - 1)$, grids of side length $1/3\epsilon d'$ are built for $(1 + 0.5(i+1)\epsilon)d'$-ball of a_i, $1 \leq i \leq n$. Then, whether there is a transformation specified by such grids to fit all the points is verified. For two

consecutive intervals λ_i and λ_{i+1}, if there is a feasible solution for interval $i+1$, and it is infeasible for interval $i+1$, then $d'(1+0.5i\epsilon) \leq d_{opt} \leq d'(1+0.5(i+2)\epsilon) + \epsilon d'$. This yields a $(1+\epsilon)d_{opt}$ algorithm readily. A binary search is conducted to find such an i, which needs $O(\log 1/\epsilon)$ search operations.

In addition, for each search operation, it is expensive if the enumerating techniques, proposed in the previous sections, are employed. Instead, any radial axis of B can be used as a radial axis candidate. In total, there are $O(1/\epsilon^5)$ possible choices for a given radial axis. Given a rotation axis, the angle to fit b_i into a_i can be modeled as an arc of a circle as discussed earlier, and reduced to a problem of checking if there is a point on the circle that is covered by n arcs. This can be done in $O(n)$ time.

Thus each search operation can be performed in $O(n/\epsilon^5)$ time. □

The remaining task is to find d' which meets the requirement of $d' \leq d_{opt} < 2d'$. The RMSD is used to achieve this goal. RMSD can be computed in linear time [13]. Recall that RMSD is defined as the minimal root mean square deviation over all the possible transformation I, i.e.,

$$RMSD(A,B) = \min_I \sqrt{\frac{\sum_{i=1}^n ||a_i - I(b_i)||^2}{n}}.$$

Let $\mathcal{D} = RMSD(A,B)$, according to the definition of RMSD, the following can be proved.

Lemma 16. $\mathcal{D} \leq d_{opt} \leq \sqrt{n}\mathcal{D}$

Proof. First, $\mathcal{D} \leq d_{opt}$ is shown. Suppose $\mathcal{D} > d_{opt}$, I' is the transformation to obtain d_{opt}, then, $\sqrt{\frac{\sum_{i=1}^n ||a_i - I'(b_i)||^2}{n}} < \sqrt{\frac{\sum_{i=1}^n d_{opt}^2}{n}} = d_{opt}$. This contradicts the definition of $RMSD$.

Secondly, $d_{opt} \leq \sqrt{n}\mathcal{D}$ is shown. If I^* is the transformation to obtain the RMSD distance, then

$$d_{opt}^2 \leq \max_{i=1}^n \{||a_i - I^*(b_i)||^2\} \leq \sum_{i=1}^n ||a_i - I^*(b_i)||^2 = n\mathcal{D}^2$$

□

Subdivide interval $[\mathcal{D}, n^{1/2}\mathcal{D}]$ into intervals $[2^i \times \mathcal{D}, 2^{i+1} \times \mathcal{D}]$, $0 \leq i \leq 1/2 \log n - 1$ (assume $1/2 \log n$ is an integer, WLOG). For each interval, grids of side length $\frac{1}{3}2^i \times \mathcal{D}\epsilon$ and balls of radius $2^{i+1} \times \mathcal{D} + 2^i \times \mathcal{D}\epsilon$ are built. If there is a feasible solution under such grids, then $d_{opt} \leq 2^{i+1} \times \mathcal{D} + 2^i \times \mathcal{D}\epsilon$. A binary search similar to the previous one can be used to find such i.

Thus, the following result can be obtained.

Theorem 17. *The bottleneck distance can be approximated with ratio* $(1+\epsilon)d_{opt}$ *in time* $O(n(\log\log n + \log 1/\epsilon)/\epsilon^5)$.

7.2 Results

The algorithm in Theorem 11 is implemented, resulting in a program called OptGDT. The implementation has been done carefully to avoid redundant computations. First, given pairs $\langle b_i, b_j \rangle$ and $\langle a_i, a_j \rangle$, $d_{i,j}$ is defined to be $d_{i,j} = |b_j - b_i| - |a_j, a_i|$. If $|d_{i,j}| < 2d$ or $|d_{i,j}| > 2d$, $\langle b_i, b_j \rangle$ is precluded as a radial axis candidate, as it is impossible for b_i to match a_i and b_j to match a_j simultaneously. Secondly, given a radial axis candidate $\langle b_i, b_j \rangle$, an upper bound for all the $O(\epsilon^5)$ axes under the pair $\langle b_i, b_j \rangle$ is computed by employing the approximation algorithm in [39]. If the bound is smaller than the best solution so far, there is no need to evaluate the pair $\langle b_i, b_j \rangle$ any further. Thirdly, the pairs which are more likely to be the best solution are first examined. Additional rules are employed to accelerate the program.

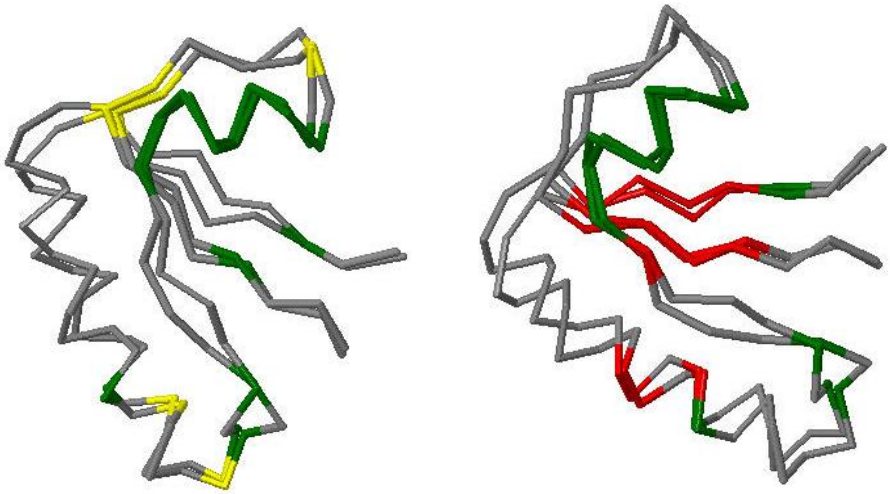
The results consist of two parts. In the first, the superposition yielded by OptGDT for two concrete examples are shown, and compared to the results from the original GDT computation. In the second, OptGDT is used to compute the GDT scores for the models predicted by the top ten servers in [1]. For each server, only the first model reported is tested. There are 172 domains. Most of the servers have predicted results for each domain, resulting in a total of 1,714 models reported for all the domains. In all the computations, $\epsilon = 0.1$.

7.2.1 Two Concrete Examples by OptGDT

Two concrete instances which demonstrate better superpositions from OptGDT are described here.

Figure 7.2 presents the superpositions from GDT1 by using the original GDT computation and OptGDT, respectively. The model is reported by BAKER-ROSETTA for target T0490, domain 2. There are 56 residues, and the residue numbers range from 88 to 143. The GDT1 score that is obtained by the original method is 0.3393, and that obtained by OptGDT is 0.4464. The increment is more than 0.1. The matched residues by the original method are the residues, numbered 90, 95, 108, 110, 114-115, 117, 122-129, 131, 135-136, and 140. The matched pairs by OptGDT are the residues numbered 90-96, 104-105 107-108, 115, 117, 121-130, and 137-140. For clarity, the superpositions are color-coded. The green parts correspond to the matched pairs, common to the superpositions from both methods. The yellow parts are the pairs matched only in the original method's superpositions, and the red parts are the pairs matched only in OptGDT's superpositions.

Figure 7.3 depicts the superpositions by the original GDT method and OptGDT based on their computations for GDT8. The model is by the Zhang-Server for the target T0496, domain 2. There are 74 residues with residue numbers from 105 to 178. The GDT8 score from the original method is 0.6351, and the score from OptGDT is 0.7838. The increment is more than 0.145. The residues matched by the original method have residue numbers 130-131 and 134-178, and the residues,



(a) Superposition by Orig. GDT (b) Superposition by OptGDT

Figure 7.2: Superpositions of the GDT1 by the original GDT and OptGDT for T0490, domain 2 for BAKER-ROSETTA.

matched by OptGDT, have numbers 105-115, 130-142 144-146, 148-165, and 167-178. The increase in the score is due to the additional matching of the residues 105-115.

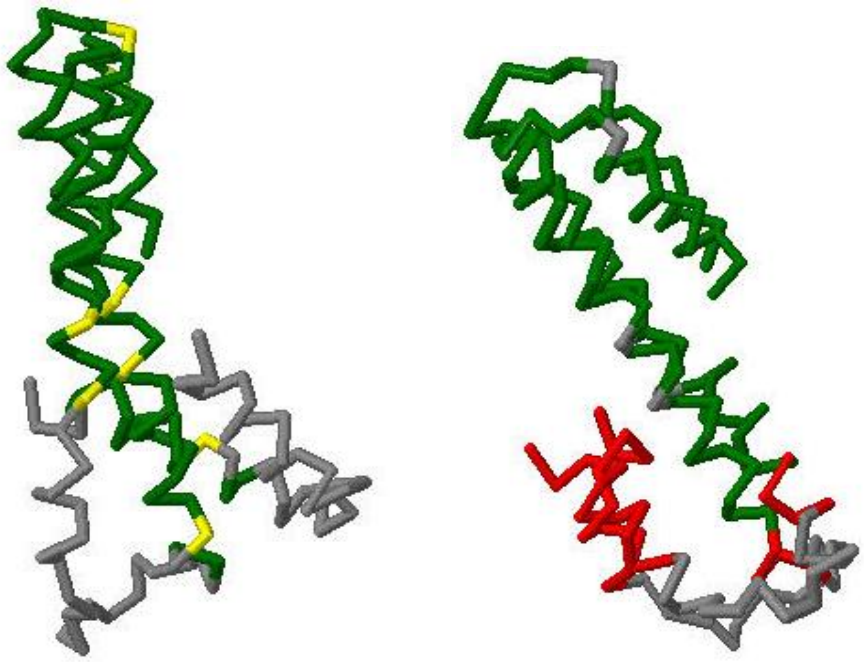
7.2.2 Performance of OptGDT on CASP8 Data

In this section, the original total GDT scores of each server in CASP8 are compared with the total GDT scores computed by OptGDT. The original scores are computed using the program TM-score. The TM-score program is able to compute the MaxSub Score, RMSD, and GDT score.

The scores are listed in Table 7.2. The servers are sorted according to the GDT scores obtained by OptGDT. The total score of each server is increased by at least two. Therefore, the GDT's average score for each domain is increased by at least 0.011. The ranks of some servers are altered due to the more accurate GDT score computation. The reordered servers are MULTICOM-REFINE, MULTICOM-CLUSTER, and Phyre_de_novo.

7.2.3 More Accurate Score Computation

OptGDT improves many of the original GDT scores computed for the predicted models. Table 7.3 shows the number of models with changes in the score that exceed a specified value. Among 1,714 models, 1,497 models have their scores increased, which are more than 87.3%. The rest remains unchanged. Larger changes are observed for GDT1 and GDT8 than for GDT2 and GDT4.



(a) Superposition by Orig. GDT. (b) Superposition by OptGDT.

Figure 7.3: Superpositions of the GDT8 by the original GDT and OptGDT for T0496, domain 2 for Zhang-Server

Table 7.2: Traditional GDT scores and OptGDT GDT scores

	Orig. GDT		New GDT	
Serves	GDT	Rank	OptGDT	Rank
Zhang-Server	114.58	1	116.79	1
RAPTOR	110.70	2	112.71	2
pro-sp3-TASSER	109.83	3	112.38	3
BAKER-ROSETTA	109.23	4	111.60	4
MULTICOM-REFINE	108.46	7	110.79	5
MULTICOM-CLUSTER	108.56	6	110.77	6
Phyre_de_novo	108.56	5	110.72	7
MUProt	108.32	8	110.65	8
MULTICOM-RANK	107.03	9	109.22	9
PS2-server	104.30	10	106.62	10

(Column 1) Server name.

(Column 2) Sum of original GDT scores for CASP8 data.

(Column 3) Rank according to the original GDT scores.

(Columns 4 and 5) GDT scores and rank by OptGDT, respectively.

Table 7.3: Number of models with scores that are increased more than a certain constant.

	GDT1	GDT2	GDT4	GDT8	GDT
>0.08	4	0	0	22	0
>0.07	20	0	2	46	0
>0.06	34	1	6	58	0
>0.05	90	20	26	93	4
>0.04	184	66	73	139	9
>0.03	441	202	219	227	54
>0.02	862	510	497	371	376
>0.01	1284	1011	979	678	1085
>0	1383	1183	1151	825	1497
=0	731	931	963	1289	217

(Column 2-6) Number of models with score changes exceeding the value specified in the first column.

7.3 Summary and Discussion

In this chapter, approximation algorithms are proposed with accuracy guarantees for computing GDT, a popular measure for evaluating predicted protein models. One of these algorithms is implemented into an efficient and practical software package called OptGDT. The software package can be used to verify the GDT values computed by using other heuristic methods, as well as to identify better superpositions in the GDT computation.

Experiments on the models predicted in CASP8 indicate that the GDT scores were underestimated previously, and better superpositions are possible for most of the models. In some cases, the scores are increased by more than 10%. The ranks of a few servers are altered due to the more accurate GDT scores. This shows that impartial tools to assess the predicted models are necessary, and that more accurate superpositions can give us better insights on the structural prediction methods being studied.

The newly developed techniques can also be applied to other problems in protein structure alignment, which is addressed in the next chapter. However, the results are less interesting practically due to their high time complexities.

A few questions remain unanswered at this point. The proposed algorithm is a *distance* approximation algorithm for structural fitting. It would be interesting to know if *size* approximation, as follows, can also be obtained efficiently.

Size approximation for GDT score	
Input:	Structure A, Model B, threshold θ and constant ϵ
Output:	Transformation T and matching set M under threshold ϵ, such that $\lvert M\rvert \geq (1-\epsilon)\lvert M_{opt}\rvert$, where M_{opt} is obtained under threshold d.

Another question is to extend the results to other scores. The current algorithm computes the GDT scores. For globular structures with meaningful models, a simple modification of the algorithm can result in a PTAS for the TM-scores. However, it is unclear whether an efficient PTAS exists for general proteins. Consider where the optimal transformation for TM-score consists of only a translation along one dimension. The problem to find the optimal translation becomes one of finding the roots for high ordered polynomials. Exact solutions for high ordered polynomials cannot be obtained. Therefore, such a method to exactly compute TM-score is not feasible. Let $TM(A, B, T)$ be the TM-score for structure A and model $T(B)$, where T is a rigid transformation. Then this problem can be stated as follows.

FPTAS for TM-scores	
Input:	Structure A, model B, and constant ϵ
Output:	Transformation T satisfies $TM(A, B, T) \geq (1-\epsilon)TM(A, B, T_{opt})$.

Another less important but still interesting open problem is to find the maximum matching set under the RMSD threshold. Given threshold d, the objective is to find a maximum matching set, such that the RMSD value for this matching set is below threshold d. This problem can be stated as follows.

Maximum matching under the RMSD distance	
Input:	Structure A, model B, and threshold d.
Output:	Transformation T to maximize $\lvert M\rvert$ under $RMSD(A[M], B[M]) \leq d$, where $M \subset \{1, 2, ..., n\}$.

Chapter 8

Structural Alignment

8.1 Introduction

Typically, the molecular shape of a protein determines the protein's biological mechanism. Hence, proteins with similar 3D structures are expected to have similar functions. This allows one to predict the functions of a protein based on its structural resemblance to proteins of known functions. The incentive of such predictions has resulted in substantial development of approaches, algorithms, and software tools for comparing 3D protein structures under the name of Protein Structure Alignment [96, 102]. As a fundamental problem in bioinformatics, many heuristic algorithms have been proposed for this problem [8,9,31,43,59,60,77,95,144]. These systems perform effectively in practice. However, relatively few theoretical studies, specific to the problem, have been conducted [6, 33, 61, 94, 168].

Due to the differences in approaching the problem, proteins have been modeled in many ways. In this chapter, both the cases of treating proteins as sequences of 3D points *and* as contact maps are considered.

In modeling a protein as a sequence of 3D points, each point represents the (relative) coordinate of a residue in the protein. In this approach, the task is to superimpose two proteins in a way such that there is a good positional correspondence between the residues of the two proteins. The quality of this positional correspondence, evaluated by some scoring function, is typically given as the similarity between the two proteins. Therefore, comparing two proteins under such a model usually translates into a task of finding a rigid transformation. Here the transformation is to superimpose the two sequences of points to fulfill some objective, such as to minimize some distance measure (e.g., the LCP problem under the Hausdorff distance); or to maximize the number of (disjoint) pairs of points, each from one of the two proteins, that are within a given proximity ϵ of each other. The latter problem is referred to as the *largest common point set (LCP) problem under bottleneck distance*. This problem is known to be exactly solvable in $O(n^{32.5})$ time [12], where n is the length of the protein sequence. Akutsu [6] has

shown that there is an approximation algorithm of $O(n^{8.5})$ runtime which returns a solution that optimizes the target parameter, but does not fulfill the proximity requirement ϵ strictly. More precisely, a pair of points in that solution can be as much as 8ϵ apart. Chakraborty and Biswas [33] demonstrated an improved algorithm for each pair of points to be at most 2ϵ apart, with the same time complexity. The present work improves this runtime to $O(n^8)$. When two properties of protein structures (namely minimum inter-residue distance and globular shape) are taken into account, the runtime is shown to be $O(n^{6.5})$.

In modeling a protein as a contact map, each residue is taken to be a vertex in a graph, where an edge (called a *contact edge*) exists between two vertices, if and only if the vertices are no further than the allowed distance (e.g., 5Å). If two proteins are similar, their contact maps tend to be similar. Consequently, a typical approach to compare two proteins under such a model, is to find the largest common subgraphs of the contact maps of the two proteins. The residues' positions are not considered in such a model except for the purpose of creating edges. Nevertheless, a more realistic model where the matched vertices are no further than a threshold distance apart in the protein molecules has been suggested [168]. The resultant problem is known as a *CMO problem with distance constraint*. The threshold distance makes a difference when there is a requirement for any two residues in a protein to be at least some fixed distance apart. This restricts each residue to be matchable to only a constant number of residues in a relatively small bounded space. An interesting consequence of this is that, while the CMO problem is NP-hard and hard to approximate in the general case [61], approximation solutions can be obtained in polynomial time with the distance constraint [168]. In this thesis, it is proven that with the distance constraint: (1) a PTAS exists for the problem, but (2) it remains NP-hard to solve the problem exactly.

By indexing the points in each input 3D structure, "sequential" versions of the aforementioned problems can be defined, where no two pairs of corresponding points in the solution are to conflict with the sequential order of their indices; that is, if points i and j in one structure, where $i < j$, are respectively matched to points i' and j' in the other structure, then $i' < j'$. These versions of the two problems are studied in this chapter.

In addition, an entirely different class of problems is considered for regarding the contact map, where, instead of geometric constraint, certain relationships between the contact edges are to be preserved. A set of such relationships is called a *model*, and the contact edges are referred to as *arcs* in the context of such a problem. The model describes whether two arcs can be in precedence order ($<$), be allowed to nest ($\sqsubset$), be allowed to cross ($\between$), or any combination of these three orders. Given the contact map of a protein and model R, the *contact map pattern problem* is to identify one of the largest subsets of contact arcs under the model.

The complexity of the contact map pattern problem under different models was first investigated by Vialette [155], and then by Blin *et al.* [19, 35]. Due to these studies, this problem is known to be NP-complete in the most general case, and

are solvable in polynomial time in some sub-cases of the problem, where some restrictions are placed on the models. However, whether the contact map pattern problem has a polynomial time algorithm in the sub-case of $\{<, \wr\}$-*structured patterns*, remains unknown. In this chapter, it is discovered that the contact map pattern problem, in this case, is NP-hard.

This problem is closely related to the second question, an open problem, known as the *contact map pattern matching problem*. As with the contact map pattern problem, the complexity of this problem was first investigated by Vialette [155], and then by Blin *et al.* [19]. The problem of whether the contact map pattern matching problem has a polynomial-time algorithm with $\{<, \wr\}$-*structured patterns* is left unanswered in these works. Gramm [66, 67] proposed a polynomial-time algorithm to solve this problem. However, there appears to be an invalid assumption in the construction of the algorithm. In this chapter, the problem is shown to be NP-hard.

8.2 Problem Formulation

8.2.1 Structural Alignment

As in the previous chapter, a protein structure is modeled as a finite, ordered sequence of 3D points, written as $(p_1, p_2, \ldots, p_n)$, where each $p_i \in \mathbb{R}^3$.

The formal statement of our problems are now given.

LCP Problem under Bottleneck Distance	
Input:	sequences $P = (p_1, \ldots, p_n)$, $Q = (q_1, \ldots, q_m)$ and distance threshold $D_c \in \mathbb{R}$. Without loss of generality assume $m \geq n$.
Output:	(i) subsets $P' \subseteq P$, $Q' \subseteq Q$, $\lvert P' \rvert = \lvert Q' \rvert$, (ii) bijection $f : P' \mapsto Q'$, and (iii) rigid transformation (rotation and translation) t, fulfilling the following conditions: (A) $\max_{p \in P'} \lvert\lvert t(p) - f(p) \rvert\rvert \leq D_c$, (B) the score $S = \lvert P' \rvert$ is maximized.

f is called an *alignment*. An alignment can be *sequential* or *non-sequential*. An alignment is *sequential*, if and only if for any two points p_{i_1}, $p_{i_2} \in P'$, where the corresponding $f(p_{i_1}) = q_{j_1}$ and $f(p_{i_2}) = q_{j_2}$, $i_1 < i_2$ iff $j_1 < j_2$. Otherwise, the alignment is *non-sequential*. The LCP problem which requires alignments to be sequential is said to be *sequential*, otherwise it is *non-sequential*. Throughout this chapter, $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{f}$, $\boldsymbol{T}$, $\boldsymbol{S}$ represent an optimal P', Q', f, t, S, respectively.

