Summer 2012

De Novo Protein Structure Modeling from Cryoem Data Through a Dynamic Programming Algorithm in the Secondary Structure Topology Graph

Kamal H. Al Nasr
Old Dominion University

DE NOVO PROTEIN STRUCTURE MODELING FROM CRYOEM DATA THROUGH A DYNAMIC PROGRAMMING ALGORITHM IN THE SECONDARY STRUCTURE TOPOLOGY GRAPH

by

Kamal H. Al Nasr

B.Sc. February 2003, Yarmouk University, Jordan
M.Sc. August 2005, Yarmouk University, Jordan
M.Sc. October 2011, New Mexico State University, NM, USA

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE
OLD DOMINION UNIVERSITY
August 2012

Approved by:

Jing He (Director)

Desh Ranjan (Member)

Yaohang Li (Member)

Mohammad Zubair (Member)

Shuiwang Ji (Member)

Lesley Greene (Member)

ABSTRACT

DE NOVO PROTEIN STRUCTURE MODELING FROM CRYOEM DATA THROUGH A DYNAMIC PROGRAMMING ALGORITHM IN THE SECONDARY STRUCTURE TOPOLOGY GRAPH

Kamal Al Nasr
Old Dominion University, 2012
Director: Jing He

Proteins are the molecules carry out the vital functions and make more than the half of dry weight in every cell. Protein in nature folds into a unique and energetically favorable 3-Dimensional (3-D) structure which is critical and unique to its biological function. In contrast to other methods for protein structure determination, Electron Cryo-microscopy (CryoEM) is able to produce volumetric maps of proteins that are poorly soluble, large and hard to crystallize. Furthermore, it studies the proteins in their native environment. Unfortunately, the volumetric maps generated by current advances in CryoEM technique produces protein maps at medium resolution about (~5 to 10Å) in which it is hard to determine the atomic-structure of the protein. However, the resolution of the volumetric maps is improving steadily, and recent works could obtain atomic models at higher resolutions (~3Å).

De novo protein modeling is the process of building the structure of the protein using its CryoEM volumetric map. Thereupon, the volumetric maps at medium resolution generated by CryoEM technique proposed a new challenge. At the medium resolution, the location and orientation of secondary structure elements (SSE) can be visually and computationally identified. However, the order and direction (called protein topology) of the SSEs detected from the CryoEM volumetric map are not visible. In order to determine the protein structure, the topology of the SSEs has to be figured out and then the backbone can be built. Consequently, the topology problem has become a bottle neck for protein modeling using CryoEM.

In this dissertation, we focus to establish an effective computational framework to derive the atomic structure of a protein from the medium resolution CryoEM volumetric maps. This framework includes a topology graph component to rank effectively the topologies of the SSEs and a model building component. In order to generate the small subset of candidate topologies, the problem is translated into a layered graph representation. We developed a dynamic programming algorithm (TopoDP) for the new representation to overcome the problem of large search space. Our approach shows the improved accuracy, speed and memory use when compared with existing methods. However, the generating of such set was infeasible using a brute force method. Therefore, the topology graph component effectively reduces the topological space using the geometrical features of the secondary structures through a constrained K-shortest paths method in our layered graph. The model building component involves the bending of a helix and the loop construction using skeleton of the volumetric map. The forward-backward CCD is applied to bend the helices and model the loops.

I dedicate this work to the soul of my mother, who believed in diligence and encouraged me throughout my study. She was always the source of moral support. She faced her death bravely. During her terminal illness, the time I first proposed this work on September 2010, she has never let me notice her suffering not to distract my attention. Her soul kept me working when I wanted to give up. May ALLAH shower her soul with mercy and blessings.

I dedicate this dissertation to my wife, Ruba Jebril. A special feeling of gratitude to her for the patience, support and understanding during my doctorate study. She always tried to provide me an excellent and ideal atmosphere to conduct research. She has never left my side; I will always appreciate all she has done.