Also, a protein can be modeled as a *contact map graph*. A *contact* is a pair of points in a protein that are no more than a given distance apart. Throughout this chapter, D_u denotes this distance. In order to form contact edges, $D_u \geq D_l$. The *contact map graph* of a protein consists of the residues (i.e., vertices) and their

contacts (i.e., edges). Each vertex v is associated with a 3D point $\text{pos}(v)$, indicating the residue's relative position from other points in the protein. In this chapter, the following problem concerning contact maps is examined.

CMO Problem with Distance Constraints	
Input:	contact maps $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ and distance threshold $D_c \in \mathbb{R}$. Without loss of generality assume $\|V_2\| \geq \|V_1\|$.
Output:	(i) subsets $V_1' \subseteq V_1$, $V_2' \subseteq V_2$, $\|V_1'\| = \|V_2'\|$, (ii) bijection $M : V_1' \mapsto V_2'$, and (iii) rigid transformation (rotation and translation) t, fulfilling the following conditions: (A) $\max_{v \in V_1'} \|\|t(\text{pos}(v)) - \text{pos}(M(v))\|\| \leq D_c$, (B) $S = \|\{(v,u) \in E_1' \mid (M(v), M(u)) \in E_2'\}\|$ is maximized.

M is called an *alignment*. The vertices in V_1, V_2 are considered to be ordered. Using the orderings, an alignment M is defined as either *sequential* or *non-sequential*, in a similar way to the definitions of the sequential and non-sequential for the alignments in the LCP problem. (In some literature, a sequential alignment is referred to as *non-crossing*.) Throughout this chapter $\boldsymbol{V}_1$, $\boldsymbol{V}_2$, $\boldsymbol{M}$, $\boldsymbol{T}$, $\boldsymbol{S}$ denote an optimal V_1, V_2, M, t, S, respectively.

Approximate Solutions For the above problems, the solutions aimed for are to

(1) fulfill Condition (A) for a slightly larger distance threshold; that is, $(1+\epsilon)D_c$ for some $\epsilon \in \mathbb{R}$,

(2) have a score at or above $r\boldsymbol{S}$ for some $r \in \mathbb{R}$.

Such a solution is called an (ϵ, r)-*approximation* herein. The algorithms have polynomial runtime w.r.t. both $1/\epsilon$ and r; that is, discrepancy from the optimal due to both (1) and (2) can be made arbitrarily small at the expense of runtime.

Other Notations Due to Property 2 in Section 7.1.1, it can be shown that each residue in P can be matched to at most $(1 + \frac{2(1+\epsilon)D_c}{D_l})^3$ residues in Q. Δ_ϵ is used to denote $(1 + \frac{2(1+\epsilon)D_c}{D_l})^3$ herein. For any point p and transformation t, $t(p)$ denotes the point obtained by transforming p with t. For a set of points, P, $t(P)$ signifies the set $\{t(p) \mid p \in P\}$. For $r \in \mathbb{R}$, the r-*sphere* of a point p is the sphere of radius r centered at p.

8.2.2 Contact Map Patterns

A *contact arc* consists of two points (residues). There is a sequential order between these points of a protein. Only the sequential order (rather than the 3D coordinates)

of protein structures are of concern in the study of contact map patterns. Hence, the representation of contact maps are simplified to a sequence integers.

A *disjoint Contact map Pattern* (DIS-CMP) consists of a pair, $(\mathcal{S}, \mathcal{D})$, where $\mathcal{S}$ is a set of integers, and $\mathcal{D}$ consists of a set of ordered pairs, that is, $\mathcal{D} = \{(s_l, s_r) | s_l, s_r \in \mathcal{S}, s_l < s_r\}$. A pair (s_l, s_r) is referred to as an *arc*. For an arc (s_l, s_r), s_l and s_r are referred to as the *left endpoint* and *right endpoint* of the arc, respectively. $L((s_l, s_r)) = s_l$ and $R((s_l, s_r)) = s_r$ and the integers in $\mathcal{S}$ are also referred to as *points*.

Three kinds of relations between two given arcs: $a = (s_l, s_r)$ and $a' = (s'_l, s'_r)$ are defined:

- $a < a'$ (a is *less than* a') iff $s_r < s'_l$
- $a \sqsubset a'$ (a is *nested in* a') iff $s'_l < s_l < s_r < s'_r$
- $a \between a'$ (a *crosses* a') iff $s_l < s'_l < s_r < s'_r$

The relations do not include all the possible relations between any two arcs such as two arcs sharing an endpoint. The DIS-CMP is equivalent to a contact map.

8.2.3 Maximum Contact Map Pattern Problem

The maximum Contact Map Pattern (CMP) problem under model R is to find a largest subset of contact map which is R-comparable. The problem is expressed as follows.

Disjoint CMP pattern (DIS-CMP) Problem	
Input:	A set of arcs $\mathcal{D}$, and a model R
Output:	An R-comparable subset of $\mathcal{D}$ with the largest cardinality.

The disjoint CMP under model R is DIS-CMP-R. The DIS-CMP problem for all the possible models, except DIS-CMP-$\{<, \between\}$, has been shown to be polynomially solvable [155]. These results are summarized in Table 8.1. In this chapter, the NP-hardness for the DIS-CMP-$\{<, \between\}$ problem is presented.

8.2.4 Disjoint Contact Map Pattern Matching Problem

Since DIS-CMPs are equivalent to contact maps, $\mathcal{CM}$ is used to denote a disjoint contact map pattern. A DIS-CMP $(\mathcal{S}, \mathcal{D})$ is called a *Crossing Contact Map* (CCM), iff it is $\{<, \between\}$-*structured*.

The Disjoint Contact Map Pattern Matching (DCMPM) problem is formalized as follows. Given two disjoint contact map patterns $\mathcal{CM}(\mathcal{S}_p, \mathcal{D}_p)$ (called the *pattern*) and $\mathcal{CM}(\mathcal{S}, \mathcal{D})$ (called the *target*), where $|\mathcal{S}_p| \leq |\mathcal{S}|$, discover a subset $\mathcal{S}'$ of $\mathcal{S}$ with

Table 8.1: Contact map pattern problem complexity for $n = |\mathcal{D}|$.

Contact map Pattern Problem	
$\{<, \sqsubset, \between\}$	$O(n\sqrt{n})$
$\{\sqsubset, \between\}$	$O(n^2)$ [35]
$\{<, \sqsubset\}$	$O(n^2)$
$\{<, \between\}$	NP-complete$\star$
$\{<\}$	$O(n \log n)$
$\{\sqsubset\}$	$O(n \log n)$
$\{\between\}$	$O(n^2)$ [35]

When it is not specified, the results are from [19,155]. $\star$ denotes the contributions of this thesis.

$|\mathcal{S}'| = |\mathcal{S}_p|$ such that there is a one-to-one mapping, $\mathcal{M}$, from the elements of $\mathcal{S}_p$ to the elements of $\mathcal{S}'$ that satisfies the following two conditions

- if $s_1, s_2 \in \mathcal{S}_p$ and $s_1 < s_2$, then $\mathcal{M}(s_1), \mathcal{M}(s_2) \in \mathcal{S}'$ and $\mathcal{M}(s_1) < \mathcal{M}(s_2)$;
- if $(s_1, s_2) \in \mathcal{D}_p$, then $(\mathcal{M}(s_1), \mathcal{M}(s_2)) \in \mathcal{D}$.

If such a mapping exists, we say that $\mathcal{CM}(\mathcal{S}_p, \mathcal{D}_p)$ *occurs* in $\mathcal{CM}(\mathcal{S}, \mathcal{D})$. Generally, the contact map pattern matching problem is NP-hard [61, 155]. However, some cases with restrictions on the patterns have been shown to be solvable in polynomial time.

The DCMPM problem with $\{<\}$, $\{\sqsubset\}$, $\{\between\}$, or $\{<, \sqsubset\}$-structured patterns can be solved in polynomial time, but is NP-hard for the $\{\sqsubset, \between\}$ and $\{<, \sqsubset, \between\}$-structured patterns [155]. In this chapter, the interest is in the remaining case, where the patterns are CCMs. The following formally states the problem

Crossing Contact Map Pattern Matching (CCMPM) Problem [67]	
Input:	Dis-CMP $\mathcal{CM}(\mathcal{S}_p, \mathcal{D}_p)$ and $\mathcal{CM}(\mathcal{S}, \mathcal{D})$ with $\mathcal{CM}(\mathcal{S}_p, \mathcal{D}_p)$ as a CCM
Output:	Does $\mathcal{CM}(\mathcal{S}_p, \mathcal{D}_p)$ occur in $\mathcal{CM}(\mathcal{S}, \mathcal{D})$?

8.3 Results for the LCP and CMO Problem

The results for the two problems, LCP under the bottleneck distance and CMO with the distance constraint, are presented first. The algorithms are similar for these two problems.

8.3.1 Finding the Rigid Transformation

The algorithms in for these problems follow the strategy used in the previous chapter. A suitable rigid transformation to superimpose the two protein structures is found through the use of radial axes.

The method assumes that a radial axis, p_1, $p_2 \in \boldsymbol{P}$, is known for the case of LCP under the bottleneck distance (similarly, v_1, $v_2 \in \boldsymbol{V}_1$ for CMO with distance constraint). Given the radial axis, a rotation angle, $\theta \in [0, 2\pi)$, along the radial axis which maximizes the score S, is required. If the interval $[0, 2\pi)$ is represented by a unit circle, then the angle that moves a point $p \in P$ into, and then out of the $((1+\epsilon)D_c)$-sphere of some $q \in Q$ forms an arc on the circle. These entry/exit points can be discovered in $O(mn)$ time, and they divide the unit circle into $O(mn)$ intervals. Each of these rotation intervals consists of a set of equivalent rotation angles. This property is utilized to construct the approximation algorithms. Without the loss of generality, no two entry/exit points are assumed to be the same.

The number of rotation intervals is less for globular proteins. Property 2 from Section 7.1.1 implies that each point $p \in P$, rotated through $[0, 2\pi)$, can come into the $(2(1+\epsilon)D_c)$-sphere of at most $O(m^{1/3})$ points in Q. This results in a total of $O(nm^{1/3})$, instead of $O(mn)$ rotation intervals.

Lemma 18. *Given that T, p_1, p_2 in Lemma 8 is known, there are at most $O(mn)$ rotations to evaluate to find R of Lemma 8 in the general case, and at most $O(nm^{1/3})$ rotations to evaluate in the case of globular proteins. These rotations can be discovered in $O(mn)$ time.*

8.3.2 Approximation Algorithm for LCP under Bottleneck Distance

Here, how to evaluate a rotation in Lemma 18 for the LCP problem with the bottleneck distance is discussed. Given that the points are fixed in position, this evaluation can be achieved by constructing a bipartite graph $G(P \cup Q, E)$, where $(u, v) \in E$ iff $u \in P$, $v \in Q$ and $||u-v|| \leq (1+\epsilon)D_c$, and find the maximum bipartite matching of G. Constructing a bipartite matching in general takes $O(nm)$ time. However, for a globular protein, this construction can be carried out in $O(n\Delta_\epsilon)$ time, since, by arranging the points in Q into cells of size $D_c \times D_c \times D_c$, all the points in Q within distance $(1+\epsilon)D_c$ of any point in P can be found in $O(\Delta_\epsilon)$ time.

It is clear that the same bipartite graph is constructed for each rotation within the same rotation interval. Furthermore, the bipartite graph for one rotation interval can be constructed in $O(1)$ time from that of an earlier rotation interval, since it differs by only a single edge from the previous bipartite graph. In order to know which edge is added or removed in a subsequent rotation intervals, the $O(mn)$ (resp. $O(mn^{1/3})$) entry/exit points are sorted.

The maximum bipartite matching differs by, at most, an edge for any two consecutive intervals. An algorithm for bipartite matching, which can be employed to reuse results obtained for an interval in the matching to the next interval.

Lemma 19 ([20])**.** *The bipartite matching problem can be solved with time complexity $O((|M| - |M_0|)|E|)$, where M is a maximum matching, and M_0 is some initial matching.*

By the algorithm, a maximum matching M can be computed in $O(|E|)$ time from a matching M_0 from the earlier rotation interval; that is, $O(nm)$ in general, and $O(n\Delta_\epsilon)$ for globular proteins. In summary,

1. Discovering the entry/exit points of the intervals takes $O(nm)$ time, and sorting them takes $O(nm \log nm)$ time, or $O(nm^{1/3} \log nm^{1/3})$ time for globular proteins.

2. Constructing the bipartite graph for the first rotation interval takes $O(nm)$ time, or $O(n\Delta_\epsilon)$ time for the proteins.

3. Initial matching for the first rotation interval requires $O(m^{2.5})$ time [80].

4. The remaining $O(nm)$ rotation intervals each takes $O(1)$ time for an input modification, and $O(|E|) = O(nm)$ time for finding a new matching. For globular proteins, there are $O(nm^{1/3})$ rotation intervals, each which takes $O(1)$ time for an input modification and $O(n\Delta_\epsilon)$ time for a new matching.

Lemma 20. *If a rotation axis is specified, the non-sequential LCP problem with bottleneck distance can be solved in $O(m^{2.5}+n^2m^2)$ time in general, and in $O(m^{2.5}+n^2m^{1/3}\Delta_\epsilon)$ time for globular proteins.*

It is unknown which pair is a radial axis of $\boldsymbol{P}$, nor their matching points in Q. For this reason all the possible m^2n^2 combinations of pairs of points in P and Q are exhaustively searched. By Lemma 9, each combination results in $O(1/\epsilon^5)$ possible matches. By Lemma 20, the following holds.

Theorem 21. *There is an algorithm of time complexity $O((n^2m^{4.5}+n^4m^4)/\epsilon^5)$ in the general case, or $O((n^2m^{4.5}+n^4m^{2.\bar{3}}\Delta_\epsilon)/\epsilon^5)$ in the case of globular proteins, that outputs an $(\epsilon, 1)$-approximate solution to the non-sequential LCP problem under the bottleneck distance.*

This improves the previously known best result of $O(n^{8.5})$ to $O(n^8)$ (letting $n = m$ and $\epsilon = 1$) for the general case [33]. When D_c/D_l is reasonably small, the runtime dependency on m, n for globular proteins can be further improved to $O(n^3m^{2.\bar{3}})$. The following shows how this is achieved.

The strategy for approximation as in Section 3.2 of [168] is employed. The strategy divides the points spatially, forming smaller, mutually independent sub cases, where each is solved with bipartite matching exactly, and then merged to form the solution. W_x, W_y, R_j, and k, are defined as in Section 3.2 of [168]. D is defined as $D = 2D_c$ (i.e. instead of $D = \max\{2D_c, D_u\}$ as in [168], since there is no D_u for the LCP problem).

However, instead of partitioning along one axis and obtaining blocks of size $W_x \times W_y \times D$, the proteins are partitioned along all three axes to obtain blocks of size D^3. A different k is used along each of the three axes. They are denoted as k_x, k_y, k_z, respectively. The definition of partition R_j is revised to adapt to such a partitioning. Instead of k, now there are $k_xk_yk_z$ partitioning schemes. Since there are now $(k_x-1)(k_y-1)(k_z-1)$ blocks in each partition R_j, the number of

residues in each partition R_j, $|R_j| = O(k_x k_y k_z (\frac{D}{D_l})^3)$ (since each block contains at most $(\frac{D}{D_l})^3$ residues). Consequently, there are $\#R_j = O(n/|R_j|)$ partitions under each partitioning scheme. Bipartite matching for the initial rotation interval takes $O((|R_j|\Delta_\epsilon)^{2.5})$ time for each R_j [80], hence, the bipartite matching for the initial rotation of each partitioning scheme takes time $O(\#R_j(|R_j|\Delta_\epsilon)^{2.5}) = O(n|R_j|(\Delta_\epsilon)^{2.5})$ time. To compute the bipartite matchings for the initial rotations of all $k_x k_y k_z$ partitioning schemes takes $O(k_x k_y k_z n|R_j|(\Delta_\epsilon)^{2.5}) = O(n(k_x k_y k_z)^{2.5}(\frac{D}{D_l})^{4.5}{\Delta_\epsilon}^{2.5})$ time. The remaining $O(nm^{1/3})$ rotation intervals under a partitioning scheme each requires recomputation for matching, at most, a single R_j to Q. For all $k_x k_y k_z$ partitioning schemes, this takes $O(nm^{1/3}k_x k_y k_z|E|) = O(nm^{1/3}k_x k_y k_z|R_j|\Delta_\epsilon) = O(nm^{1/3}(k_x k_y k_z)^2(\frac{D}{D_l})^3\Delta_\epsilon)$ time with the algorithm in Lemma 19. An analysis, similar to the one in Section 3.2 of [168], shows that the method leads to a $(1-\frac{4}{k_x})(1-\frac{4}{k_y})(1-\frac{4}{k_z})$ approximation.

Theorem 22. *In the case of globular proteins, there is an algorithm of*

$$O(n^3(m^2(k_x k_y k_z)^{2.5}(\tfrac{2D_c}{D_l})^{4.5}{\Delta_\epsilon}^{2.5} + m^{2.\bar{3}}(k_x k_y k_z)^2(\tfrac{2D_c}{D_l})^3\Delta_\epsilon)/\epsilon^5)$$

time complexity that outputs an $(\epsilon, 1-4(\frac{1}{k_x}+\frac{1}{k_y}+\frac{1}{k_z}))$*-approximate solution to the non-sequential LCP problem under bottleneck distance.*

For the sequential LCP problem under bottleneck distance, instead of bipartite matching, straight-forward dynamic programming is used to find the maximum number of matches. $f(\eta,\mu)$, where $1 \le \eta \le n$ and $1 \le \mu \le m$, is used to denote the optimal number of matches for the subsequences $(p_\eta, p_{\eta+1}, \ldots, p_n)$ and $(q_\mu, q_{\mu+1}, \ldots, q_m)$. It can be shown that

$$f(\eta,\mu) = \max\left\{\begin{array}{ll} f(\eta+1,\mu) & ||p_{\eta+1}-q_\mu|| \le (1+\epsilon)D_c \\ & (p_\eta \text{ is not matched in this case}) \\ f(\eta,\mu+1) & ||p_\eta - q_{\mu+1}|| \le (1+\epsilon)D_c \\ & (q_\mu \text{ is not matched in this case}) \\ f(\eta+1,\mu+x+1)+1 & \mu \le \mu+x \le m \wedge ||p_\eta - q_{\mu+x}|| \le (1+\epsilon)D_c \\ f(\eta+y+1,\mu+1)+1 & \eta \le \eta+y \le n \wedge ||p_{\eta+y} - q_\mu|| \le (1+\epsilon)D_c \end{array}\right\}$$

The number of $f(\eta,\mu)$ values to be computed in this dynamic programming is $O(nm)$ in the general case, and $O(n|\Delta_\epsilon|)$ for globular proteins. Note that by using the bipartite graphs, it is easy to search for points within distance $(1+\epsilon)D_c$ apart. For each $f(\eta,\mu)$, there are $O(m)$ values from which to find a maximum in the general case, and $O(\Delta_\epsilon)$ values in the case of globular proteins. Thus, the runtime complexity for a single rotation interval is $O(nm^2)$ in the general case, and $O(n{\Delta_\epsilon}^2)$ for globular proteins. Therefore, the total time complexity is $O(n^4m^5/\epsilon^5)$ for the general case, and $O(n^4m^{2.3\bar{3}}{\Delta_\epsilon}^2/\epsilon^5)$ for globular proteins.

Theorem 23. *There is an algorithm of time complexity* $O(n^4m^5/\epsilon^5)$ *for the general case, and* $O(n^4\, m^{2.3\bar{3}}{\Delta_\epsilon}^2/\epsilon^5)$ *for globular proteins, that outputs an* $(\epsilon,1)$*-approximate solution to the sequential LCP problem under bottleneck distance.*

8.3.3 Results for the CMO Problem with Distance Constraints

Combining the rotation interval technique in Section 8.3.1 with the partitioning technique from Xu *et al.* [168] a PTAS is now to be demonstrated for the non-sequential CMO problem with distance constraints under reasonable D_l, D_u and D_c parameters.

In the proposed method, a radial axis of V_1 is matched to a pair in V_2 by using some transformation T. Then, for each rotation $R \in [0, 2\pi)$ along the radial axis, a contact map, (V_1', E_1), is attained from (V_1, E_1) with the position of each $v \in V_1$ replaced by $R(T(\text{pos}(v)))$. Now with the positions of both (V_1', E_1), (V_2, E_2) fixed, An alignment f which maximizes S is to be found. Since the result is equivalent for rotations within the same rotation interval, only one representative R is used for each rotation interval.

The partitioning strategy in Section 3.2 of [168] is used on (V_1', E_1). The partitioning scheme is modified so that it is along all three axes, as in Section 8.3.2. Here, $D = \max\{2D_c, D_u\}$ is used. For each partition R_j at the initial rotation interval, all the $\Delta_\epsilon^{|R_j|}$ possible matchings of the residues in R_j to (V_2, E_2) are enumerated. By reusing the computation for one matching on the matchings with only a single change, an optimal solution for R_j is possible in $O(\Delta_\epsilon^{|R_j|}) = O(\Delta_\epsilon^{O(k_x k_y k_z (\frac{D}{D_l})^3)})$ time. The total time needed for all $\#R_j$ partitions and for all $k_x k_y k_z$ partitioning schemes is $O(\#R_j k_x k_y k_z \Delta_\epsilon^{O(k_x k_y k_z (\frac{D}{D_l})^3)})$.

For the subsequent rotation intervals, first note that the difference between the contact maps of any two consecutive rotation intervals is contained in a single block, and note that any block is shared by, at most, $O(k_x k_y k_z)$ partitions. Hence, for each subsequent rotation interval, the only need is to recompute for these partitions. This is carried out for all $k_x k_y k_z$ partitioning schemes and for all rotation intervals. Thus, $O(nm(k_x k_y k_z)^2 \Delta_\epsilon^{O(k_x k_y k_z (\frac{D}{D_l})^3)})$ time is required for a general protein, and $O(nm^{1/3}(k_x k_y k_z)^2 \Delta_\epsilon^{O(k_x k_y k_z (\frac{D}{D_l})^3)})$ time for the globular proteins. Since $\#R_j \leq n$, the final runtime is $O(nm^{1/3}(k_x k_y k_z)^2 \Delta_\epsilon^{O(k_x k_y k_z (\frac{D}{D_l})^3)})$

Theorem 24. *There is an algorithm of time complexity* $O(n^3 m^3 (k_x k_y k_z)^2 \Delta_\epsilon^{O(k_x k_y k_z (\frac{\max\{2D_c, D_u\}}{D_l})^3)} / \epsilon^5)$ *for general proteins, and* $O(n^3 m^{2.3\bar{3}} (k_x k_y k_z)^2 \Delta_\epsilon^{O(k_x k_y k_z (\frac{\max\{2D_c, D_u\}}{D_l})^3)} / \epsilon^5)$ *for globular proteins that outputs an* $(\epsilon, 1 - 4(\frac{1}{k_x} + \frac{1}{k_y} + \frac{1}{k_z}))$*-approximate solution to the non-sequential CMO problem with distance constraint.*

Although this runtime is not as good as that in Theorem 4 in [168], it shows that the non-sequential CMO problem with distance constraint allows a PTAS when $\frac{\max\{2D_c, D_u\}}{D_l}$ is bounded below a constant. However, even under such a condition, no exact solution to the non-sequential CMO problem with distance constraint can be obtained in polynomial time.

Theorem 25. *The non-sequential CMO problem with distance constraint is NP-hard, even when* $\frac{\max\{2D_c, D_u\}}{D_l}$ *is bounded below a constant.*

Proof. A reduction from the planar 1-in-3-SAT problem [53] is employed. The planarity of the problem allows us to construct a geometrical representation of the 1-in-3-SAT problem where none of the legs cross, to be used as input to the CMO problem. It is assumed that $D_c > D_u$ throughout this proof. As stated earlier, $D_u \geq D_l$. No other assumption is made regarding D_c, D_u, and D_l. That is, $\frac{\max\{2D_c, D_u\}}{D_l}$ can be set to a small constant.

Given an input formula, two sequences of 3D points P and Q are constructed as input to the CMO problem. It is assumed that the optimal solution will have the points in exactly the positions as they assume in the construction. For each clause some points are constructed for both P and Q, where the contact edges form chain-like structures, called *chains*. The points constructed for different clauses are separated by more than D_c, forming no contact edges except for a few end-points, which are explained later in this section using Figure 8.6. Each point in a chain forms contact edges only with its two (one in the case of end-points) immediate neighbors. For each clause, six such chains are constructed for Q, and three such chains are constructed for P. Each chain of the same clause has the same number of points, say, η. A cyclic structure, formed using three additional points (called *pivot points*), connects the six chains for Q through their end-points. One additional pivot point connects the end-points of the three chains of P.

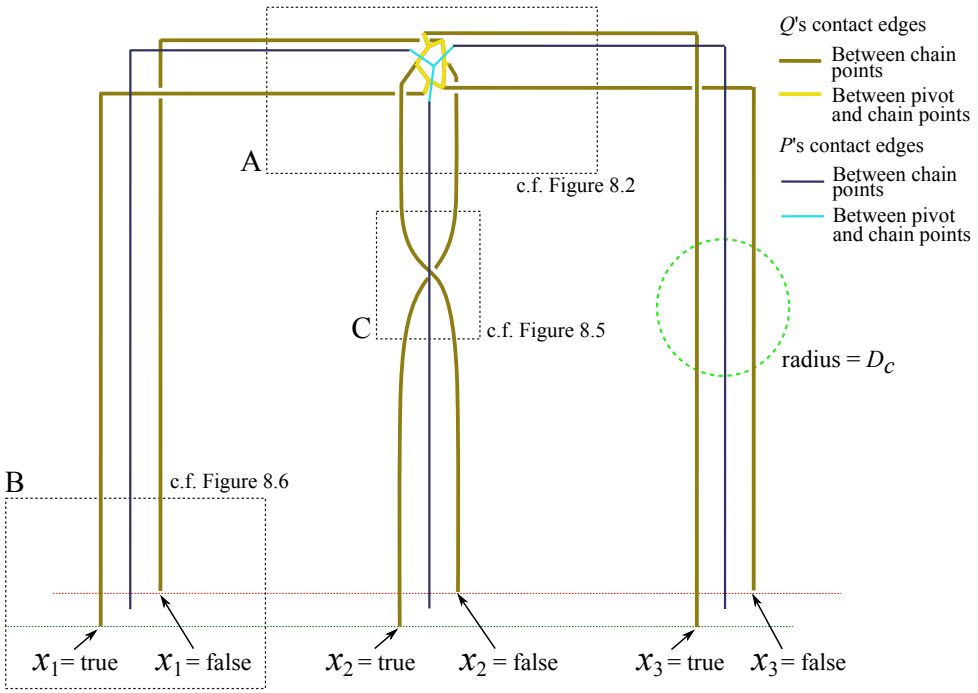


Figure 8.1: Overall view of the construction for the clause $(x_1 \vee x_2 \vee x_3)$

Figure 8.1 shows an overall construction for the clause $(x_1 \vee x_2 \vee x_3)$. Note that in the points constructed for a single clause, no contact edge exists besides those that form the chains, and the connections between the end-points of the chains

to the pivot points. The details at the pivot points (P's elements are not drawn accurately to avoid obstructing the view of Q's elements) are shown in Figure 8.2. They will be explained in detail later.

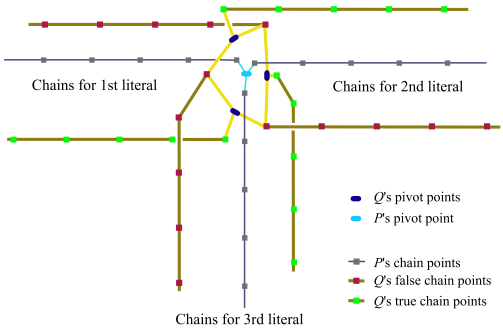


Figure 8.2: Chains connected by the pivot points

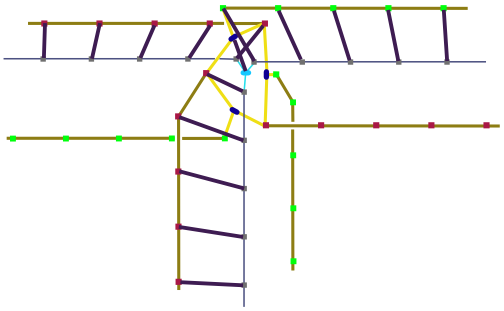


Figure 8.3: Optimal mapping

In the construction of a clause, each literal is represented by using two chains for Q and one chain for P. The CMO problem asks for a mapping between P and Q which maximizes the number of corresponding contact edges in P and Q. To achieve this maximization, the chain for P must be completely mapped to only one of the two chains for Q (i.e., giving $\eta - 1$ corresponding contact edges). The literal is assigned to be true or false, depending on which of the two chains for Q, P's chain is found to mapped to in a CMO output. A Q chain for true assignment is called a *true chain*, and a Q chain for false assignment is called a *false chain*.

The purpose of the the pivot structure is to ensure that exactly one literal in the clause can be assigned to be true. All the contact edges near the structure are illustrated in Figure 8.2. Each false chain is in contact with two pivot points, and each true chain is in contact with one. The end-points of the chains for P need to be more than distance D_u from each other, but each within distance D_c from the points in Q that they may be mapped to in the optimal mappings. Figure 8.4 shows a possible arrangement.

There are exactly three mappings between P and Q which maximizes the num-

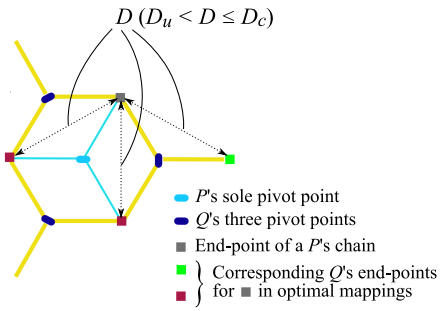


Figure 8.4: Distances between points at the pivots

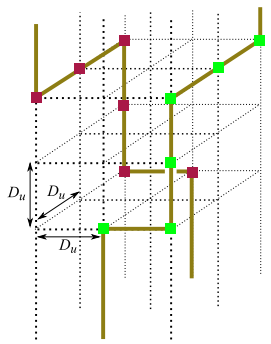


Figure 8.5: Re-positioning chains

ber of corresponding contact edges. Each maps the single pivot point of P to one of the three pivot points of Q. Each mapping gives a total of $3(1 + \eta)$ corresponding contact edges. One such mapping is reflected in Figure 8.3. In all three optimal mappings, only one literal in the clause is assigned as true.

The points at the other end of the chains (i.e., the end not connected to a pivot point) are collected along two lines: one for all the chains representing true assignment to a variable and the other for false assignment. (A true chain does not imply a true assignment to a variable due to negation; that is, a literal of a negated variable should have its true assignment chains along the line for false assignment, and vice versa.) Such positioning might require the two Q chains for a literal to swap positions to connect properly to the pivot. Such swappings can be done through arrangements as show in Figure 8.5.

Chains for the same variable from different clauses are placed at close proximity to each other with a distance of D_u between their consecutive end-points. This is shown in Figure 8.6. The planarity of the problem allows such a construction of the chains, where only the end-points form contact edges. Figure 8.7 shows the arrangement. Note that under this arrangement, the points in P can be mapped to their respective points in Q.

For any variable, an optimal mapping has either all its corresponding chains in P mapped to their corresponding chains in Q which represent true, or to their corresponding chains in Q which represent false. This occurs because such a mapping increases the number of corresponding contact edges through the additional edges at the bottom-most layer; that is, by exactly the number of occurrences of the variable (in the 1-in-3-SAT formula) minus one.

At this point, it is clear that an optimal solution for the CMO problem with distance constraints results in a specific number of corresponding contact edges, which can be calculated from the number of clauses and variable occurrences in the 1-in-3-SAT formula. A solution with this optimal score for the CMO problem corresponds to whether there is a set of variable assignments that fulfills the 1-in-3-SAT formula.

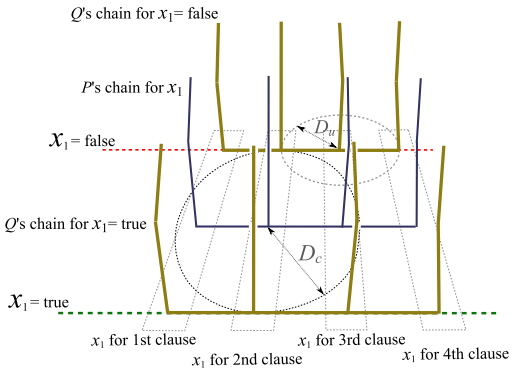


Figure 8.6: Construction for a variable that appears in four clauses

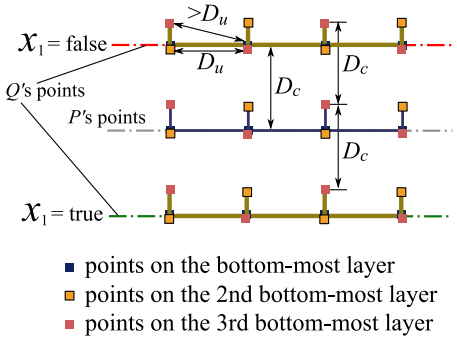


Figure 8.7: View from top of Figure 8.6

It remains to be seen if the number of points needed for constructing P and Q is polynomial with respect to the input size. If α is the minimal number of points needed to construct a single clause, then that the number of points needed to construct an input of n clauses is of order $O(n^2\alpha)$.

Finally, in order to make the positions of the points immutable in an optimal solution in the CMO problem, additional points are added on P and Q (the additional points have only one optimal mapping, and this mapping produces more corresponding edges than the total number of edges in the construction). This can be achieved with, for example, points that form a "grid" on the X-Y plane. These points can be placed sufficiently far away from the points in the construction to avoid forming contact edges with the construction. It is clear that $O(n^2\alpha)$ points suffice to construct these "grid" points.

This polynomial reduction hence witnesses the NP-hardness of the CMO problem with distance constraints. □

Using a slightly different construction, the sequential variant of the problem can be shown to be NP-hard.

Theorem 26. *The sequential CMO problem with distance constraint is NP-hard, even when* $\frac{\max\{2D_c, D_u\}}{D_l}$ *is bounded below a constant.*

8.4 Clique Problem

The clique problem, a known NP-hard problem [57] is used in the proofs of the next two subsections. Let an instance of the clique problem be given by a directed graph $G(V, E)$ and by a positive integer ℓ. Without loss of generality, assume $V = \{1, \ldots, n\}$. For an edge $(u, v) \in E$, $u < v$. In general, the clique problem is defined for undirected graphs. For ease of notation, a linear order is assigned to the vertices, and the edge is assumed to be directed from u to v if $u < v$, where u is referred to as the source vertex of edge (u, v), and v is referred to as the target

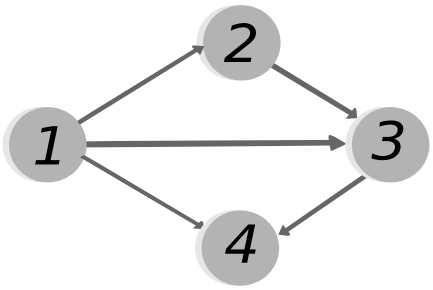


Figure 8.8: Graph G_0 to Illustrate the Reduction.

vertex. An ℓ-clique of a directed graph consists of ℓ vertices: $u_i \in V$, $1 \leq i \leq \ell$ such that $u_1 < u_2 < ... < u_\ell$, and $\forall\ 1 \leq i < j \leq \ell$, $(u_i, u_j) \in E$.

8.5 NP-hardness of CMP-DIS-$\{<, \not\}$

Since the construction is rather complicated, an overview is presented before the details are given. In the construction, a set of contact maps $\mathbb{D}$ is built, then $\mathbb{D}$ is proven to have a DIS-CMP-$\{<, \not\}$ of size $(2\ell-1)n^2+(\ell-1)n+\ell$ iff $G(V, E)$ contains an ℓ-clique. In the reduction, subsets of arcs are constructed to represent edge set E, and these arc sets are denoted $\mathbf{QR}^1$, ..., $\mathbf{QR}^\ell$. For a maximum DIS-CMP-$\{<, \not\}$ pattern,

- ℓ arcs, which correspond to $\ell - 1$ edges in E, from $\mathbf{QR}^1$ are selected,
- $\ell - 1$ arcs from $\mathbf{QR}^2$ (corresponding to $\ell - 2$ edges in E) are selected,
- ...,
- and only one arc is chosen from $\mathbf{QR}^\ell$

The edges corresponding to the arcs, selected from $\mathbf{QR}^1$, are denoted (u_1, u_2), (u_1, u_3), ..., (u_1, u_ℓ), and the edges corresponding to the arcs selected from $\mathbf{QR}^j$, are denoted (u_j, u_{j+1}), (u_j, u_{j+2}), ..., (u_j, u_ℓ), for some u_1, u_2 ..., u_ℓ. If the selection of these arcs are successful, u_1, u_2 ..., u_ℓ form an ℓ-clique.

Graph G_0 in Figure 8.8 illustrates the construction.

In the following subsections, some additional notations are defined first, then the endpoints and the orders between the points are specified, followed by the construction of the arcs, and finally the correctness of the construction is shown.

8.5.1 Additional Notations

A set $\mathbf{D}$ of k distinct arcs, where $\forall a, a' \in \mathbf{D}$, either $a \between a'$ or $a' \between a$, is called a *k-arc crossing cluster*. Given two disjoint sets of arcs, $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_1$ is *nested* in $\mathbf{D}_2$ ($\mathbf{D}_1 \sqsubset \mathbf{D}_2$), iff $\forall a_1 \in \mathbf{D}_1$, $\forall a_2 \in \mathbf{D}_2$, $a_1 \sqsubset a_2$.

Arc a is *propagated* to arc a' if for a maximum DIS-CMP-$\{<, \between\}$ pattern, the selection of a' ensures the selection of a for this DIS-CMP-$\{<, \between\}$. Arc set D is *propagated* to arc set D', $|D| \geq |D'|$, if the selection of D' ensures the selection of D for a maximum DIS-CMP-$\{<, \between\}$. For k DIS-CMP-$\{<, \between\}$ arc sets D_1, ..., D_k with $|D_1| \geq \ldots \geq |D_k|$, the k arc sets are *propagated*, if the selection of D_1 ensures the selections of D_2, ..., D_k for a maximum DIS-CMP-$\{<, \between\}$.

Given two point sets, $\mathcal{S}_1$ and $\mathcal{S}_2$, $\mathcal{S}_1 < \mathcal{S}_2$ iff $\forall s_1 \in \mathcal{S}_1$ and $\forall s_2 \in \mathcal{S}_2$, $s_1 < s_2$.

8.5.2 Set of Endpoints

The following sets of endpoint are constructed: (1) $\mathcal{I}$; (2) $\mathcal{P}^j$, $\mathcal{Q}^j$ and $\mathcal{R}^j$, $1 \leq j \leq \ell$; and (3) $\mathcal{S}^j$, $\mathcal{T}^j$ and $\mathcal{U}^j$, $2 \leq j \leq \ell$.

The details of the construction are as follows.

- $\mathcal{I}$ contains ℓ points, they are ordered according to their subscripts in an increasing order. $\mathcal{I} = \{I_j | 1 \leq j \leq \ell\}, I_1 < \ldots < I_\ell$.
- $\mathcal{P}^j = \{P^j_{u,v} | 1 \leq u \leq n, 1 \leq v \leq n\}$, $1 \leq j \leq \ell$. $\mathcal{P}^j$ contains n^2 points, and those points are ordered, according to their first subscripts, increasingly, and then, according to the second subscripts increasingly: (1) if $u_1 < u_2$, then $P^j_{u_1,v_1} < P^j_{u_2,v_2}$ or (2) if $u_1 = u_2$ and $v_1 < v_2$ then $P^j_{u_1,v_1} < P^j_{u_2,v_2}$.
- $\mathcal{Q}^j = \{Q^j_{u,v} | 1 \leq u \leq n, 1 \leq v \leq p\}$, $1 \leq j \leq \ell$. $\mathcal{Q}^j$ contains n^2 points. The order relation is similar to that for the case of $\mathcal{P}^j$; that is $Q^j_{u_1,v_1} < Q^j_{u_2,v_2}$ (1) if $u_1 < u_2$ or (2) if $u_1 = u_2$ and $v_1 < v_2$.
- $\mathcal{R}^j = \{R^j_{u,v} | 1 \leq u \leq n, 0 \leq v \leq n\}$, $1 \leq j \leq \ell$. $\mathcal{R}^j$ has $n^2 + n$ elements. The elements are ordered, according to the first subscripts, decreasingly, and then, according to the second subscripts, increasingly; that is, $R^j_{u_1,v_1} < R^j_{u_2,v_2}$ if (1) $u_1 > u_2$ or (2) $u_1 = u_2$ and $v_1 < v_2$. Note that case (1) is different from the case (1) of $\mathcal{P}^j$ and $\mathcal{Q}^j$.
- $\mathcal{S}^j = \{S^j_{u,v} | 1 \leq u \leq n, 1 \leq v \leq n\}$, $2 \leq j \leq \ell$. $\mathcal{S}$ has n^2 elements. Its order relation is similar to that for the case of $\mathcal{R}^j$; that is, (1) if $u_1 < u_2$, $S^j_{u_1,v_1} > S^j_{u_2,v_2}$ or (2) if $u_1 = u_2$ and $v_1 < v_2$, $S^j_{u_1,v_1} < S^j_{u_2,v_2}$.
- $\mathcal{T}^j = \{T^j_u | 1 \leq u \leq n\}$, $2 \leq j \leq \ell$. $\mathcal{T}^j$ contains n points, and the elements are ordered according to the subscripts, increasingly; that is, $T^j_{u_1} < T^j_{u_2}$ if $u_1 < u_2$.

- $\mathcal{U}^j = \{U^i_u | 1 \leq u \leq n\}$, $2 \leq j \leq \ell$. $\mathcal{U}^j$ contains n points, and the elements are ordered the same way as those in $\mathcal{T}^j$; that is, $U^j_{u_1} < U^j_{u_2}$ if $u_1 < u_2$.