I also dedicate this work to my wonderful family for their endless encouragement and patience. I am grateful to them for being a source of motivation and inspiration.

Finally, this work is dedicated to all of my friends and all those who believe in the importance and richness of learning.

ACKNOWLEDGMENTS

I would like to acknowledge the inspirational instruction and guidance of my advisor and committee chair Dr. Jing He. I would like to thank her for the countless hours of reflecting, reading, encouraging, and most of all patience throughout the entire research. She has given me a deep appreciation and love for the beauty and detail of the subject of this dissertation.

I wish to thank my committee members who were more than generous with their expertise and precious time. A special thanks to Dr. Desh Ranjan and Dr. Mohammad Zubair for their excitement and willingness to provide feedback made the completion of this research an enjoyable experience. Special thanks also go to Dr. Yaohang Li, Dr. Shuiwang Ji and Dr. Lesley Greene for serving in my committee and the invaluable counseling they have provided.

I would like to express my deepest gratitude to my colleagues. They were always willing to help and give their best suggestions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Proteins are complex molecules play an essential role and involved in every process within cells. The Protein is a linear copolymer made of sequence of molecules called amino acids, referred to residue, arranged linearly. Naturally, there are 20 kinds of amino acids build up all proteins. Amino acids are made of amine group (H_2N), carboxyl group (COOH) and a side chain (R) that differs from one amino acid to another. Amino acids are connected by peptide bonds between the carboxyl and amino groups of any two adjacent amino acids to form one protein [3]. Amino acids can be classified into several groups according to some features and properties they may have. They can be classified according to their charge, hydrophilic or hydrophobic, size and functional group. Such properties play a main role in protein structure folding and protein-protein interactions. For example the location of some amino acids determined by its charge, while hydrophobic amino acids (i.e., Leu, Ile, and Val) buried in the middle of the protein, hydrophilic amino acids like to be in the outer shell of the protein. Fig. 1 shows the 20 amino acids found in the nature and some groups they are classified into. University of California at San Francisco (UCSF) Chimera [2] was used to view the structure of proteins shown in the course of this dissertation.

The general chemical formula of the amino acid is $H_2NC_\alpha HRCOOH$, where R is a chemical side chain (one atom or a group of atoms) differ from one amino acid to another. In this formula, the carboxyl group, amino group, and side chain are connected to the same carbon atom which called alpha-carbon (C_α). All amino acids are made up of a backbone consists of main element atoms carbon, oxygen, hydrogen, and nitrogen, where a bonded sequence of nitrogen and two carbon atoms, one of them is C_α. Fig. 2 shows these main element atoms and the side chain, represented by R. Each amino acid has its own side chain attached to alpha-carbon varies in size from just a hydrogen atom as in glycine to heterocyclic group as in tryptophan (see Fig. 1). The sequence of amino acids form the protein are connected to each other by a peptide bond between the carbon (C) atom from carboxyl group of one amino acid to nitrogen (N) atom from amine group of the following amino acid and form what is called a polypeptide chain. Fig. 2 depicts the formation of peptide bond between two residues. When two residues form a peptide bond, they lose a water molecule.

1.1 PROTEIN STRUCTURE

Protein in nature folds into a unique and energetically favorable 3-dimensional (3-D) structure which is critical and unique to its biological function [4,5] which can vary from structural over immunological to material and signal transporting, cell adhesion and cell cycle [6,7]. A particular function can also be achieved by a stable complex of proteins [8]. Proteins participate in every process within cells [8]. Enzymes proteins are vital to metabolism. Several proteins have structural functions; such function maintains the shape of the cell by forming a system of scaffolding. Examples are actin and myosin in muscle [8]. Proteins are also important in animals' diet in which it broken down into free amino acids that are used in metabolism [8].

Proteins can be classified based on the structure and function they carry out. According to the molecular structure, it can be divided into three groups, fibrous proteins, globular proteins, and conjugated proteins. Fibrous proteins are fiber-like proteins that help in structural purposes in organism. One example of such proteins is collagen. Globular proteins are knot-like proteins serve in hormones or enzymes. Hemoglobin is a good example of globular proteins. On the other hand, conjugated proteins are globular proteins have sugars or nucleic acids bound to it [8]. Proteins can be divided according to their functions into enzymes, hormones, transport proteins, antibodies, motor proteins, receptor proteins, structural proteins, signaling proteins and storage proteins [8]. One possible classification of proteins is according to their location relative to the cell. For example, membrane proteins are the proteins that located in the cell membrane lipid bi-layer. Internal and external proteins are located within living cell and outside the cell respectively. Virus proteins are proteins that can be found only in virus organism. Example of virus proteins is Gp41 present as a coat for HIV virus [8].

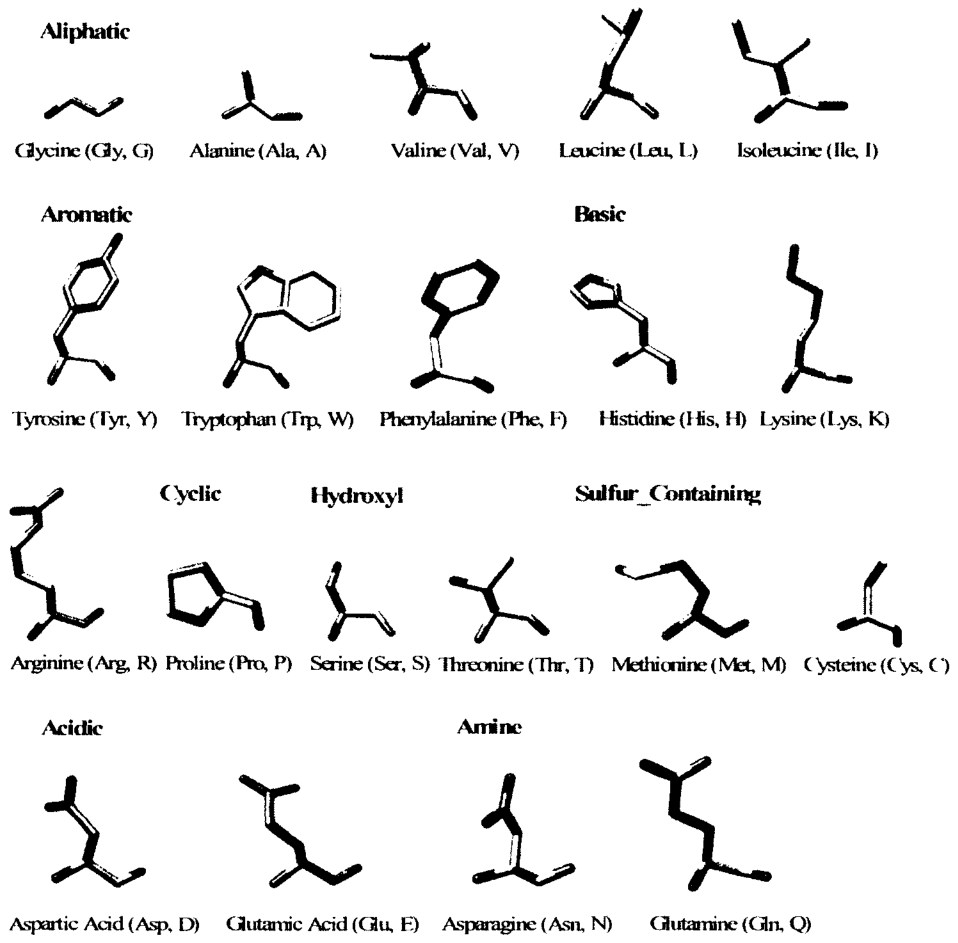


Figure 1. The structures of the 20 amino acids build up the proteins in nature. On the top of some amino acids is the name of the group they are classified into. The backbone of the amino acid is the bottom part (clearly found in Glycine because it has no side chain), the side chain is the upper part of each amino acid. The three-letter and one-letter abbreviation code of each amino acid is listed between two brackets.