Furthermore, the following order is specified,

$$\mathcal{P}_1 < \mathcal{Q}_1 < \mathcal{R}_1$$
$$\mathcal{S}_j < \mathcal{T}_j < \mathcal{U}_j < \mathcal{P}_j < \mathcal{Q}_j < \mathcal{R}_j, 2 \leq j \leq \ell.$$

$\mathcal{W}_j$s, $2 \leq j \leq \ell$, is defined as $\mathcal{W}_1 = \mathcal{P}_1 \cup \mathcal{Q}_1 \cup \mathcal{R}_1$ and $\mathcal{W}_j = \mathcal{S}_j \cup \mathcal{T}_j \cup \mathcal{U}_j \cup \mathcal{P}_j \cup \mathcal{Q}_j \cup \mathcal{R}_j$ for $2 \leq j \leq \ell$.

The following order is introduced,

$$\mathcal{I} < \mathcal{W}^1 < \mathcal{W}^2 < \ldots < \mathcal{W}^\ell.$$

Hence a total order of all the points has been specified. $\mathbb{S}$ denotes the set of points that have been created. $\mathbb{S} = \mathcal{I} \cup \bigcup_{j=1}^{\ell} \mathcal{W}_j$.

8.5.3 Construction of the Arcs

In this subsection, the arcs construction is described.

Arc Set IP

One arc is created between each pair of points, one point from $\mathcal{I}$, and the other from $\mathcal{P}^1$ as in Figure 8.9.

$$\mathbf{IP} = \{(I_j, P^1_{u,v}) | 1 \leq j \leq \ell, 1 \leq u, v \leq n\}.$$

This construction ensures that at most ℓ arcs of **IP** can occur in a DIS-CMP-$\{<, \between\}$ pattern, since $\mathcal{I}$ contains ℓ points, and no arcs in a DIS-CMP-$\{<, \between\}$ pattern can share an endpoint.

Arc Set $\mathbf{PQ}^j, 1 \leq j \leq \ell$

There are n^2 arcs in $\mathbf{PQ}^j$. Each arc is created between a pair of points, one from $P^j_{u,v}$ and one from $Q^j_{u,v}$, that shares the same subscript; that is,

$$\mathbf{PQ}^j = \{(P^j_{u,v}, Q^j_{u,v}) | 1 \leq u \leq n, 1 \leq v \leq n\}.$$

All the arcs in $\mathbf{PQ}^j$ cross each other as depicted in Figure 8.9 and Figure 8.12. Any combination of the arcs in $\mathbf{PQ}^j$ can occur in a DIS-CMP-$\{<, \between\}$.

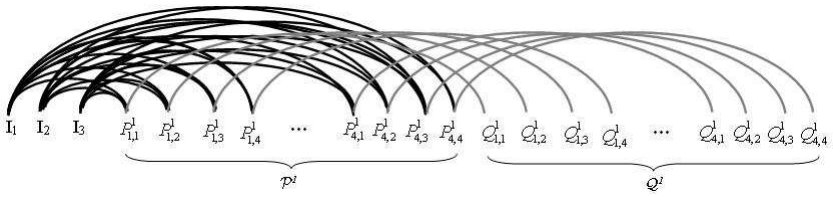


Figure 8.9: Arc Sets **IP** and $\mathbf{PQ}^1$ for graph G_0.
IP is a full bipartite connection between $\mathcal{I}$ and $\mathcal{P}^j$. $\mathbf{PQ}^j$ is an n^2-arc crossing cluster.

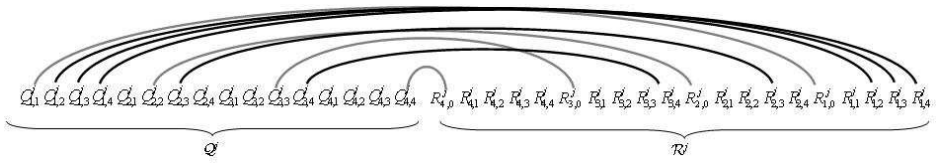


Figure 8.10: Arc Set $\mathbf{QR}^j$ for G_0. $\mathbf{QR}^j$ codes the edge information.
$\mathbf{QR}^\ell$ contains only the anchor arcs.

Arc Set $\mathbf{QR}^j, 1 \leq j \leq \ell$

$\mathbf{QR}^j$ is the place to code the edge information. $\mathbf{QR}^j$, $1 \leq j \leq \ell - 1$, contains $|E| + n$ arcs, and $\mathbf{QR}^\ell$ contains n arcs. For $1 \leq j \leq \ell$ and $1 \leq u \leq n$, an arc is created between $Q^j_{u,u}$ and $R^j_{u,0}$, and for $1 \leq j \leq \ell - 1$, there is an arc between $Q^j_{u,v}$ and $R^j_{u,v}$, if and only if (u, v) is an edge of G, $u < v$; that is

$$\mathbf{QR}^j = \{(Q^j_{u,v}, R^j_{u,v}) | (u, v) \in E\} \cup \{(Q^j_{u,u}, R^j_{u,0}) | 1 \leq u \leq n\}, 1 \leq j \leq \ell - 1$$
$$\mathbf{QR}^\ell = \{(Q^\ell_{u,u}, R^\ell_{u,0}) | 1 \leq u \leq n\}.$$

$\mathbf{QR}^j_u$ is a subset of $\mathbf{QR}^j$, $\mathbf{QR}^j_u = \{(Q^j_{u,v}, R^j_{u,v}) | (Q^j_{u,v}, R^j_{u,v}) \in \mathbf{QR}^j, 0 \leq v \leq n\}$.

$\mathbf{QR}^j_u$ is a crossing cluster. As the elements in $\mathcal{Q}^j$ are ordered increasingly according to the second subscripts, and the elements in $\mathcal{R}^j$ are ordered decreasingly, according to the second subscripts, $\mathbf{QR}^j_{u_1}$ is nested in $\mathbf{QR}^j_{u_2}$,$1 \leq u_2 < u_1 \leq n$ (Figure 8.10), stated in the following:

Lemma 27. $\mathbf{QR}^j_{u_1} \sqsubset \mathbf{QR}^j_{u_2}, 1 \leq u_2 < u_1 \leq n$.

This property ensures that only those arcs in $\mathbf{QR}^j$, whose endpoints share the same first subscripts can occur in a DIS-CMP-$\{<, \between\}$. This implies that only the edges with the same source node (denoted by the first subscripts) can have their corresponding arcs occurring in a DIS-CMP-$\{<, \between\}$.

The arc $(Q^j_{u,u}, R^j_{u,0})$ is called the **anchor** of $\mathbf{QR}^j_u$. The anchor $(Q^j_{u,u}, R^j_{u,0})$ crosses the rest of the arcs in $\mathbf{QR}^j_u$. The anchor arcs in $\mathbf{QR}^j$ are nested with each other, and at most one anchor arc can occur in a DIS-CMP-$\{<, \between\}$.

In Section 8.5.4, it will be proven that to attain a maximum DIS-CMP-$\{<, \between\}$, one of the anchor arcs must be selected for each $\mathbf{QR}^j_u$, $1 \leq j \leq \ell$.

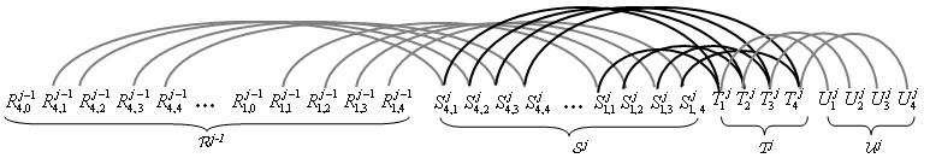


Figure 8.11: Arc sets $\mathbf{RS}^{j-1}$, $\mathbf{ST}^j$, and $\mathbf{TU}^j$ for G_0.
$\mathbf{RS}^{j-1}$ is an n^2-arc crossing cluster. Every n arcs in $\mathbf{ST}^j$ share one endpoint in $\mathcal{T}^j$. $\mathbf{TU}^j$ is an n-arc crossing cluster.

By the proposed construction, at most one arc from $\mathbf{QR}^\ell$ can be selected for a maximum DIS-CMP-$\{<, \between\}$.

Lemma 28. *At most one arc from* $\mathbf{QR}^\ell$ *can be selected for a maximum DIS-CMP-*$\{<, \between\}$.

Arc Set $\mathbf{RS}^j, 2 \le j \le \ell$

$\mathbf{RS}^j$ contains n^2 arcs, whose construction is similar to that of $\mathbf{PQ}^j$, as in Figure 8.11,

$$\mathbf{RS}^j = \{(R^j_{u,v}, S^{j+1}_{u,v}) \mid 1 \le u \le n, 1 \le v \le n\}.$$

$\mathbf{RS}^j$ is an n^2-arc crossing cluster, and any combination of the arcs can occur in a DIS-CMP-$\{<, \between\}$.

Arc Set $\mathbf{ST}^j, 2 \le j \le \ell$

$\mathbf{ST}^j$ contains n^2 arcs, and every n arcs in $\mathbf{ST}^j$ share one endpoint in $\mathcal{T}^j$ as in Figure 8.11; namely,

$$\mathbf{ST}^j = \{(S^j_{u,v}, T^j_v) | 1 \le u \le n, 1 \le v \le n\}$$

At most n arcs in $\mathbf{ST}^j$ can occur in a DIS-CMP-$\{<, \between\}$.

Arc Set $\mathbf{TU}^j, 2 \le j \le \ell$

$\mathbf{TU}^j$ is an n-arc crossing cluster as in Figure 8.11. An arc is created between two points of the same subscripts with one point in $\mathcal{T}^j$ and the other point in $\mathcal{U}^j$. Therefore,

$$\mathbf{TU}^j = \{(T^j_v, U^j_v) | 1 \le v \le n\}.$$

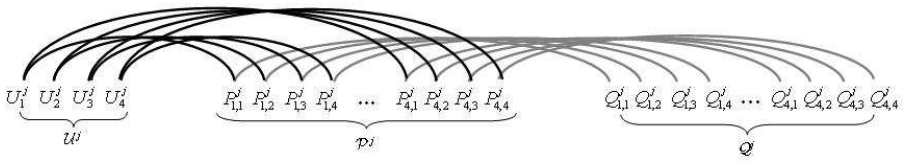


Figure 8.12: Arc Set $\mathbf{UP}^j$ and $\mathbf{PQ}^j$ for G_0.
Each n arcs, $\mathbf{UP}^j$, share point $\mathcal{U}^j$. $\mathbf{PQ}^j$ is a n^2-arc crossing cluster.

Arc Set $\mathbf{UP}^j, 2 \leq j \leq \ell$

Each n arcs in $\mathbf{UP}^j$ share one endpoint in $\mathcal{U}^j$ (Figure 8.12), expressed as

$$\mathbf{UP}^j = \{(U_v^j, P_{u,v}^j) | 1 \leq u \leq n, 1 \leq v \leq n\}.$$

At most n arcs of $\mathbf{UP}^j$ can appear in a DIS-CMP-$\{<, \between\}$.

$\mathbf{A}^j$s, $1 \leq j \leq \ell$, are defined as $\mathbf{A}^1 = \mathbf{IP} \cup \mathbf{PQ}^1 \cup \mathbf{QR}^1$ and $\mathbf{A}^j = \mathbf{QR}^{j-1} \cup \mathbf{RS}^j \cup \mathbf{ST}^j \cup \mathbf{TU}^j \cup \mathbf{UP}^j \cup \mathbf{PQ}^j \cup \mathbf{QR}^j$, $2 \leq j \leq \ell$. The set of all the arcs constructed is represented by $\mathbb{D}$.

8.5.4 Correctness of the Construction

Define $\mathcal{L} = (2\ell - 1)(n^2 + 1) + (\ell - 1)n + \ell$. First, the task is to prove that to obtain a maximum DIS-CMP-$\{<, \between\}$ $\mathbb{P}$ of size $\mathcal{L}$ in $\mathbb{D}$, the only way is to select ℓ arcs from $\mathbf{IP}$, and $\ell - j + 1$ arcs from $\mathbf{QR}^j$, $1 \leq j \leq \ell$. Secondly, by using the edge information coded in $\mathbf{QR}^j$, it is proven that the edges corresponding to the arcs selected from $\mathbf{QR}^j$, $1 \leq j \leq \ell$, form a clique.

Theorem 29. *There is a maximum DIS-CMP-$\{<, \between\}$ $\mathbb{P}$ in $\mathbb{D}$ of size $\mathcal{L}$, if and only if $\mathbb{P}$ contains ℓ arcs from $\mathbf{IP}$, and $\ell - j + 1$ arcs from $\mathbf{QR}^j$, $1 \leq j \leq \ell$.*

Furthermore, the arcs of $\mathbb{P}$ contain arcs from $\mathbf{IP}$, which are $(I_1, P^1_{u_1,u_1})$,$(I_2, P^1_{u_1,u_2})$, ..., $(I_\ell, P^1_{u_1,u_\ell})$, and contain arcs from $\mathbf{QR}^j$, which are $(Q^j_{u_j,u_j}, R^j_{u_f j,0})$, $(Q^j_{u_j,u_2}, R^j_{u_j,u_2})$, ..., $(Q^j_{u_j,u_\ell}, R^j_{u_j,u_\ell})$, for some $u_1, u_2, \ldots, u_\ell$ with $1 \leq u_1 < u_2 < \ldots < u_\ell \leq n$.

The proof consists of three steps. First, in $\mathbf{A}^1$, the arcs selected from $\mathbf{IP}$ are propagated to the arcs selected from $\mathbf{QR}^1$. Secondly, in $\mathbf{A}^j$, the arcs selected from $\mathbf{QR}^{j-1}$ are propagated to the arcs from $\mathbf{QR}^j$. Thirdly, by combining the first two steps, the arcs selected from $\mathbf{IP}$ and $\mathbf{QR}^j$, $1 \leq j \leq \ell$, are all propagated.

First, the arcs, selected from $\mathbf{IP}$ and from $\mathbf{QR}^1$, are propagated. An example is given in Figure 8.13.

Lemma 30. *If k_0 arcs are selected from $\mathbf{IP}$ and k_1 arcs are selected from $\mathbf{QR}^1$ for $\mathbb{P}$, then $n^2 + \min\{k_0, k_1\}$ arcs are selected from $\mathbf{A}^1$ for $\mathbb{P}$.*

Furthermore, if $k_0 = k_1$ and the number of arcs selected from $\mathbf{A}^1$ is $n^2 + k_0$, then the arcs selected from $\mathbf{QR}^1$ have endpoints in $\mathcal{Q}^1$ as Q^1_{u,u_1}, ..., $Q^1_{u,u_{k_0}}$ and the

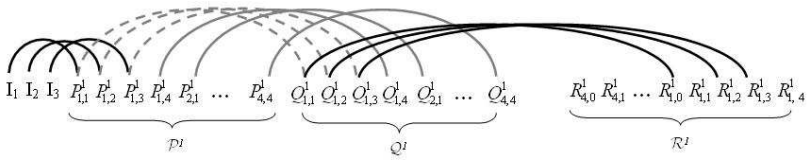


Figure 8.13: Arc propagation for $\mathbf{A}^1$.

arcs selected from **IP** *have their endpoints in* $\mathcal{P}^1$ *as* P^1_{u,u_1}, ..., $P^1_{u,u_{k_0}}$ *for some* u, u_1, ..., u_{k_0}, $1 \leq u_1 < \ldots < u_{k_0} \leq n$.

Proof. k_0 arcs from **IP** imply that at most $n^2 - k_0$ points in $\mathcal{P}^1$ can be used for the arcs from $\mathbf{PQ}^1$. k_1 arcs from $\mathbf{QR}^1$ imply that at most $n^2 - k_1$ points in $\mathcal{Q}^1$ can be applied for the arcs selected from $\mathbf{PQ}^1$. $\mathbf{PQ}^1$ is an n^2 crossing cluster and at least $\max\{k_0, k_1\}$ arcs in $\mathbf{PQ}^1$ share endpoints with arcs from **IP** or $\mathbf{QR}^1$. Therefore, the number of arcs that are selected from $\mathbf{A}^1$ is, at most, $k_0 + k_1 + n^2 - \max\{k_0, k_1\}$, or equivalently, $n^2 + \min\{k_0, k_1\}$.

If $k_0 = k_1$ and the number of arcs selected from $\mathbf{A}^1$ is $n^2 + k_0$, the maximum possible number of arcs that can be selected from $\mathbf{A}^1$ is achieved. This maximum value is achievable, if and only if the following two statement are valid,

- the number of arcs from $\mathbf{PQ}^1$ is $n^2 - k_0$, and
- the subscripts for the right endpoints of the k_0 arcs from **IP** have a one-to-one correspondence with the subscripts for the left endpoints of the k_0 arcs from $\mathbf{QR}^1$.

According to Lemma 27, the first subscripts for the endpoints of the arcs from $\mathbf{QR}^1$ are the same. As a result, the statement holds. □

The arcs selected from $\mathbf{QR}^{j-1}$ and $\mathbf{QR}^j$ are propagated, as shown by the following lemma.

Lemma 31. *If* k_{j-1} *arcs are selected from* $\mathbf{QR}^{j-1}$ *and* k_j *arcs are selected from* $\mathbf{QR}^j$ *for* $\mathbb{P}$, *then* $2n^2 + n + \min\{k_{j-1}, k_j + 1\}$ *arcs are selected from* $\mathbf{A}^j$ *for* $\mathbb{P}$.

Furthermore, if $k_{j-1} = k_j + 1$ *and the number of arcs from* $\mathbf{A}^j$ *is* $2n^2 + n + k_{j-1}$,*then arcs selected from* $\mathbf{QR}^{j-1}$, *have endpoints in* $\mathcal{R}^{j-1}$, *as* $R^{j-1}_{u,0}$, R^{j-1}_{u,u_1}, R^{j-1}_{u,u_2} ..., $R^{j-1}_{u,u_{k_j}}$ *and the arcs, selected from* $\mathbf{QR}^j$, *have their endpoints in* $\mathcal{Q}^j$ *as* Q^j_{u',u_1}, Q^j_{u',u_2}, ..., $Q^j_{u',u_{k_j}}$ *for some* u, u', u_1, u_2, ..., u_{k_j} *with* $1 \leq u_1 < u_2 < \ldots < u_{k_j} \leq n$.

Proof. The arc set $\mathbf{QR}^{j-1} \cup \mathbf{RS}^j \cup \mathbf{ST}^j$ is considered. If s arcs are selected from $\mathbf{ST}^j$, and t arcs are selected from $\mathbf{UP}^j$, then like in the argument in Lemma 30, the number of arcs selected from $\mathbf{RS}^j$ is $n^2 - \max\{k_{j-1} - 1, s\}$. This is due to that at most one of the anchor arcs in $\mathbf{QR}^{j-1}$ is selected and the anchor arc does not

share the endpoints with the arcs in $\mathbf{RS}^j$. Similarly, the numbers of arcs that can be selected from $\mathbf{TU}^j$ and $\mathbf{PQ}^j$ are $n-\max\{s,t\}$ and $n^2-\max\{t,k_j\}$, respectively. Thus the number of arcs can be selected from $\mathbf{A}^j$ is

$$2n^2+n+k_{j-1}+k_j+s+t-\max\{k_{j-1}-1,s\}-\max\{s,t\}-\max\{t,k_j\}. \quad (8.1)$$

Equation 8.1 is rearranged as

$$\begin{aligned} & 2n^2+n+k_{j-1}+\min\{s-k_{j-1}+1,0\}+\min\{s-t,0\}+\min\{k_j-t,0\} \\ \leq\ & 2n^2+n+k_{j-1} \end{aligned}$$

The maximum $2n^2+n+k_{j-1}$ is achievable, only if (1) $s+1\geq k_{j-1}$; (2) $s\geq t$; and (3) $k_j\geq t$.

Similarly, Equation 8.1 is written as

$$\begin{aligned} & 2n^2+n+k_j+1+\min\{0,k_{j-1}-s-1\}+\min\{s-t,0\}+\min\{0,t-k_j\} \\ \leq\ & 2n^2+n+k_j+1. \end{aligned}$$

The maximum $2n^2+n+k_j+1$ is achievable, only if (4) $k_{j-1}\geq s+1$; (5) $s\geq t$; and (6) $t\geq k_j$.

It can be verified that $2n^2+n+\min\{k_{j-1},k_j+1\}$ arcs are selected from $\mathbf{A}^l$ for $\mathbb{P}$.

If $k_{j-1}-1=k_j$, to maximize Equation 8.1, by combining condition (1)-(6), $s=t=k_j$. It is known that one anchor arc in $\mathbf{QR}^{j-1}$ is selected. Let the endpoints of the arcs from $\mathbf{QR}^{j-1}$ in $\mathcal{R}^{j-1}$ be $R^{j-1}_{u,0}$, R^{j-1}_{u,u_1}, R^{j-1}_{u,u_2} ..., $R^{j-1}_{u,u_{k_j}}$, for some u, u_1, u_2, ..., u_{k_j}, with $1\leq u_1<u_2<\ldots<u_{k_j}\leq n$. Given arc set $\mathbf{QR}^{j-1}\cup\mathbf{RS}^j\cup\mathbf{ST}^j$, the s arcs from $\mathbf{ST}^j$ have their endpoints in $\mathcal{T}^j$ as $T^j_{u_1}$, $T^j_{u_2}$ ..., $T^j_{u_{k_j}}$. Since the left endpoints of the arcs from $\mathbf{QR}^j$ have the same first subscripts by Lemma 28, the t arcs from $\mathbf{UP}^j$ have their endpoints in $\mathcal{P}^j$ as P^j_{u',u_1}, P^j_{u',u_2} ..., $P^j_{u',u_{k_j}}$ for some u', $1\leq u'\leq n$. This implies that the arcs, selected from $\mathbf{QR}^j$, have their endpoints in $\mathcal{Q}^j$ as Q^j_{u',u_1}, Q^j_{u',u_2}, ..., $Q^j_{u',u_{k_j}}$. □

To achieve a maximum pattern, it will be proved that one anchor arc must be selected for each arc set $\mathbf{QR}^j$, $1\leq j\leq\ell$, and this implies that $u'=u_1$ in Lemma 31.

By combining the results from Lemma 30 and Lemma 31, the arcs selected, from $\mathbf{IP}$, and $\mathbf{QR}^j$, $1\leq j\leq\ell$, are propagated, which results in Theorem 29.

Proof. (of Theorem 29) If k_0 arcs are selected from $\mathbf{IP}$, and k_j arcs are selected from $\mathbf{QR}^j$ for $\mathbb{P}$. Lemma 31 implies that at most $2n^2+n+\min\{k_{j-1},k_j+1\}-k_{j-1}$

arcs can be selected from $\mathbf{A}^j - \mathbf{QR}^{j-1}$. This DIS-CMP-$\{<, \emptyset\}$ has the following size,

$$\begin{aligned} & n^2 + \min\{k_0, k_1\} + \sum_{j=2}^{\ell}(2n^2 + n + \min\{k_{j-1}, k_j + 1\}) - \sum_{j=2}^{\ell} k_{j-1} \\ = \;& (2\ell - 1)n^2 + (\ell - 1)n + \min\{k_0, k_1\} + \sum_{j=2}^{\ell} \min\{k_j + 1 - k_{j-1}, 0\} \\ \leq \;& (2\ell - 1)n^2 + (\ell - 1)n + k_0 \\ \leq \;& \mathcal{L}. \end{aligned}$$

The value is maximized, only if (1) $k_0 \leq \ell$; (2) $k_1 \geq \ell$, and $k_{j-1} \leq k_j + 1$ for $j \geq 2$.

By Lemma 28, it is known that the maximum value for k_ℓ is 1. So, $k_0 = \ell$, and $k_j = \ell - j + 1$ for $1 \leq j \leq \ell$.

If the arcs from **IP** are $(I_1, P^1_{u_1,u_1})$,$(I_2, P^1_{u_1,u_2})$, ..., $(I_\ell, P^1_{u_1,u_\ell})$ for some $u_1, u_2, \ldots, u_\ell$ with $1 \leq u_1 < u_2 < \ldots < u_\ell \leq n$. It is known that one anchor arc is selected from $\mathbf{QR}^1$ to achieve the maximum value. Also, the two subscripts of the left endpoint for an anchor arc are equal. According to Lemma 30, the arcs from $\mathbf{QR}^1$ are $(Q^1_{u_1,u_1}, R^1_{u_1,0})$, $(Q^1_{u_1,u_2}, R^1_{u_1,u_2})$, ..., $(Q^1_{u_1,u_\ell}, R^1_{u_1,u_\ell})$. By similar arguments, the arcs from $\mathbf{QR}^j$ are $(Q^j_{u_j,u_j}, R^j_{u_j,0})$, $(Q^j_{u_j,u_2}, R^j_{u_j,u_2})$, ..., $(Q^j_{u_j,u_\ell}, R^j_{u_j,u_\ell})$ $(2 \leq j \leq \ell)$, according to Lemma 31. □

Lastly, according to the edge information coded in the arcs, selected from $\mathbf{QR}^j$, $1 \leq j \leq \ell$, it is demonstrated that a clique of size ℓ exists, if there is a DIS-CMP-$\{<, \emptyset\}$ size $\mathcal{L}$ in $\mathbb{D}$.

Lemma 32. *If there is a DIS-CMP-$\{<, \emptyset\}$ $\mathbb{P}$ of size $\mathcal{L}$ in $\mathbb{D}$, then there is an ℓ-clique in G.*

Proof. From Theorem 29, to establish a size $\mathcal{L}$ DIS-CMP-$\{<, \emptyset\}$, the arcs selected from $\mathbf{QR}^j$, $1 \leq j \leq \ell$, must be $(Q^j_{u_j,u_j}, R^j_{u_j,0})$, $(Q^j_{u_j,u_{j+1}}, R^j_{u_j,u_{j+1}})$, ..., $(Q^j_{u_j,u_\ell}, R^j_{u_j,u_\ell})$. Then, the j arcs selected from $\mathbf{QR}^j$ imply that E contains edges (u_j, u_{j+1}), $\ldots (u_j, u_\ell)$. This implies that all the edges (v_1, v_2), $v_1 < v_2$, $v_1, v_2 \in \{u_1, u_2, \ldots, u_\ell\}$ are in E. Therefore, $u_1, \ldots, u_\ell$ form a clique of size ℓ. □

It is easy to prove that if G contains an ℓ-clique, then a maximum DIS-CMP-$\{<, \emptyset\}$ in $\mathbb{D}$ has size $\mathcal{L}$. It is obvious that this reduction is polynomial. Inherently, the following is established.

Theorem 33. *$\mathbb{D}$ has a size $\mathcal{L}$ DIS-CMP-$\{<, \emptyset\}$, if and only if G has a clique of size ℓ; hence, the DIS-CMP-$\{<, \emptyset\}$ problem is NP-hard.*

8.6 NP-hardness of the Crossing Pattern Matching Problem

In the following, some terms are defined to facilitate the presentation of the reduction. Given a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we construct

- a target map $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{D}_\mathcal{G})$, and
- a pattern $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ with parameters ℓ and n.

Lastly, the reduction is analyzed and its correctness is shown.

8.6.1 Additional Notations and Definitions

A set $\mathcal{D}$ of k distinct arcs, where $\forall a, a' \in \mathcal{D}$, either $a \between a'$ or $a' \between a$, is called a *k-arc crossing cluster*. Let $\mathcal{D}_1$ and $\mathcal{D}_2$ denote two arc sets. $\mathcal{D}_1$ *crosses* $\mathcal{D}_2$, or $\mathcal{D}_2$ *is crossed by* $\mathcal{D}_1$ (written $\mathcal{D}_1 \between \mathcal{D}_2$), if either (1) $\forall a_1 \in \mathcal{D}_1$, $\forall a_2 \in \mathcal{D}_2$, $a_1 \between a_2$, or (2) if one of $\mathcal{D}_1$ or $\mathcal{D}_2$ is an empty set. $\mathcal{D}_1 < \mathcal{D}_2$ ($\mathcal{D}_1$ is *less than* $\mathcal{D}_2$, or $\mathcal{D}_2$ is *greater than* $\mathcal{D}_1$), and $\mathcal{D}_1 \sqsubset \mathcal{D}_2$ ($\mathcal{D}_1$ is *nested* in $\mathcal{D}_2$) can be defined similarly. Also, an arc a *crosses* a set of arcs $\mathcal{D}$ if $\{a\} \between \mathcal{D}$ (the cases for $\sqsubset$ and $<$ can be defined similarly).

For any three sets of arcs, $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, the following terms are defined.

- $\mathcal{D}_3$ is *from* $\mathcal{D}_1$ *to* $\mathcal{D}_2$, iff $\mathcal{D}_1 < \mathcal{D}_2$ and $\mathcal{D}_1 \between \mathcal{D}_3$, $\mathcal{D}_3 \between \mathcal{D}_2$, and
- $\mathcal{D}_3$ is *anchored* by $\mathcal{D}_1$ and $\mathcal{D}_2$, iff $\mathcal{D}_1 < \mathcal{D}_3$ and $\mathcal{D}_2 \between \mathcal{D}_3$.

Given two point sets, $\mathcal{S}_1$ and $\mathcal{S}_2$, $\mathcal{S}_1 < \mathcal{S}_2$ iff $\forall s_1 \in \mathcal{S}_1$ and $\forall s_2 \in \mathcal{S}_2$, $s_1 < s_2$. For an arc set $\mathcal{D}$, $L(\mathcal{D}) = \bigcup_{a \in \mathcal{D}} \{L(a)\}$, and $R(\mathcal{D}) = \bigcup_{a \in \mathcal{D}} \{R(a)\}$.

The subscript, '$*$', is a special symbol which matches each defined subscript; that is, $A_{*,j}$ refers to the set $\{A_{ij} | A_{i,j}$ is defined, and j is fixed$\}$, and $A_{*,*}$ refers to the set of all $A_{i,j}$ that have been defined.

If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{D}_\mathcal{G})$, a one-to-one mapping $\mathcal{M}$ exists between the elements in $\mathcal{D}_{n,\ell}$ and some elements in $\mathcal{D}_\mathcal{G}$. Here, the definition of the mapping is extended to any set $\mathcal{D}'_p \subseteq \mathcal{D}_{n,\ell}$ such that $\mathcal{M}(\mathcal{D}'_p) = \bigcup_{a \in \mathcal{D}'_p} \{\mathcal{M}(a)\}$.

8.6.2 Target Contact Map Construction

In this section, a target contact map $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{D}_\mathcal{G})$ is built for a given graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Some large crossing clusters are built first, the arcs which connect these clusters are then constructed.

Large Crossing Clusters

$2n+2$ crossing clusters are constructed, which are H, Z_u $(1 \leq u \leq n)$, T, and V_u $(1 \leq u \leq n)$. H is a $28n^4$-arc crossing cluster, Z_u is a $5n^3$-arc crossing cluster, T is a $9n^4$-arc crossing cluster, and V_u is a $5n^3$-arc crossing cluster. Z and V are defined as $Z = \bigcup_{u=1}^{n} Z_u$ and $V = \bigcup_{i=1}^{n} V_u$. Furthermore, the following order is defined for these large clusters,

$$H < Z_1 < \ldots < Z_n < T < V_1 < \ldots < V_n. \tag{8.2}$$

Arcs from H to Z_u

There is a 2-arc crossing cluster from $\mathbf{H}$ to $\mathbf{Z}_u$ for each u, $1 \leq u \leq n$. Signify the two arcs as $A_{u,1}$, and $A_{u,2}$, $A_{u,1} \between A_{u,2}$. Let $A_u = \{A_{u,1}, A_{u,2}\}$. Furthermore, the following orders are defined,

$$H \between A_u, A_u \between Z_u \qquad 1 \leq u \leq n \tag{8.3}$$

$$A_{u_1} \sqsubset A_{u_2}, \qquad 1 \leq u_1 < u_2 \leq n. \tag{8.4}$$

Equation 8.3 ensures that A_u is from H to Z_u and Equation 8.4 ensures that at most one pair of arcs in $A_{*,*}$ can be included in a CCM.

If $A = \bigcup_{u=1}^{n} A_u$, it is clear that $|A| = 2n$.

Arcs from Z_u to Z_v

There are two kinds of arcs from Z_u to Z_v $(1 \leq u < v \leq n)$: $E_{u,v}$ and $C_{u,v}$. $E_{u,v}$ consists of u crossing clusters, $E_{u,v,w}$, $1 \leq w \leq u$. Each cluster $E_{u,v,w}$ contains three arcs, respectively, $E_{u,v,w,1}$, $E_{u,v,w,2}$ and $E_{u,v,w,3}$ with $E_{u,v,w,1} \between E_{u,v,w,2}$, $E_{u,v,w,1} \between E_{u,v,w,3}$ and $E_{u,v,w,2} \between E_{u,v,w,3}$. Each $C_{u,v}$ is a single arc. The orders among the arcs $E_{*,*,*,*}$ and $C_{*,*}$ which are needed for our proof are now defined.

The following orders ensure that $E_{u,*,*,*}$ and $C_{u,*}$ are crossed by Z_u, while $E_{*,v,*,*}$ and $C_{*,v}$ crosses Z_v.

$$Z_u \between E_{u,*,*,*}, Z_u \between C_{u,*}, \qquad 1 \leq u \leq n-1 \tag{8.5}$$

$$E_{*,v,*,*} \between Z_v, C_{*,v} \between Z_v, \qquad 2 \leq v \leq n \tag{8.6}$$

The arcs which crosses Z_v $(2 \leq v \leq n)$ are ordered by,

$$R(E_{*,v,1,*}) < R(E_{*,v,2,*}) < \ldots < R(E_{*,v,v-1,*}) < R(C_{*,v}) \tag{8.7}$$

$$R(E_{*,v,w,1}) < R(E_{*,v,w,2}) < R(E_{*,v,w,3}), 1 \leq w < v \tag{8.8}$$

$$E_{v-1,v,w,i} \sqsubset E_{v-2,v,w,i} \sqsubset \ldots \sqsubset E_{w,v,w,i}, 1 \leq w < v, 1 \leq i \leq 3 \tag{8.9}$$

$$C_{v-1,v} \sqsubset C_{v-2,v} \sqsubset \ldots \sqsubset C_{1,v} \tag{8.10}$$

$$E_{*,v,*,*} \sqsubset A_v, C_{*,v} \sqsubset A_v \tag{8.11}$$

Equation 8.7 ensures that for the arcs crossing Z_v, the right endpoints are ordered according to the third subscripts. Also the right endpoints for $C_{*,v}$ should be greater than the right endpoints of $E_{*,v,*,*}$. Furthermore, Equation 8.8 orders (the right endpoints of) $E_{*,v,w,*}$ according to the fourth subscripts for any given v and w, and then Equation 8.9 orders them by their first subscripts. Equation 8.10 defines the order between the arcs of $C_{*,*}$, and at most one arc in $C_{*,v}$ can be selected for a CCM. Equation 8.11 expresses the relations between the arcs of $C_{*,v}$, $E_{*,v,*,*}$, and $A_{v,*}$. If $A_{v,*}$ is selected for a CCM, none of the arcs in $E_{*,v,*,*}$ and $C_{*,v}$ can be used.

Thirdly, the arcs which are crossed by Z_u $(1 \leq u \leq n-1)$ are ordered by

$$E_{u,u+1,w,*} \sqsubset E_{u,u+2,w,*} \sqsubset \ldots \sqsubset E_{u,n,w,*}, \qquad 1 \leq w \leq u \tag{8.12}$$

$$C_{u,u+1} \sqsubset C_{u,u+2} \sqsubset \ldots \sqsubset C_{u,n} \tag{8.13}$$

Equation 8.12 implies that for any given u and w, at most one 3-arc crossing cluster can be chosen for a CCM; namely, $E_{u,v,w,*}$ for some v. Similarly, Equation 8.13 means that for a given u, at most one arc in $C_{u,*}$ appears in a CCM.

Lastly, the arcs crossed by Z_z and crossing Z_z $(1 \leq z \leq n)$ are ordered by,

$$E_{*,z,w,1} < E_{z,*,w,*}, E_{*,z,w,2} \between E_{z,*,w,*}, \qquad 1 \leq w < z < n \tag{8.14}$$

$$A_{z,1} < E_{z,*,z,*}, A_{z,2} \between E_{z,*,z,*}, \qquad 1 \leq z < n \tag{8.15}$$

$$A_{z,2} < C_{z,*}, \qquad 1 \leq z < n \tag{8.16}$$

Equation 8.14 guarantees that the arcs from Z_u, for a given w, are anchored by $E_{*,z,w,1}$ and $E_{*,z,w,2}$. For $w = z$, the set $E_{*,z,z,*}$ is not defined. The arcs $E_{z,*,z,*}$ are anchored by arcs $A_{z,1}$ and $A_{z,2}$, according to Equation 8.15. Equation 8.5 and Equation 8.16 together assert that arc $C_{z,*}$ is anchored by arc $A_{z,2}$ and arc set Z_z.

Define $C = C_{*,*}$. It is known that $|C| = 1/2(n^2 - n)$. If $E = E_{*,*,*,*}$, then $|E| = 1/2(n^3 - n)$.

Arcs from Z_u to T

Arcs from Z_u to T are referred to as F_u. F_u consists of u 2-arc crossing clusters, and the clusters are denoted as $F_{u,w}$, $1 \leq w \leq u$. $F_{u,w}$ contains two arcs: $F_{u,w,1}$ and $F_{u,w,2}$, where $F_{u,w,1} \between F_{u,w,2}$. First, F_u is ensured to be from Z_u to T $(1 \leq u \leq n)$; that is,

$$Z_u \between F_{u,*,*}, F_{u,*,*} \between T.$$

Furthermore, the following orders are defined,

$$E_{*,u,w,1} < F_{u,w,*}, E_{*,u,w,2} \between F_{u,w,*}, \qquad 1 \leq w < u \leq n \tag{8.17}$$

$$A_{u,1} < F_{u,u,*}, A_{u,2} \between F_{u,u,*}, \qquad 1 \leq u \leq n \tag{8.18}$$

$$E_{u,*,w,*} \sqsubset F_{u,w,*}, \qquad 1 \leq w \leq u < n \tag{8.19}$$

$$R(F_{*,1,*}) < R(F_{*,2,*}) < \ldots < R(F_{*,n,*}) \tag{8.20}$$

$$R(F_{*,w,1}) < R(F_{*,w,2}), \qquad 1 \leq w \leq n \tag{8.21}$$

$$F_{n,w,i} \sqsubset F_{n-1,w,i} \sqsubset \ldots \sqsubset F_{w,w,i}, \qquad 1 \leq w \leq n, 1 \leq i \leq 2 \tag{8.22}$$

Equation 8.17 and 8.18 ensure that $F_{u,w,*}$ are anchored by $E_{*,u,w,1}$ and $E_{*,u,w,2}$ or by $A_{u,1}$ and $A_{u,2}$, respectively. Equation 8.19 guarantees that if some arcs of $F_{u,w,*}$ appear in a CCM, then none of the arcs of $E_{u,*,w,*}$ can appear in a CCM. The right endpoints of $F_{*,*,*}$ are ordered, according to their second subscripts by Equation 8.20, and then by the third subscript (by Equation 8.21). Furthermore, Equation 8.22 means that only one arc is possible for a CCM in set $F_{*,w,i}$ for any given w and i.

If $F = F_{*,*,*}$, $|F| = n^2 + n$.

Arcs from T to V_v and from V_u to V_v

Two kinds of arcs, $I_{u,v}$ and $P_{u,v}$, are involved. $I_{u,v}$ and $P_{u,v}$ are induced from the edges of $\mathcal{G}(\mathcal{V}, \mathcal{E})$. $I_{u,v}$ can be either a 3-arc crossing cluster or an empty set. If $I_{u,v} \neq \emptyset$, the three arcs in it are $I_{u,v,1}$, $I_{u,v,2}$ and $I_{u,v,3}$ with $I_{u,v,1} \between I_{u,v,2}$, $I_{u,v,1} \between I_{u,v,3}$, and $I_{u,v,2} \between I_{u,v,3}$. $P_{u,v}$ contains $(n - v)$ crossing clusters. Each cluster, $P_{u,v,w}$ $(v < w \leq n)$ is empty or has two crossing arcs. If $P_{u,v,w} \neq \emptyset$, the two arcs are $P_{u,v,w,1}$ and $P_{u,v,w,2}$, $P_{u,v,w,1} \between P_{u,v,w,2}$.

The arcs from T to V_v are partitioned into two sets: $P_{0,v,w}$ and $I_{0,v}$. $P_{0,v,w}$ is a 2-arc crossing cluster and $I_{0,v}$ is a 3-arc crossing cluster. Are all nonempty sets.

The edge information of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is used to construct the arcs from V_u to V_v. For $I_{u,v}$, $1 \leq u < v \leq n$, if $(u, v) \notin \mathcal{E}_\mathcal{G}$ $I_{u,v} = \emptyset$; otherwise, $(u, v) \in \mathcal{E}_\mathcal{G}$, $I_{u,v}$ is a 3-arc crossing cluster.

For $P_{u,v,w}$, $1 \leq u < v < w \leq n$, if $(u, w) \notin \mathcal{E}_\mathcal{G}$, $P_{u,v,w} = \emptyset$; otherwise, $(u, w) \in \mathcal{E}_\mathcal{G}$, $P_{u,v,w}$ is a 2-arc crossing cluster.

The construction is to ensure that $I_{0,*,*}$ and $P_{0,*,*,*}$ are crossed by T; $I_{u,*,*}$ and $P_{u,*,*,*}$ are crossed by V_u $(1 \leq u \leq n-1)$, and $I_{*,v}$ and $P_{*,v,*,*}$ are crossing V_v; that is,

$$T \between I_{0,*,*}, \qquad T \between P_{0,*,*,*} \tag{8.23}$$

$$V_u \between I_{u,*,*}, 1 \leq u < n \qquad V_u \between P_{u,*,*,*}, 1 \leq u < n-1 \tag{8.24}$$

$$I_{*,v,*} \between V_v, 1 \leq v \leq n \qquad P_{*,v,*,*} \between V_v, 1 \leq v < n \tag{8.25}$$

For the arcs which are crossing V_v, the following orders are specified,

$$R(I_{*,v,*}) < R(P_{*,v,v+1,*}) \qquad 1 \leq v \leq n-1 \tag{8.26}$$

$$R(P_{*,v,v+1,*}) < R(P_{*,v,v+2,*}) < \ldots < R(P_{*,v,n,*}), \qquad 1 \leq v \leq n-1 \tag{8.27}$$

$$R(P_{*,v,w,1}) < R(P_{*,v,w,2}), \qquad 1 \leq v < w \leq n \tag{8.28}$$

$$P_{v-1,v,w,i} \sqsubset P_{v-2,v,w,i} \sqsubset \ldots \sqsubset P_{0,v,w,i}, 1 \leq v < w \leq n, \qquad 1 \leq i \leq 2 \tag{8.29}$$

$$I_{v-1,v,*} \sqsubset I_{v-2,v,*} \sqsubset \ldots \sqsubset I_{0,v,*}, \qquad 1 \leq v \leq n \tag{8.30}$$

For a given v, Equation 8.27 enforces the right endpoints of $P_{*,v,*,*}$ to be sorted according to the third subscript. Then Equation 8.28 guarantees that for any given

v and w, the right endpoints for $P_{*,v,w,*}$ are sorted according to the fourth subscript. Furthermore, for any given v, w, and i, Equation 8.29 asserts that, at most, one arc in $P_{*,v,w,i}$ can be selected for a CCM.