The unique conformation in which the protein folds into is called a native structure [9]. The sequence of amino acids build up the protein ultimately determine its native structure, in which corresponds to the favorable energy of the molecule [10]. Structure of Protein can be expressed to four levels as follow:

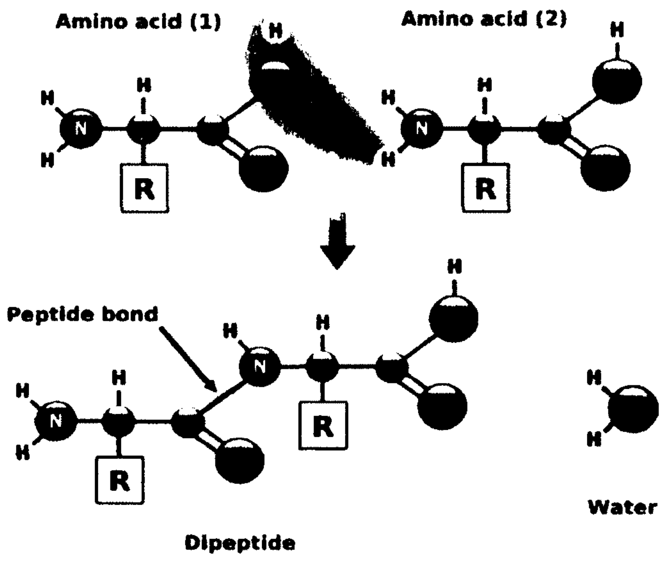


Figure 2. The formation of the peptide bond between two amino acids. Public domain permission (Wikipedia.com)

1. **Primary Structure:** refers to the sequence number and order presents of amino acids in the protein. The two ends of the polypeptide chain of the protein are referred to amino terminus (N-terminus) to the left and the carboxyl terminus (C-terminus) to the right. The numbering of amino acids start from N-terminus toward C-terminus. Fig. 4A depicts a portion of primary structure.

2. **Secondary structure:** refers to a regular sub-conformational structure formed by consecutive amino acids stabilized by hydrogen bonds (H-bonds). The most common examples of secondary structures are alpha-helices (α-Helices), beta-sheets (β-

Sheets), and turns/loops (see Fig. *4B*). The two secondary structures, helices and sheets, are geometry stabilized by hydrogen bonds between amino acids peptide groups. Different regions on the polypeptide chain may adopt different secondary structures according to the primary sequence of amino acids in the protein.

a. **α-Helix:** The conformation of the α-helix is stabilized by H-bonding between N-H group and C=O group of peptide bonds four residues apart. The orientation of such a conformation produces a helical coiling of the peptide backbone such that the side chain groups stem out of the helix coil and perpendicular to its axis. Not all amino acids prefer forming alpha helices due to some constraints of their side chains. Amino acids such as alanine (Ala), asparatic acid (Asp), glutamic acid (Glu), isoleucine (Ile), leucine (Leu), and methionine (Met) favor the formation of α-helices, whereas, glycine (Gly) and proline (Pro) favor disruption of the helix [11,12,13,14]. Among types of local structure in proteins, the α-helix is the most regular and the most predictable from sequence, as well as the most common. The average length of α-helices in proteins is 12 residues [15] which corresponds to 3.33 turns and 18Å length on 3-D space (1.5Å is the rise length of the amino acid in α-helix). Fig. *3A* shows the geometry of a helix and the H-bonding formed to stabilize it.