Next, the orders for the arcs which are crossed by T, and those arcs which are crossed by V_u are introduced as follows.

$$P_{u,u+1,w,*} \sqsubset P_{u,u+2,w,*} \sqsubset \ldots \sqsubset P_{u,n,w,*}, \qquad 0 \leq u, u+1 < w \leq n \tag{8.31}$$

$$I_{u,w,*} \sqsubset P_{u,*,w,*}, \qquad 0 \leq u, u+1 < w \leq n \tag{8.32}$$

Equation 8.31 and 8.32 mean that either (1) one 2-arc crossing cluster $P_{u,v,w,*}$ can be selected for a CCM, or (2) the 3-arc crossing cluster $I_{u,w,*}$ is selected for a CCM, or (3) none of them is selected.

Furthermore, for the arcs which are crossed by T, the following orders are specified,

$$F_{*,w,1} < I_{0,w}, F_{*,w,2} \between I_{0,w}, \qquad 1 \leq w \leq \ell \tag{8.33}$$

$$F_{*,w,1} < P_{0,*,w,*}, F_{*,w,2} \between P_{0,*,w,*}, \qquad 2 \leq w \leq \ell \tag{8.34}$$

Equation 8.33 implies that $I_{0,w}$ is anchored by $F_{*,w,1}$ and $F_{*,w,2}$, and Equation 8.34 implies that $P_{0,*,w,*}$ is anchored by $F_{*,w,1}$ and $F_{*,w,2}$.

Lastly, those arcs crossed by V_z and the arcs which cross V_z are ordered by,

$$P_{*,z,w,1} < I_{z,w}, P_{*,z,w,2} \between I_{z,w}, \qquad 1 \leq z, z+1 \leq w \leq n \tag{8.35}$$

$$P_{*,z,w,1} < P_{z,*,w,*}, P_{*,z,w,2} \between P_{z,*,w,*}, \qquad 1 \leq z, z+1 < w \leq n \tag{8.36}$$

Equation 8.35 ensures that $I_{z,w}$ is anchored by $P_{*,z,w,1}$ and $P_{*,z,w,2}$, and Equation 8.36 guarantees that $P_{z,*,w,*}$ is anchored by $P_{*,z,w,1}$ and $P_{*,z,w,2}$.

If $P = P_{*,*,*,*}$ and $I = I_{*,*}$, $|P| \leq 1/3(n^3 - n) \cdot |I| \leq 3/2(n^2 + n)$.

Denote $\mathcal{D}_\mathcal{G} = H \cup A \cup C \cup Z \cup E \cup F \cup I \cup P \cup V$. $\mathcal{S}_\mathcal{G}$ is the set of those endpoints of the arcs in $\mathcal{D}_\mathcal{G}$. The target contact map $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{D}_\mathcal{G})$ is fully specified. The following results can be shown for $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{D}_\mathcal{G})$:

Lemma 34. *(i) An arc $a \in E$ crosses no more than $9n^3$ arcs.*
(ii) An arc $a \in F$ crosses no more than $17n^4$ arcs.
(iii) An arc $a \in I$ crosses no more than $9n^3$ arcs.
(iv) $|\mathcal{D}_\mathcal{G} - H| < |H|$.

Proof. Since $|A| + |C| + |E| + |F| \leq 2n + 1/2(n^2 - n) + 1/2(n^3 - n) + (n^2 + n) \leq 4n^3$, the only possible arcs an arc $a \in E$ can cross are from A, C, E, F, and Z_u for some u with $1 \leq u \leq n$.

With the exception of A, C, E, F, an arc $a \in F$ can cross some arcs in P, I, and T. Hence, $|P| + |I| < 4n^3$ and $|T| = 9n^4$. The arc $a \in I$ crosses those arcs from P, I, and one V_u for some u with $1 \leq u \leq n$. It is easy to verify that $|\mathcal{D}_\mathcal{G} - H| < |H|$. $\square$

8.6.3 Pattern Construction

Large Crossing Clusters

Like for the target, first, $2\ell+2$ crossing clusters are constructed, which are H', Z'_u $(1 \le u \le \ell)$, T', and V'_u $(1 \le u \le \ell)$. H' is a $28n^4$-arc crossing cluster. Here, Z'_u is a $5n^3$-arc crossing cluster, T' is a $9n^4$-arc crossing cluster, and V'_u $(1 \le u \le \ell)$ is a $5n^3$-arc crossing cluster. Denote $Z' = \bigcup_{u=1}^{\ell} Z'_u$ and $V' = \bigcup_{i=1}^{\ell} V'_u$. Furthermore, these large clusters are ordered by,

$$H' < Z'_1 < \ldots < Z'_\ell < T' < V'_1 < \ldots < V'_\ell.$$

Arcs from H' to Z'_1

A 2-arc crossing cluster from H' to Z'_1 exists, and is denoted A'. The two arcs are denoted as A'_1 and A'_2, $A'_1 \between A'_2$. Furthermore, A' is from H' to Z'_1.

$$H' \between A', A' \between Z'_1 \tag{8.37}$$

Arcs from Z'_u to Z'_{u+1}

There are two types of arcs from Z'_u to Z'_{u+1}: E'_u and C'_u. C'_u is a single arc. E'_u contains u 3-arc crossing clusters, $E'_{u,w}$, $1 \le w \le u$. For each cluster, $E'_{u,w}$, its three arcs are $E'_{u,w,1}$, $E'_{u,w,2}$ and $E'_{u,w,3}$ with $E'_{u,w,1} \between E'_{u,w,2}$, $E'_{u,w,1} \between E'_{u,w,3}$, and $E'_{u,w,2} \between E'_{u,w,3}$

$E'_{u,*,*}$ and C'_u are ensured to be from Z'_u and to Z'_{u+1}; that is,

$$Z'_u \between E'_{u,*,*}, \qquad E'_{u,*,*} \between Z'_{u+1}, 1 \le u \le \ell-1 \tag{8.38}$$
$$Z'_u \between C'_u, \qquad C'_u \between Z'_{u+1}, 1 \le u \le \ell-1 \tag{8.39}$$

Furthermore, the following orders are defined.

$$A'_1 < E'_{1,*,*}, A'_2 \between E'_{1,*,*} \tag{8.40}$$
$$E'_{u,w_1,*} \between E'_{u,w_2,*}, \qquad 1 \le w_1 < w_2 \le u \le \ell-1 \tag{8.41}$$
$$E'_{u,w,1} < E'_{u+1,w,*}, E'_{u,w,2} \between E'_{u+1,w,*}, \qquad 1 \le w \le u < \ell-1 \tag{8.42}$$
$$E'_{u,*,*} \between C'_u, \qquad 1 \le u \le \ell-1 \tag{8.43}$$
$$C'_{u-1} < E'_{u,u,*} \qquad 2 \le u \le \ell-1 \tag{8.44}$$

$E'_{1,*,*}$ (a 3-arc crossing cluster) is anchored by A'_1 and A'_2 (Equation 8.40). Equation 8.41 ensures that arcs in $E_{u,*,*}$ form a crossing cluster. Furthermore, Equation 8.42 ensures that the 3-arc crossing cluster $E'_{u+1,w,*}$, is anchored by $E'_{u,w,1}$ and $E'_{u,w,2}$. Equation 8.43 ensures that the crossing cluster, $E'_{u,*,*}$, crosses arc C'_u. Together with the requirement in Equation 8.38 and Equation 8.44, the arc set, $E'_{u,u,*}$, is anchored by C'_{u-1} and Z'_u.

$C' = C'_*$ and $E' = E'_{*,*,*}$ are introduced for notation simplicity.

Arcs from Z'_ℓ to T'

The arcs from Z'_ℓ to T' are called F'. F' has ℓ crossing clusters, where each contains two arcs. The crossing clusters are signified as F'_w $(1 \le w \le \ell)$ and the two arcs in F'_w are represented as $F'_{w,1}$ and $F'_{w,2}$, $F'_{w,1} \between F'_{w,2}$. Furthermore, the following orders are placed.

$$V'_\ell \between F'_{*,*}, F'_{*,*} \between T' \tag{8.45}$$
$$E'_{\ell-1,w,1} < F'_{w,*}, E'_{\ell-1,w,2} \between F'_{w,*}, \qquad 1 \le w \le \ell - 1 \tag{8.46}$$
$$C'_{\ell-1} < F'_\ell \tag{8.47}$$
$$F'_{w_1,*} \between F'_{w_2,*}, \qquad 1 \le w_1 < w_2 \le \ell \tag{8.48}$$

Equation 8.45 ensures that $F'_{*,*}$ is from V'_ℓ to T'. $F'_{w,*}$ is anchored by $E'_{\ell-1,w,1}$ and $E'_{\ell-1,w,2}$ by Equation 8.46; F'_ℓ is anchored by $C'_{\ell-1}$ and V'_ℓ by Equation 8.47. Furthermore, arcs in $F'_{*,*}$ form a crossing cluster according to Equation 8.48.

Arcs from T' to V'_1 and from V'_u to V'_{u+1}

There are two kinds of arcs: I'_u, $(0 \le u < \ell)$, and P'_u. I'_u $(0 \le u < \ell - 1)$ is a 3-arc crossing cluster: $I'_{u,1}$, $I'_{u,2}$, and $I'_{u,3}$, where $I'_{u,1} \between I'_{u,2}$, $I'_{u,1} \between I'_{u,3}$ and $I'_{u,2} \between I'_{u,3}$. P'_u contains $(\ell - u - 1)$ $(0 \le u \le n-2)$ 2-arc crossing clusters. Each cluster is identified as $P'_{u,w}$ $(u + 1 < w \le \ell)$. The two arcs of $P'_{u,w}$ are $P'_{u,w,1}$ and $P'_{u,w,2}$, $P'_{u,w,1} \between P'_{u,w,2}$.

First, to enforce that I'_0 and $P'_{0,*,*}$ are crossed by T', I_u and $P_{u,*,*}$ are crossed by V'_u $(1 \le u \le \ell - 1)$, and I'_u and $P'_{u,*,*}$ crosses V'_{u+1}, the following orders are defined.

$$T' \between I'_0, T' \between P'_{0,*,*} \tag{8.49}$$
$$V'_u \between I'_u, \qquad 1 \le u < \ell \tag{8.50}$$
$$V'_u \between P'_{u,*,*}, \qquad 1 \le u < \ell - 1 \tag{8.51}$$
$$I'_{u,*} \between V'_{u+1}, \qquad 0 \le u < \ell \tag{8.52}$$
$$P'_{u,*,*} \between V'_{u+1}, \qquad 0 \le u < \ell - 1 \tag{8.53}$$

Furthermore, the arcs in I'_u and $P'_{u,*}$ form a crossing cluster; that is,

$$I'_u \between P'_{u,*}, \qquad 0 \le u < \ell - 1 \tag{8.54}$$
$$P'_{u,w_1} \between P'_{u,w_2}, \qquad 0 \le u, u + 1 < w_1 < w_2 \le \ell \tag{8.55}$$

In addition, the following orders are introduced.

$$F'_{1,1} < I'_0, F'_{1,2} \between I'_0 \tag{8.56}$$
$$F'_{w,1} < P'_{0,w}, F'_{w,2} \between P'_{0,w}, \qquad 2 \le w \le \ell \tag{8.57}$$
$$P'_{u,u+2,1} < I'_{u+1}, P'_{u,u+2,2} \between I'_{u+1}, \qquad 1 \le u < \ell - 1 \tag{8.58}$$
$$P'_{u,w,1} < P'_{u+1,w,*}, P'_{u,w,2} \between P'_{u+1,w,*}, \qquad 1 \le u, u + 2 < w \le \ell \tag{8.59}$$

Equation 8.56 ensures that I'_0 is anchored by $F'_{1,1}$ and $F'_{1,2}$; Equation 8.57 ensures that $P'_{0,w}$ is anchored by $F'_{w,1}$ and $F'_{w,2}$. I'_{u+1} is anchored by $P'_{u,u+2,1}$ and $P'_{u,u+2,2}$; $P'_{u+1,w,*}$ is anchored by $P'_{u,w,1}$ and $P'_{u,w,2}$.

P' and I' are defined as $P' = P'_{*,*,*}$ and $I' = I'_*$, for notation simplicity.

$\mathcal{D}'_{n,\ell} = H' \cup A' \cup C' \cup Z' \cup E' \cup F' \cup I' \cup P' \cup V'$ and $\mathcal{S}'_{n,\ell}$ are the endpoints of those arcs in $\mathcal{D}'_{n,\ell}$. It is not difficult to verify the following result by the constructions.

Lemma 35. *$\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ is a $\{<, \between\}$-structured contact map, and $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{D}_{\mathcal{G}})$ is a $\{<, \sqsubset, \between\}$-structured contact map.*

8.6.4 Correctness

According to the construction, the following results hold.

Lemma 36. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{D}_{\mathcal{G}})$, then $\forall \mathcal{M}$, $\mathcal{M}(H') = H$ and $\mathcal{M}(A') = A_{u_1,*}$ for some u_1, with $1 \le u_1 \le n$.*

Proof. H' is a $28n^4$-arc crossing cluster, and the number of arcs $\mathcal{A}_G - H$ is less than $28n^4$. Thus for a mapping $\mathcal{M}$, there exists $h_1 \in H'$ such that $\mathcal{M}(h_1) \in H$. The arc set $\mathcal{M}(h_1)$ crosses and is crossed by $H \cup A$. Thus, $\mathcal{M}(H') \subset (H \cup A)$.

In contrast, there are $28n^4$ arcs crossing A'. Similarly, it is argued that there exists $h_2 \in H'$ such that $\mathcal{M}(h_2) \in H$, and h_2 crosses A'. $\mathcal{M}(A') \subset (H \cup A)$ since $\mathcal{M}(h_2)$ crosses $\mathcal{M}(A')$. Now, it is known that $\mathcal{M}(H' \cup A') \subset (H \cup A)$.

Since $A_u \sqsubset A_v$, for $1 \le u < v \le n$, A_u and A_v cannot occur simultaneously in a CCM. Thus, to form a $28n^4 + 2$-arc crossing cluster, all the arcs in H and one pair of arcs in A, say $A_{u_1,*}$ for some u_1 ($1 \le u_1 \le n$), are to be chosen. Therefore, the statement holds. □

Lemma 37. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{D}_{\mathcal{G}})$, then $\forall \mathcal{M}$, $\mathcal{M}(E'_{1,1,*}) = E_{u_1,u_2,u_1,*}$, and $\mathcal{M}(C'_1) = C_{u_1,u_2}$ for some u_1, u_2 with $1 \le u_1 < u_2 \le n$.*

Proof. From Lemma 36, if $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{A}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{A}_{\mathcal{G}})$, then $\mathcal{M}(A') = A_{u_1}$ for some u_1. It is specified that $A'_1 < E'_{1,1,*}$ and $A'_2 \between E'_{1,1,*}$. $\mathcal{M}(E'_{1,1,*}) \subset E_{u_1,*,u_1,*} \cup F_{u_1,u_1,*}$ since $E_{u_1,*,u_1,*} \cup F_{u_1,u_1,*}$ contains all the arcs which are greater than $A_{u_1,1}$ and are crossed by $A_{u_1,2}$. Since $E_{u_1,v_1,u_1,*} \sqsubset E_{u_1,v_2,u_1,*}$, for $v_1 < v_2$, and $E_{u_1,*,u_1,*} \sqsubset F_{u_1,u_1,*}$, it follows that $\mathcal{M}(E'_{1,1,*}) = E_{u_1,u_2,u_1,*}$ for some u_2, or $\mathcal{M}(E'_{1,1,*}) = F_{u_1,u_1,*}$. However, $F_{u_1,u_1,*}$ contains only two arcs, thus, $\mathcal{M}(E'_{1,1,*}) = E_{u_1,u_2,u_1,*}$ for some u_2.

$E'_{1,1,*}$ is crossed by a Z'_1 which is a $5n^3$-arc crossing cluster. Therefore, there exists $z_1 \in \mathcal{M}(Z'_1)$ with $\mathcal{M}(z_1) \in Z_{u_1}$, since E_{u_1,u_2,u_1} is crossed by fewer than $2 \times 5n^3$ arcs. Also, $E'_{1,1,*}$ crosses Z'_2, and Z'_2 is a $5n^3$ arc-crossing cluster. By a similar argument, there exists $z_2 \in \mathcal{M}(Z'_2)$ with $\mathcal{M}(z_2) \in Z_{u_2}$. In $\mathcal{CM}(\mathcal{V}_G, \mathcal{A}_G)$, the only arc that satisfies the following three conditions is C_{u_1,u_2}: (1) is greater than $A_{u_1,2}$; (2) is crossed by $\mathcal{M}(z_1)$, and (3) is crossing $\mathcal{M}(z_2)$. Thus, $\mathcal{M}(C'_1) = C_{u_1,u_2}$. □

Lemma 38. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{D}_{\mathcal{G}})$ and $\mathcal{M}(E'_{1,1,*}) = E_{u_1,u_2,u_1,*}$, then $\mathcal{M}(E'_{2,v,*}) = E_{u_2,u_3,u_v,*}$ and $\mathcal{M}(C'_v) = C_{u_v,u_{v+1}}$ $(v = 1, 2)$ for some u_3 with $u_2 < u_3 \leq n$.*

Proof. For $E'_{2,1,*}$. $E'_{1,1,1} < E'_{2,1,*}$, and $E'_{1,1,2} \between E'_{2,1,*}$. Also, $E_{u_2,v_1,u_1,*} \sqsubset E_{u_2,v_2,u_1,*}$ for $u_2 < v_1 < v_2 \leq n$, and $F_{u_2,1,*}$ contains only two arcs. Consequently, $\mathcal{M}(E'_{2,1,*}) = E_{u_2,u_3,u_1,*}$ for some u_3.

Now it is necessary to prove that $\mathcal{M}(E'_{2,2,*}) = E_{u_2,u_3,u_2,*}$. $E'_{1,1,*}$ crosses Z'_2, and Z'_2 is a $5n^3$-arc crossing cluster. Thus, $\exists z_2 \in Z'_2$ with $\mathcal{M}(z_2) \in Z_2$. By a similar argument for the arc set, $E'_{2,1,*}$, $\exists z_3 \in Z'_3$ with $\mathcal{M}(z_3) \in Z_3$. By Lemma 37, $\mathcal{M}(C'_1) = C_{u_1,u_2}$. In $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{A}_{\mathcal{G}})$, there are four crossing arcs which are greater than C_{u_1,u_2}, are crossed by $\mathcal{M}(z_2)$, and are crossing $\mathcal{M}(z_3)$. These arcs are $\{C_{u_2,u_3}\} \cup E_{u_2,u_2,u_3,*}$. In $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{A}_{n,\ell})$, there are four crossing arcs which are greater than C'_1, are crossed by z_2, and are crossing z_3. These arcs are $\{C'_2\} \cup E'_{2,2,*}$. Therefore, $\mathcal{M}(C'_2) = C_{u_2,u_3}$ and $\mathcal{M}(E'_{2,2,*}) = E_{u_2,u_3,u_2,*}$. □

Lemma 39. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{D}_{\mathcal{G}})$, $\mathcal{M}(E'_{k,v_1,*}) = E_{u_k,u_{k+1},u_{v_1},*}$, and $\mathcal{M}(C'_{v_1}) = C_{u_{v_1},u_{v_1+1}}$, $(v_1 = 1, \ldots, k)$ for $u_1 < \ldots < u_{k+1} \leq n$, then $\mathcal{M}(E'_{k+1,v_2,*}) = E_{u_{k+1},u_{k+2},u_{v_2},*}$, and $\mathcal{M}(C'_{v_2}) = C_{u_{v_2},u_{v_2+1}}$ $(v_2 = 1, \ldots, k+1)$ for some u_{k+2} with $u_{k+1} < u_{k+2} \leq n$.*

Lemma 39 can be illustrated by using arguments similar to those in Lemma 37 and 38.

Lemma 40. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{D}_{\mathcal{G}})$, then $\forall \mathcal{M}$, $\mathcal{M}(E'_{\ell-1,v,*}) = E_{u_{\ell-1},u_\ell,u_v,*}$ and $\mathcal{M}(C'_{\ell-1}) = C_{u_{\ell-1},u_\ell}$, with $v = 1, \ldots, \ell-1$ and for some $u_1, \ldots, u_\ell$, $1 \leq u_1 < \ldots < u_\ell \leq n$.*

Lemma 40 can be shown by induction, using Lemma 36-39.

Lemma 41. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{D}_{\mathcal{G}})$, and if $\mathcal{M}(E'_{\ell-1,v,*}) = E_{u_{\ell-1},u_\ell,u_v,*}$, and $\mathcal{M}(C'_{\ell-1}) = C_{u_{\ell-1},u_\ell}$ for $u_1, ..., u_\ell$, with $1 \leq u_1 < \ldots < u_\ell \leq n$, then $\mathcal{M}(F'_{v,*}) = F_{u_\ell,u_v,*}$ $(1 \leq v \leq \ell)$.*

Proof. Since the arc set $F'_{1,*}$ is greater than $E'_{\ell-1,1,1}$ and is crossed by $E'_{\ell-1,1,2}$, $\mathcal{M}(F'_1) \subset (F_{u_\ell,u_1,*} \cup E_{u_\ell,*u_1,*})$ (if $E_{u_\ell,*u_1,*}$ is not defined, treat it as an empty set). As F'_1 crosses no fewer than $9n^4$ arcs in $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{A}_{n,\ell})$ and $E_{u_\ell-1,u_1,w}$ crosses fewer than $9n^4$ arcs, then the only possible choice is $\mathcal{M}(F'_1) = F'_{u_\ell,u_1}$. The argument is similar to that for $F'_{v,i}$ with $2 \leq v < \ell$. F'_ℓ is greater than $C'_{\ell-1}$, and crosses a $9n^4$-arc crossing cluster, so that the only choice is $\mathcal{M}(F'_\ell) = F_{u_\ell,u_\ell}$. □

Lemma 42. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_{\mathcal{G}}, \mathcal{D}_{\mathcal{G}})$, then $\forall \mathcal{M}$, $\mathcal{M}(I'_{0,*}) = I_{0,u_1,*}$, and $\mathcal{M}(P'_{0,v,*}) = P_{0,u_1,u_v,*}$, $(2 \leq v \leq \ell)$ for some $u_1, ..., u_\ell$ with $1 \leq u_1 < \ldots < u_\ell \leq n$.*

Proof. By Lemma 41, given a mapping $\mathcal{M}$, $\mathcal{M}(F'_v) = F_{u_\ell,u_v}$, for some $u_1, ..., u_\ell$ with $1 \leq u_1 < \ldots < u_\ell \leq n$. The arc set $I'_{0,*}$ is greater than $F'_{1,1}$ and it is crossed by $F'_{1,2}$. Therefore, $\mathcal{M}(I'_{0,*}) \in (I_{0,u_1,*} \cup P_{0,*,u_1,*})$, where $I'_{0,*}$ is a 3-arc crossing cluster. The only 3-arc crossing cluster in $I_{0,u_1,*} \cup P_{0,*,u_1,*}$ is $I_{0,u_1,*}$. Thereafter, $\mathcal{M}(I'_{0,*}) = I_{0,u_1,*}$.

For the arc set $P'_{0,2,*}$, $F'_{2,1} < P'_{0,2,*}$ and $F'_{2,2} \between P'_{0,2,*}$. To satisfy these relations after applying the mapping $\mathcal{M}$, $\mathcal{M}(P'_{0,2,*}) \in (I_{0,u_2,*} \cup P_{0,*,u_2,*})$. In contrast, I'_0 crosses V'_1, which is a crossing arc cluster with $5n^3$ arcs. I_{0,u_1} crosses fewer than $2 \times 5n^3$ arcs, and there exists $z_1 \in V'_1$ with $\mathcal{M}(v_1) \in V_{u_1}$. Then, $\mathcal{M}(P'_{0,2})$ must cross $\mathcal{M}(v_1)$, the only possible pair of arcs in $(I_{0,u_2} \cup P_{0,u_2})$ which crosses $\mathcal{M}(v_1)$ is P_{0,u_1,u_2}. The same can be shown for $3 \leq v \leq \ell$. □

Similarly, the following can be shown.

Lemma 43. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{D}_\mathcal{G})$, then $\forall \mathcal{M}$, $\mathcal{M}(I'_{w,*}) = I_{u_w,u_{w+1},*}$, and $\mathcal{M}(P'_{w,v,*}) = P_{u_w,u_{w+1},u_v,*}$($1 \leq w < \ell$, $1 \leq w+1 < v \leq \ell$) for some $u_1, ..., u_\ell$ with $1 \leq u_1 < \ldots < u_\ell \leq n$.*

By the construction of $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{D}_\mathcal{G})$, the following holds.

Lemma 44. *If $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{D}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{D}_\mathcal{G})$, $\mathcal{G}$ has a size ℓ clique.*

Proof. By Lemma 43, if $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{A}_{n,\ell})$ occurs in $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{A}_\mathcal{G})$, $\mathcal{M}(I'_{w,*}) = I_{u_w,u_{w+1},*}$ and $\mathcal{M}(P'_{w,v,*}) = P_{u_w,u_{w+1},u_v,*}$ ($1 \leq w \leq \ell$, $1 \leq w+1 < v \leq \ell$) for some $u_1, ..., u_\ell$ with $1 \leq u_1 < \ldots < u_\ell \leq n$.

By the construction of $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{A}_\mathcal{G})$, $P_{u_w,u_{w+1},u_v} \in \mathcal{A}_\mathcal{G}$, P_{u_w,u_{w+1},u_v} is not empty iff $(u_w, u_v) \in \mathcal{E}_\mathcal{G}$, and $I_{u_w,u_{w+1}}$ is not empty iff $(u_w, u_{w+1}) \in \mathcal{E}_\mathcal{G}$. Therefore, $u_1, \ldots, u_\ell$ forms a size ℓ clique and the statement holds. □

Finally, the following theorem can be shown:

Theorem 45. *$\mathcal{CM}(\mathcal{V}_{\ell,n}, \mathcal{D}_{\ell,n})$ occurs in $\mathcal{CM}(\mathcal{V}_G, \mathcal{D}_G)$ if and only if G contains a clique with size ℓ, and hence, the CCMPM problem is NP-hard.*

Proof. The only if direction has already been shown. For the if direction, suppose there is a clique $u_1, \ldots u_\ell$, a mapping $\mathcal{M}$ can be constructed straightforwardly between $\mathcal{CM}(\mathcal{S}_{n,\ell}, \mathcal{A}_{n,\ell})$ and a subset of arcs in $\mathcal{CM}(\mathcal{S}_\mathcal{G}, \mathcal{A}_\mathcal{G})$. The reduction is polynomial, and thus, the statement holds. □

Actually, a stronger result is implied. That is, the problem is NP-hard, even for the case where the target is a $\{<, \sqsubset, \between\}$-structured contact map (in general, the arcs in target can share endpoints).

This also settles another problem, namely, the CMO problem with $\{<, \between\}$-structured patterns, where a maximized common CCM between two given contact maps is to be found. The complexity of this problem has been open [66].

Theorem 46. *The CMO problem is NP-hard.*

Proof. Given a CCMPM problem instance: $\mathcal{CM}(\mathcal{S}_p, \mathcal{D}_p)$ and $\mathcal{CM}(\mathcal{S}, \mathcal{D})$, find the maximized common CCM $\mathcal{CM}(\mathcal{S}'_p, \mathcal{D}'_p)$ between $\mathcal{CM}(\mathcal{S}_p, \mathcal{D}_p)$ and $\mathcal{CM}(\mathcal{S}, \mathcal{D})$, and then verify whether $\mathcal{CM}(\mathcal{S}'_p, \mathcal{D}'_p)$ is identical to $\mathcal{CM}(\mathcal{S}_p, \mathcal{D}_p)$.

Clearly this reduction is polynomial. Thus, the theorem holds. □

8.7 Counterexample

In this section, a counterexample is given for the algorithm in [66,67]. The example is displayed in Figure 8.14. The arcs are labeled with letters instead of numbers for ease of illustration. The pattern is a CCM with 24 arcs. The target contains 42 arcs, and is $\{<, \sqsubset, \between\}$-structured. The arcs are labeled in a way, for mapping an arc of a pattern to an arc of the target which is labeled with the same letter in a different case.

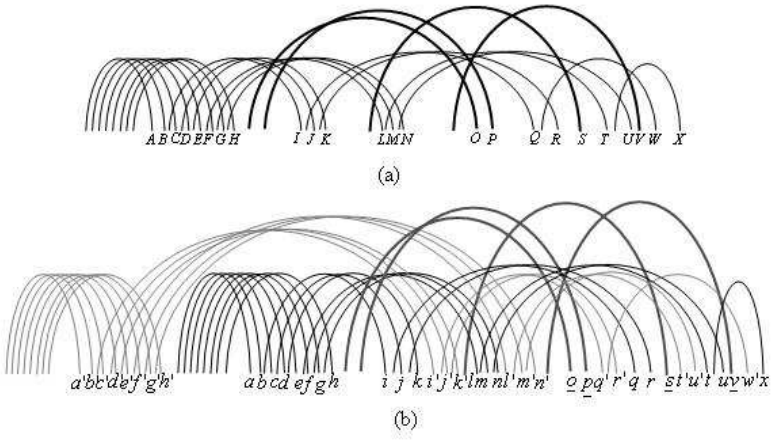


Figure 8.14: Example demonstrating the flaw of the algorithm: (a) A $\{\between, <\}$-structured CM as the pattern, and (b) the target CM.

It can be verified that the pattern does not occur in the target, but the algorithm in [66, 67] produces a "yes" answer.

8.8 Conclusion and discussion

This section offers some remarks on the results on the LCP problem under bottleneck distance and CMO problem under the distance constraint.

The PTASs for both problems have high runtime complexities, rendering them impractical. However, it is possible to reduce the runtime complexity further. First, a large part of the runtime complexity of the current method is due to an $O(m^2n^2)$ term, from the exhaustive search for an appropriate rotation axis. Since two similar structures can share quite a number of well-aligned local structural fragments, it is conceivable that

these fragments be used to anchor the rotation axis, and this can reduce the time needed to find the rotation axis. The random sampling and other techniques, as mentioned in Chapter 7, can also be adopted to reduce this time complexity to $O(n^2 \log^3 m)$, but with some compromise in the solution's accuracy.

Chapter 9

Conclusion and Future Work

In this thesis, protein structure prediction is explored in various steps: structural fragments, sampling structures, side chain packing, decoy selection, structural comparison, and structural alignment. The problems can be divided into two categories: computational based and informatics based.

Some problems are solved from the computational aspect. A PTAS was presented for clustering structural fragments. It is proven that the side chain packing problem remains NP-complete, even if each residue contains, at most, three rotamer candidates. Effective speed-up techniques for clustering very large decoy sets are proposed and tested successfully on cases of 10^5 decoys. Also, a distance approximation algorithm is proposed to find nearly optimal GDT scores. Based on the method, a new approach to the structural alignment problem is developed. Lastly, three open questions from the area of structural alignment are answered, confirming that the three problems from the area are NP-complete.

Five programs for problems from the informatics aspect are developed. A program, called FRazor, is created to predict the structural candidates for the sequence fragments. A program, called FALCON, is developed to sample the structures candidate for an entire sequence. The HEXAGON program utilizes neighborhood information to predict the side chain conformations. In addition, a hypothesis is proposed to explain why clustering approaches work for selecting decoys from *ab initio* structure prediction methods, and this results in a program named ONION. FALCON and ONION together allows the structures to be refined.

Typically, there are two main components in the protein structure prediction problem: the energy function and search method. From the studies on the ONION program, the FALCON-ONION approach is promising as a search strategy. Consequently, the future focus should be on the design of energy function. However, it is unlikely that a universal energy function exists for all structures. Using existing structures as a skeleton is a solution; that is, threading can be considered an initial step of this approach. In most of the present threading methods, the backbone is assumed to be rigid and adopts existing structures, which hinders the search for exploring new structures. To use the existing structures as a soft (rather than rigid) skeleton might be a better solution.

The most successful methods for protein structures prediction assume a hidden Markov model or an identically independent distribution model explicitly or implicitly. A direct

extension of this is to employ long range contact information. However, to factor such considerations into the problem involves very high time complexities. Furthermore, one concern is that the current training data may not be sufficient for such an extension. Generally speaking, there are two technicalities involved: one on the kinds of information to add, and the other on how to combine the new information. We are of the belief that long range information is very important, and the present problem is to have a systematic way to adopt them. Perhaps a better question to ask is: to what extent can meaningful information be derived from the current database of structures? Our conjecture is that the skeleton information and long range contact information can reveal most of the results that can be achieved for the task of protein structure predictions, based on the database at this moment.

There are quite a number of computational problems in the field of protein structures. Some of these problems are well formulated, and are well-known open questions. This thesis offers answers to several such problems in various degrees of satisfaction. In some cases, the solutions are only partial, or are heuristic, for example, our solution for structural comparison. As a result, it would certainly be beneficial if these solutions can be re-examined, and extended upon.

Another problem where further work is needed is with regard to the speed performance of the FALCON-ONION program. This is due to the current implementation by the Monte-Carlo search, as well as the redundancy in the movements. A torsional angle dynamic approach might work better to improve the method, since it can couple the prediction methods and structure determination methods more tightly. However, to derive the formulas would be non-trivial.

Although a number of problems are solved in this thesis, it also raises more questions. Further research is being carried out, even as this thesis is being completed, and more results can be expected from these efforts in the near future.

Bibliography

[1] 8th community wide experiment on the critical assessment of techniques for protein structure prediction. http://www.predictioncenter.org/casp8/.

[2] I-TASSER protein structure decoys. http://zhang.bioinformatics.ku.edu/I-TASSER/decoys/.

[3] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* . New York: Dover , 1972.

[4] A. De Brevern and C. Benros and R. Gautier and H. Valadi and S. Hazout and C. Etchebest. Local backbone structure prediction of proteins. *In Silico Biol.*, 4(3):381–386, 2004.

[5] P. K. Agarwal, J. Matoušek, and S. Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom. Theory Appl.*, 1(4):189–201, 1992.

[6] T. Akutsu. Protein structure alignment using dynamic programming and iterative improvement. *IEICE Trans. Information and Systems*, E79-D(12):1629–1636, 1996.

[7] T. Akutsu. Np-hardness results for protein side-chain packing. *Genome Information Series*, 8:180–186, 1997.

[8] T. Akutsu and H. Tashimo. Protein structure comparison using representation by line segment sequences. In *Pac Symp Biocomput*, pages 25–40, 1996.

[9] N. N. Alexandrov. SARFing the PDB. *Protein Eng.*, 9(9):727–732, 1996.

[10] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. In *SCG '87: Proceedings of the third annual symposium on Computational geometry*, pages 308–315, New York, NY, (USA), 1987. ACM Press.

[11] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acids Res.*, 25(17):3389–3402, 1997.

[12] C. Ambühl, S. Chakraborty, and B. Gärtner. Computing largest common point sets under approximate congruence. In *ESA '00: Proceedings of the 8th Annual European Symposium on Algorithms*, pages 52–63, London, UK, 2000. Springer-Verlag.

[13] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 9(5):698–700, 1987.

[14] K. C. D. Bahadur, T. Akutsu, E. Tomita, and T. Seki. Protein side-chain packing problem: a maximum edge-weight clique algorithmic approach. In *APBC '04: Proceedings of the second conference on Asia-Pacific bioinformatics*, pages 191–200, Darlinghurst, Australia, 2004. Australian Computer Society, Inc.

[15] D. Baker. A surprising simplicity to protein folding. *Nature*, 405:39–42, 2000.

[16] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[17] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucl. Acids Res.*, 28(1):235–242, 2000.

[18] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, August 2006.

[19] G. Blin, G. Fertin, and S. Vialette. New results for the 2-interval problem. In *Proc. Fifteenth Annual Combinatorial Pattern Matching Symposium (CPM 2004)*, volume 3109 of Lecture Notes in Computer Science, pages 311–322. Springer-Verlag, 2004.

[20] John Adrian Bondy. *Graph Theory with Applications*. Elsevier Science Ltd, 1976.

[21] R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. E. Strauss, and D. Baker. Rosetta in casp4: progress in ab initio protein structure prediction. *Proteins*, Suppl 5:119–126, 2001.

[22] S. Boris. A revised proof of the metric properties of optimally superimposed vector sets. *Acta Crystallographica Section A*, 58(5):506, 2002.

[23] J. U. Bowie and D. Eisenberg. An evolutionary approach to folding small α-helical proteins that uses sequence information and an empirical guiding fitness function. *Proceedings of the Academy of Sciences, (USA)*, 91(10):4436–4440, 1994.

[24] P. Bradley, D. Chivian, J. Meiler, K. M. S. Misura, C. A. Rohl, W. R. Schief, W. J. Wedemeyer, O. Schueler-Furman, P. Murphy, J. Schonbrun, C. E. M. Strauss, and D. Baker. Rosetta predictions in CASP5: Successes, failures, and prospects for complete automation. *Proteins: Struct. Funct. Genet.*, 53(S6):457–468, 2003.

[25] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: a program for macromolecular energy minimization and dynamics calculations. *J. Comput. Chem.*, 41(4):187–217, 1983.

[26] C. Bystroff and D. Baker. Prediction of local structure in proteins using a library of sequence-structure motifs. *Journal of Molecular Biology*, 281(3):565–577, 1998.

[27] C. Bystroff, K.T. Simons, K.F. Han, and D. Baker. Local sequence-structure correlations in proteins. *Current Opinion in Biotechnology*, 7(4):417–21, 1996.

[28] C. Bystroff, V. Thorsson, and D. Baker. HMMSTR: a hidden markov model for local sequence-structure correlations in proteins. *J. of Mol. Biol.*, 301(1):173–190, 2000.

[29] A.C. Camproux, P. Tuffery, J.P. Chevrolat, J.F. Boisvieux, and S. Hazout. Hidden Markov model approach for identifying the modular framework of the protein backbone. *Protein Eng.*, 12(12):1063–1073, 1999.

[30] A. A. Canutescu, A. A. Shelenkov, and R. L. Dunbrack. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Science*, 12(9):2001–2014, 2003.

[31] A. Caprara and G. Lancia. Structural alignment of large-size proteins via lagrangian relaxation. In *RECOMB '02: Proceedings of the sixth annual international conference on Computational biology*, pages 100–108, New York, NY, (USA), 2002. ACM.

[32] A. Cavalli, X. Salvatella, C. M.Dobson, and M. Vendruscolo. Protein structure determination from nmr chemical shift. *Proceedings of the National Academy of Sciences*, 104:9615–9620, 2007.

[33] S. Chakraborty and S. Biswas. Approximation algorithms for 3-d commom substructure identification in drug and protein molecules. In *WADS*, pages 253–264, 1999.

[34] B. Chazelle, C. Kingsford, and M. Singh. A Semidefinite Programming Approach to Side Chain Positioning with New Rounding Strategies. *INFORMS JOURNAL ON COMPUTING*, 16(4):380–392, 2004.

[35] E. Chen, L. Yang, and H. Yuan. Improved algorithms for largest cardinality 2-interval pattern problem. *Journal of Combinatorial Optimization*, 13:263–275, April 2007.

[36] Y. Chen, S. Ora, and W. Yair. Minimizing and learning energy functions for side-chain prediction. *Journal of Computational Biology*, 15(7):899–911, 2008.

[37] D. Chivian, D. E. Kim, L. Malmstrom, J. Schonbrun, C. A. Rohl, and D. Baker. Rosetta predictions in CASP5: Successes, failures, and prospects for complete automation. *Proteins: Struct. Funct. Genet*, 61(S7):157–166, 2005.

[38] V. Choi and N. Goyal. A combinatorial shape matching algorithm for rigid protein docking. In *CPM*, pages 285–296, 2004.

[39] V. Choi and N. Goyal. An efficient approximation algorithm for point pattern matching under noise. In *LATIN 2006: Theoretical Informatics, 7th Latin American Symposium, Valdivia, Chile, March 20-24, 2006, Proceedings.*, volume 3887 of *Lecture Notes in Computer Science*, pages 298–310. Springer-Verlag, 2006.

[40] M. Claessens, E. van Cutsem, I. Lasters, and S. Wodak. Modelling the polypeptide backbone with 'spare parts' from known protein structures. *Protein Eng.*, 2(5):335–345, 1989.

[41] C. Colovos and T. O. Yeates. Verification of protein structures: patterns of nonbonded atomic interactions. *Protein Science*, 2:1511–1519, 1993.

[42] A. Colubri, A. K. Jha, M. Y. Shen, A. Sali, R. S. Berry, T. R. Sosnick, and K. F. Freed. Minimalist representations and the importance of nearest neighbor effects in protein folding simulations. *Journal of Molecular Biology*, 363(4):835–857, November 2006.

[43] M. Comin, C. Guerra, and G. Zanotti. Proust: a comparison method of three-dimensional structure of proteins using indexing techniques. *Journal of Computational Biology*, 11:1061–1072, 2004.

[44] G. Cornilescu, F. Delaglio, and A. Bax. Protein backbone angle restraints from searching a database for chemical shift and sequence homology. *Journal of Biomolecular NMR*, 13:289–302, 1999.

[45] M. De Maeyer, J. Desmet, and I. Lasters. The dead-end elimination theorem: mathematical aspects, implementation, optimizations, evaluation, and performance. *Methods in Molecular Biology (Clifton, N.J.)*, 143:265–304, 2000.

[46] E. D. Demaine, S. Langerman, and J. O'Rourke. Geometric restrictions on producible polygonal protein chains. *Algorithmica*, 44(2):167–181, 2006.

[47] J. Desmet, M. D. Maeyer, B. Hazes, and I. Lasters. The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, 356(6369):539–542, April 1992.

[48] J. Desmet, M. De Maeyer, and I. Lasters. Theoretical and algorithmical optimization of the dead-end elimination theorem. In *Pacific Symposium in Biocomputing*, pages 122–133. World Scientific, 1997.

[49] K.A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–9, December 1985.

[50] R. L. Dunbrack and F. E. Cohen. Bayesian statistical analysis of protein side-chain rotamer preferences. *Protein Science*, 6(8):1661–1681, August 1997.