b. **β-sheet:** the second common conformation after α-helix. It is composed of two or more different segments (strands) of stretches along the primary structure of the protein. The conformation of the β-sheet is stabilized by H-bonding between N-H group of one strand with C=O group of an adjacent strand. The average number of strands in a β-sheet is between two to as many as 22 strands [15]. The average length of a strand is six residues. However, each strand may have up to 15 residues [15]. β-sheets are either parallel or anti-parallel. In parallel sheets following peptide chains proceed in the same direction, whereas, in anti-parallel sheets following chains are aligned in opposite directions. Fig. *3B* shows two strands in a sheet with hydrogen bonding between them. Fig. *4B* shows an example of two anti-parallel strands.

c. **Turns:** refers to the close approach of two consecutive C_α atoms (less than 7Å) in which residues do not form any kind of other secondary structure (i.e., α-helix or β-sheet) [16]. Turns can be classified into many types according to the number of residues separate the two ends residues [17]. α-turn is one example of such classification. In α-turns, the two residues at the ends of the turn are four peptide bonds apart. Other types include β-turn, γ-turn, δ-turn and π-turn. A special case of turn is β-bends which connects two successive anti-parallel beta strands [15]. A loop is an extended or disordered structure. Most of proteins contain more than 60 residues have one or more of Ω-loop. Ω-loop contains six to 16 residues and it is adopt the Greek uppercase letter omega [15].

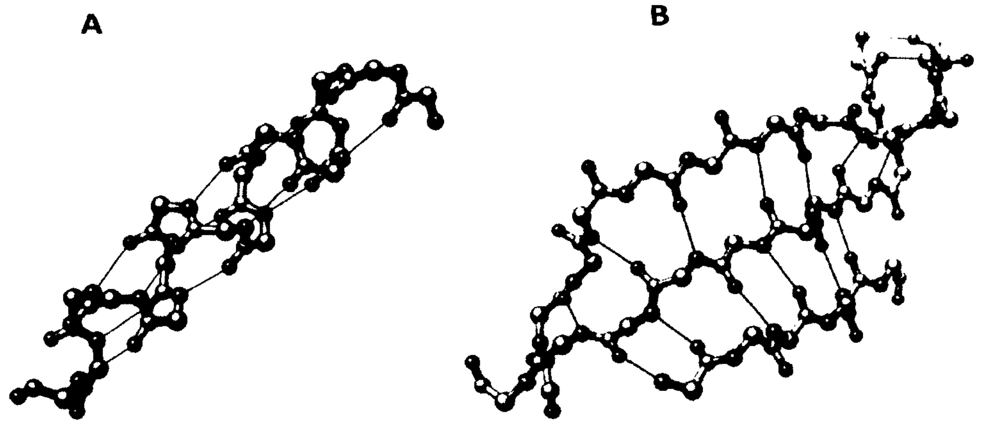


Figure 3. The geometry of α-helices and β-sheets. (A) The geometry of α-helix and the H-bonding (thin lines) formed to stabilize the structure. (B) The geometry of a sheet contains three strands forming and the H-bonding (thin lines). Two secondary structures are taken from Alzheimer's Amyloid Precursor Protein (PDB ID: 2FMA).

3. Tertiary Structure: refers to the complete 3-D structure of a single protein molecule. It defines the spatial relationship of different secondary structures to one another within a polypeptide chain and also describes the relationship of different domains to one another within a protein. The physics of the intra-protein and the environment interactions beside the primary chain govern the interaction of different domains such as H-bonding, hydrophobic interactions, electrostatic interactions and Van Der Waals forces. An example of tertiary structure is shown in Fig. 4*C*.

4. **Quaternary Structure:** refers to multiple polypeptide chains may form the protein molecule. The quaternary structure is stabilized by the same non-covalent interactions and disulfide bonds as the tertiary structure. Fig. 4*D* shows one example of quaternary structure.