[51] R. L. Dunbrack and M. Karplus. Backbone-dependent rotamer library for proteins application to side-chain prediction. *Journal of Molecular Biology*, 230(2):543 – 574, 1993.

[52] R. L. Dunbrack and M. Karplus. Conformational analysis of the backbone-dependent rotamer preferences of protein sidechains. *Nature Structural Biology*, 1(5):334–340, May 1994.

[53] M. E. Dyer and A. M. Frieze. Planar 3dm is np-complete. *J. of Algorithms*, 7(2):174–184, 1986.

[54] B. Efron. The convex hull of a random set of points. *Biometrika*, 52(3-4):331–343, 1965.

[55] O. Eriksson, Y. Zhou, and A. Elofsson. Side chain-positioning as an integer programming problem. In *WABI '01: Proceedings of the First International Workshop on Algorithms in Bioinformatics*, pages 128–141, London, UK, 2001. Springer-Verlag.

[56] K. Fidelis, P. S. Stern, D. Bacon, and J. Moult. Comparison of systematic search and database methods for constructing segments of protein structure. *Protein Eng.*, 7(8):953–960, 1994.

[57] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, 1979.

[58] J.K. Gerard. Recognition of spatial motifs in protein structures. *Journal of Molecular Biology*, 285(4):1887–1897, 1999.

[59] M. Gerstein and M. Levitt. Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures. In *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, pages 59–67. AAAI Press, 1996.

[60] J. F. Gibrat, T. Madej, and S. H. Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, 6(3):377–385, 1996.

[61] Deborah Goldman, Christos H. Papadimitriou, and Sorin Istrail. Algorithmic aspects of protein structure similarity. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 512, Washington, DC, (USA), 1999. IEEE Computer Society.

[62] R. F. Goldstein. Efficient rotamer elimination applied to protein side-chains and related spin glasses. *Biophysical Journal*, 66(5):1335 – 1340, 1994.

[63] H. Gong, P.J. Fleming, and G.D. Rose. Building native protein conformation from highly approximate backbone torsion angles. *Proceedings of the National Academy of Sciences*, 102(45):16227–16232, 2005.

[64] H. P. Gong, Y. Shen, and G. D. Rose. Building native protein conformation from nmr backbone chemical shifts using monte carlo fragment assembly. *Protein Science*, 16:1515–1521, 2007.

[65] D. B. Gordon and S. L. Mayo. Branch-and-terminate: a combinatorial optimization algorithm for protein design. *Structure*, 7(9):1089 – 1098, 1999.

[66] J. Gramm. A polynomial-time algorithm for the matching of crossing contact-map patterns. In *WABI*, pages 38–49, 2004.

[67] J. Gramm. A polynomial-time algorithm for the matching of crossing contact-map patterns. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 1(4):171–180, 2004.

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

[69] T. Hamelryck, J.T. Kent, and K. Anders. Sampling realistic protein conformations using local structural bias. *PLoS Computational Biology*, 2(9):e131, 2006.

[70] T. Hamelryck, J.T. Kent, and A. Krogh. Sampling realistic protein conformations using local structural bias. *PLoS Comput Biol*, 2(9), 2006.

[71] K.F. Han, C. Bystroff, and D. Baker. Three-dimensional structures and contexts associated with recurrent amino acid sequence patterns. *Protein Science*, 6(7):1587–1590, 1997.

[72] M. Hao, S. Rackovsky, A. Liwo, M.R. Pincus, and H.A. Scheraga. Effects of compact volume and chain stiffness on the conformations of native proteins. *Proceedings of the Academy of Sciences, (USA)*, 89:6614–6618, 1992.

[73] N. Haspel, C. Tsai, H. Wolfson, and R. Nussinov. Reducing the computational complexity of protein folding via fragment folding and assembly. *Protein Science*, 12(6):1177–1187, 2003.

[74] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the Academy of Sciences, (USA)*, 89(22):10915–10919, 1992.

[75] L. Holm and C. Sander. Database algorithm for generating protein backbone and side-chain co-ordinates from a c[alpha] trace : Application to model building and detection of co-ordinate errors. *Journal of Molecular Biology*, 218(1):183 – 194, 1991.

[76] L. Holm and C. Sander. Evaluation of protein models by atomic solvation preference. *Journal of Molecular Biology*, 225:93–105, 1992.

[77] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233(1):123–138, September 1993.

[78] J. B. Holmes and J. Tsai. Some fundamental aspects of building protein structures from fragment libraries. *Protein Science*, 13(6):1636–1650, 2004.

[79] R. W. W. Hooft, G. Vriend, C. Sander, and E. E. Abola. Errors in protein structures. *Nature*, 381:272, 1996.

[80] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[81] J. Hwang and W. Liao. Side-chain prediction by neural networks and simulated annealing optimization. *Protein Eng.*, 8(4):363–370, 1995.

[82] Y. Inbar, H. Benyamini, R. Nussinov, and H. J. Wolfson. Protein structure prediction via combinatorial assembly of sub-structural units. *Bioinformatics*, 19(S1):158–168, 2003.

[83] J. Janin, S. Wodak, M. Levitt, and B. Maigret. The conformation of amino acid side-chains in proteins. *Journal of Molecular Biology*, 125(3):357–386, 1978.

[84] M. S. Johnson, N. Srinivasan, R. Sowdhamini, and T. L. Blundell. Knowledge-based protein modeling. *Critical Reviews in Biochemistry and Molecular Biology*, 29(1):1–68, 1994.

[85] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195–202, September 1999.

[86] T. A. Jones and S. Thirup. Using known substructures in protein model building and crystallography. *EMBO Journal*, 5:819–823, 1986.

[87] R. L. Dunbrack Jr. Rotamer libraries in the 21st century. *Current Opinion in Structural Biology*, 12(4):431–440, August 2002.

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

[89] D. A. Keller, M. Shibata, E. Marcus, R. L. Ornstein, and R. Rein. Finding the global minimum: a fuzzy end elimination implementation. *Protein Engineering*, 8(9):893–904, 1995.

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

[91] C. L. Kingsford, B. Chazelle, and M. Singh. Solving and analyzing side-chain positioning problems using linear and integer programming. *Bioinformatics*, 21(7):1028–1039, 2005.

[92] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Discret. Math.*, 5(3):422–427, 1992.

[93] R. Kolodny, P. Koehl, L. Guibas, and M. Levitt. Small libraries of protein fragments model native protein structures accurately. *Journal of Molecular Biology*, 323:297–307, 2002.

[94] R. Kolodny and N. Linial. Approximate protein structural alignment in polynomial time. *Proceedings of the Academy of Sciences, (USA)*, 101:12201 – 12206, 2004.

[95] G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal pdb structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In *RECOMB '01: Proceedings of the fifth annual international conference on Computational biology*, pages 193–202, New York, NY, (USA), 2001. ACM.

[96] G. Lancia and S. Istrail. Protein structure comparison: Algorithms and applications. In *Mathematical Methods for Protein Structure Analysis and Design*, pages 1–33, 2003.

[97] R. A. Laskowski, M. W. Macarthur, D. S. Moss, and J. M. Thornton. PROCHECK: a program to check the steroechemical quality of protein structures . *Journal of Applied Crystallography.*, 26:283–291, 1993.

[98] I. Lasters, M. De Maeyer, and J. Desmet. Enhanced dead-end elimination in the search for the global minimum energy conformation of a collection of protein side chains. *Protein Eng.*, 8(8):815–822, 1995.

[99] I. Lasters and J. Desmet. The fuzzy-end elimination theorem: correctly implementing the side chain placement algorithm based on the dead-end elimination theorem. *Protein Eng.*, 6(7):717–722, 1993.

[100] C. Lee and S. Subbiah. Prediction of protein side-chain conformation by packing optimization. *Journal of Molecular Biology*, 217(2):373 – 388, 1991.

[101] J. Lee, S. Kim, and J. Lee. Protein structure prediction based on fragment assembly and parameter optimization, Biophysical Chemistry. *Biophysical Chemistry*, 115(2-3):209–214, 2005.

[102] C. Lemmen and T. Lengauer. Computational methods for the structural alignment of molecules. *Journal of Computer Aided Molecular Design*, 14(3):215–232, March 2000.

[103] C. Levinthal. Are there pathways for protein folding? *Extrait du Journal de Chimie Physique*, 65(1), 1968.

[104] M. Levitt. Accurate modeling of protein conformation by automatic segment matching. *Journal of Molecular Biology*, 226(2):507–533, 1992.

[105] H. Li, K. Zhang, and T. Jiang. The regularized em algorithm. In *AAAI*, pages 807–812, 2005.

[106] H. Li and Y. Zhou. Scud: Fast structure clustering of decoys using reference state to remove overall rotation. *Journal of Computational Chemistry*, 26(11):1189–92, 2005.

[107] M. Li and P. Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer, 1997.

[108] S. C. Li, J. Xu, X. Gao, D. Bu, and M. Li. Designing Succinct Structural Alphabets. *ISMB'08*, 2008.

[109] S. Liang and N. V. Grishin. Side chain modeling with an optimized scoring function. *Protein Science*, 11(2):322 – 331, February 2002.

[110] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[111] L. L. Looger and H. W. Hellinga. Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: implications for protein design and structural genomics. *Journal of Molecular Biology*, 307(1):429–445, March 2001.

[112] S. C. Lovell, J. M. Word, J. S. Richardson, and D. C. Richardson. The penultimate rotamer library. *Proteins*, 40(3):389–408, August 2000.

[113] R. Luthy, J. U. Bowie, and D. Eisenberg. Assessment of protein models with three-dimensional profiles. *Nature*, 356(6364):83–85, 1992.

[114] S. C. Lovell M. A. DePristo, P. I. de Bakker and T. L. Blundell. Ab initio construction of polypeptide fragments: efficient generation of accurate, representative ensembles. *Proteins*, 51(1):41–55, April 2003.

[115] M. D. Maeyer, J. Desmet, and I. Lasters. All in one: a highly detailed rotamer library improves both accuracy and speed in the modelling of sidechains by dead-end elimination. *Folding and Design*, 2(1):53 – 66, 1997.

[116] K.V. Mardia, C.C. Taylor, and G.K. Subramaniam. Protein bioinformatics and mixtures of bivariate von Mises distributions for angular data. *Biometrics*, 63(2):505–512, June 2007.

[117] B. J. McConkey, V. Sobolev, and M. Edelman. Discrimination of native protein structures using atom-atom contact scoring. *Proc Natl Acad Sci (USA)*, 100:3215–3220, 2003.

[118] M. J. McGregor, S. A. Islam, and M. J. E. Sternberg. Analysis of the relationship between side-chain conformation and secondary structure in globular proteins. *Journal of Molecular Biology*, 198(2):295 – 310, 1987.

[119] J. Mendes, A. M. Baptista, M. A. Carrondo, and C. M. Soares. Improved modeling of side-chains in proteins with rotamer-based methods: A flexible rotamer model. *Proteins: Structure, Function, and Genetics*, 37(4):530–543, 1999.

[120] J. Moult, K. Fidelis, B. Rost, T. Hubbard, and A. Tramontano. Critical assessment of methods of protein structure prediction (casp):round 6. *Proteins: Struct. Funct. Genet.*, 61:3–7, 2005.

[121] W. Mulzer and G. Rote. Minimum-weight triangulation is np-hard. *J. ACM*, 55(2):1–29, 2008.

[122] L. Pauling and R. B. Corey. The pleated sheet, a new layer configuration of polypeptide chains. *Proceedings of the Academy of Sciences, (USA)*, 37(5):251–256, 1951.

[123] J. Peng and J. Xu. Boosting protein threading accuracy. *Proceedings of RECOMB 2009*, 2009.

[124] N. A. Pierce, J. A. Spriet, J. Desmet, and S. L. Mayo. Conformational splitting: A more powerful criterion for dead-end elimination. *Journal of Computational Chemistry*, 21(11):999–1009, 2000.

[125] N. A. Pierce and E. Winfree. Protein Design is NP-hard. *Protein Eng.*, 15(10):779–782, 2002.

[126] J. W. Ponder and F. M. Richards. Tertiary templates for proteins : Use of packing criteria in the enumeration of allowed sequences for different structural classes. *Journal of Molecular Biology*, 193(4):775 – 791, 1987.

[127] J. Qian, S. C. Li, D. Bu, M. Li, and J. Xu. Finding compact structural motifs. In *Combinatorial Pattern Matching, 18th Annual Symposium, CPM 2007, Proceedings*, volume 4580, pages 142–149. Springer, 2007.

[128] G.N. Ramachandran and V. Sasisekharan. Conformation of polypeptides and proteins. *Advances in Protein Chemistry*, 23:283–438, 1968.

[129] C. A. Rohl, C. E. Strauss, K. M. Misura, and D. Baker. Protein structure prediction using Rosetta. *Methods Enzymol*, 383:66–93, 2004.

[130] A. Roitberg and R. Elber. Modeling side chains in peptides and proteins: Application of the locally enhanced sampling and the simulated annealing methods to find minimum energy conformations. *The Journal of Chemical Physics*, 95(12):9277–9287, 1991.

[131] M.J. Rooman, J. Rodriguez, and S.J. Wodak. Automatic definition of recurrent local structure motifs in proteins. *Journal of Molecular Biology*, 213(2):327–336, 1990.

[132] R. Samudrala and J. Moult. An all-atom distance-dependent conditional probability discriminatory function for protein structure prediction. *Journal of Molecular Biology*, 275:895–916, 1998.

[133] Y. Shen and A. Bax. Protein backbone chemical shifts predicted from searching a database for torsion angle and sequence homology. *Journal of Biomolecular NMR*, 38(4):289–302, 2007.

[134] Y. Shen, O. Lange, F. Delaglio, P. Rossi, G. Liu J. M. Aramini, A. Eletsky, B. Wu, K. K. Singarapu, A. Lemak, A. Ignatchenko, C. Arrowsmith, T. Szyperski, G. T. Montelione, D. Baker, and A. Bax. Consistent blind protein structure generation from nmr chemical shift data. *Proceedings of the National Academy of Sciences*, 105:4685–4690, 2008.

[135] D. Shortle. Composites of local structure propensities: Evidence for local encoding of long-range structure. *Protein Science*, 11(1):18–26, 2002.

[136] D. Shortle, K. T. Simons, and D. Baker. Clustering of low-energy conformations near the native structures of small proteins. *Proc Natl Acad Sci (USA)*, 95:11158–11162, 1998.

[137] 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.

[138] I. Simon, L. Glasser, and H. A. Scheraga. Calculation of protein conformation as an assembly of stable overlapping segments: application to Bovine pancreatic trypsin inhibitor. *Proceedings of the Academy of Sciences, (USA)*, 88(9):3661–3665, 1991.

[139] K. T. Simons, R. Bonneau, I. Ruczinski, and D. Baker. Ab initio protein structure prediction of casp iii targets using rosetta. *Proteins*, Suppl 3:171–176, 1999.

[140] K. T. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *Journal of Molecular Biology*, 268(1):209–25, Apr 1997.

[141] K.T. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of Protein Tertiary Structures from Fragments with Similar Local Sequences using Simulated Annealing and Bayesian Scoring Functions. *Journal of Molecular Biology*, 268, 1997.

[142] G. E. Sims and S. Kim. A method for evaluating the structural quality of protein models by using higher-order varphi-psi pairs scoring. *Proceedings of the National Academy of Sciences*, 103(12):4428–4432, 2006.

[143] G. E. Sims and S. H. Kim. A method for evaluating the structural quality of protein models by using higher-order varphi-psi pairs scoring. *Proceedings of the National Academy of Sciences*, 103(12):4428–4432, 2006.

[144] A. P. Singh and D. L. Brutlag. Hierarchical protein structure superposition using both secondary structure and atomic representations. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*, pages 284–293. AAAI Press, 1997.

[145] Harshinder Singh, Vladimir Hnizdo, and Eugene Demchuk. Probabilistic model for two dependent circular variables. *Biometrika*, 89(3):719–723, 2002.

[146] M. J. Sippl. Recognition of errors in three-dimensional structures of proteins. *Proteins: Struct. Funct. Genetics*, 17:355–362, 1993.

[147] A. G. Street and S. L. Mayo. Intrinsic beta-sheet propensities result from van der waals interactions between side chains and the local backbone. *Proceedings of Nattional Academy of Science (USA)*, 96(16):9074–9076, August 1999.

[148] S. Subramaniam, D. K. Tcheng, and J. M. Fenton. A knowledge-based method for protein structure refinement and prediction. *Proceedings of International Conference Intelligent Systems for Moleculur Biology*, 4:218–229, 1996.

[149] A. V. Tendulkar, M. A. Sohoni, B. Ogunnaike, and P. P. Wangikar. A geometric invariant-based framework for the analysis of protein conformational space. *Bioinformatics*, Advance Access, August 2005.

[150] S. C. E. Tosatto. The victor/frst function for model quality esitmation. *J. Computational Biology*, 12(10), 2005.

[151] P. Tuffery, C. Etchebest, S. Hazout, and R. Lavery. A new approach to the rapid determination of protein side chain conformations. *J Biomol Struct Dyn*, 8(6):1267–1289, 1991.

[152] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380, 1991.

[153] R. Unger, D. Harel, S. Wherland, and J. L. Sussman. A 3D building blocks approach to analyzing and predicting structure of proteins. *Proteins: Struct. Funct. Genet.*, 5(4):355–373, 1989.

[154] M. Vásquez. An evaluvation of discrete and continuum search techniques for conformational analysis of side chains in proteins. *SO: Biopolymers*, 36(1):53–70, 1995.

[155] Stéphane Vialette. On the computational complexity of 2-interval pattern matching problems. *Theor. Comput. Sci.*, 312(2-3):223–249, 2004.

[156] C. A. Voigt, D. B. Gordon, and S. L. Mayo. Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design. *Journal of Molecular Biololgy*, 299(3):789–803, June 2000.

[157] B. Wallner and A. Elofsson. Can correct protein models be identified? *Protein Science*, 12(5):1073–1086, 2003.

[158] G. Wang and R. L. Dunbrack Jr. PISCES: a protein sequence culling server. *Bioinformatics*, 19(12):1589–1591, 2003.

[159] K. Wang, B. Fain, M. Levitt, and R. Samudrala. Improved protein structure selection using decoy-dependent discriminatory functions. *BMC Structural Biology*, 4(1):8, 2004.

[160] J. J. Wendoloski and F. R. Salemme. Probit: a statistical approach to modeling proteins from partial coordinate data using substructure libraries. *J. Mol. Graph.*, 10(2):124–126, 1992.

[161] Wikipedia. Amino acid and peptite bond. http://en.wikipedia.org/wiki/Amino_acid.

[162] J. M. Word, S. C. Lovell, T. H. LaBean, H. C. Taylor, M. E. Zalis, B. K. Presley, J. S. Richardson, and D. C. Richardson. Visualizing and quantifying molecular goodness-of-fit: small-probe contact dots with explicit hydrogen atoms. *Journal of Molecular Biology*, 285(4):1711 – 1733, 1999.

[163] J.Michael Word, Simon C. Lovell, Jane S. Richardson, and David C. Richardson. Asparagine and glutamine: using hydrogen atom contacts in the choice of side-chain amide orientation. *Journal of Molecular Biology*, 285(4):1735 – 1747, 1999.

[164] S. Wu, J. Skolnick, and Y. Zhang. Ab initio modeling of small proteins by iterative tasser simulations. *BMC Biology*, 5(17), 2007.

[165] Z. Xiang and B. Honig. Extending the accuracy limits of prediction for side-chain conformations. *Journal of Molecular Biology*, 311(2):421 – 430, 2001.

[166] J. Xu. Protein fold recognition by predicted alignment accuracy. *ACM/IEEE Transactions on Computational Biology and Bioinformatics*, 2(2):157–165, 2005.

[167] J. Xu and B. Berger. Fast and accurate algorithms for protein side-chain packing. *Journal of ACM*, 53(4):533–557, 2006.

[168] J. Xu, F. Jiao, and B. Berger. A parameterized algorithm for protein structure alignment. *Journal of Computational Biology*, 14(5):564–577, 2007.

[169] J. Xu and M. Li. Assessment of RAPTORs linear programming approach in CAFASP3. *Proteins*, 53(S6):579–584, 2003.

[170] A. Zemla. LGA: a method for finding 3D similarities in protein structures. *Nucl. Acids Res.*, 31(13):3370–3374, 2003.

[171] Y. Zhang. Template-based modeling and free modeling by I-TASSER in CASP7. *Proteins*, Suppl 8:108–117, 2007.

[172] Y. Zhang. Progress and challenges in protein structure prediction. *Current Opinion in Structural Biology*, 18(3):342–348, June 2008.

[173] Y. Zhang, A. Arakaki, and J. Skolnick. TASSER: An automated method for the prediction of protein tertiary structures in CASP6. *Proteins*, 61(S7):91–98, September 2005.

[174] Y. Zhang and J. Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

[175] Y. Zhang and J. Skolnick. Spicker: Approach to clustering protein structures for near-native model selection. *Journal of Computational Chemistry*, 25:865–871, 2004.

[176] F. Zhao, S. C. Li, B. W. Sterner, and J. Xu. CRFSampler: Discriminative learning for protein conformation sampling. *Proteins*, 73(1):228–240, 2008.

[177] H. Zhou and Y. Zhou. SPARKS 2 and SP^3 servers in CASP6. *Proteins: Structure, Function, and Bioinformatics*, 61(S7):152–156, 2005.

[178] R Zwanzig, A Szabo, and B Bagchi. Levinthal's Paradox. *Proceedings of the National Academy of Sciences*, 89(1):20–22, 1992.