A EACKFLHQERMDVCETHLHWHTVAKETCSEKSTNLHDYGMLLPCGIDKFRGVEFVCCPL

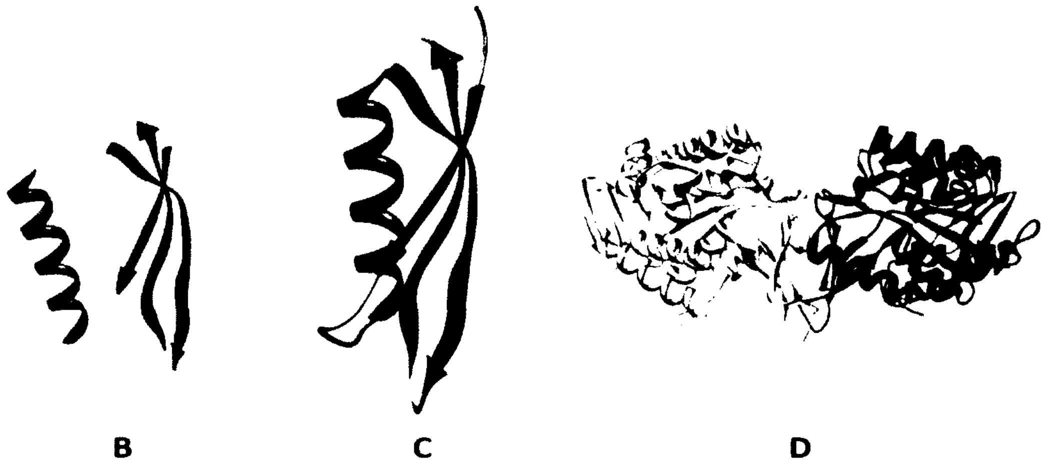


Figure 4. The four levels of protein structures. (A) The primary structure, only ordered sequence of amino acids for Alzheimer's Amyloid Precursor Protein (PDB ID: 2FMA) (B) Secondary structures, β-sheet shown as segments of stretches, helices are spiral, and loop/turn connects other secondary structures (PDB ID: 2FMA). (C) Tertiary structure, complete 3-D structure of a single protein molecule (PDB ID: 2FMA). (D) Quaternary structure, multiple polypeptide, of a sugar kinase (PDB ID: 4E69).

Several numbers of experimental techniques are used to determine the structure of proteins. The most common technique is X-ray crystallography which measures the 3-D density distribution of electrons in the protein. Another, valuable technique is NMR spectroscopy. Unfortunately, these techniques are expensive and time consuming (sometimes longer than a year) [18]. Therefore, developing new computational methods to predict the structure of proteins has given a considerable attention and effort [19].

1.1.1 Protein Structure Determination

The early effort on protein determination began in 1958 when British scientists John Kendrew and Max Perutz published the very first high resolution protein structure. For this work which had started as early as 1937, Kendrew and Perutz shared the 1962 Nobel Prize in chemistry [20]. Two techniques are used to determine structure of proteins:

1. **X-ray Crystallography:** the most common technique used to solve 3-D protein structures. Most of proteins found on Protein Data Bank (PDB) [21,22] have been solved by this technique. A beam of X-rays hits a crystal then diffracts into many specific directions. A 3-D picture of the density of electrons within the crystal can be produced by a crystallographer from diffraction patterns, angles and intensities, of these diffracted beams. The actual positions of atoms and their chemical bonds can be determined by this produced electron density. The set up process of this experimental technique is still one of the difficulties. The main difficulty is the recovery of phase information. The diffraction pattern produced from the crystallographer has only the information about angles and intensities of the beams that hit the detector. The phase of the beams is lost and the method to recover it is yet hard and iterative. Fig. 5*B* shows a workflow for solving the structure of a molecule by X-ray crystallography.

2. **Nuclear Magnetic Resonance (NMR):** is the second popular technique used to determine the structure of the proteins. A significant number of determined protein structures have been resolved by NMR. The main idea of the technique is to place the

protein inside a strong magnetic field and irradiate it with radio-frequency pulses. The energy radiated back at specific resonance frequency depends on neighborhood. The distances between neighboring hydrogen atoms are measured by a Nuclear Overhauser Effects (NOEs). Some constraints such as primary structure and reference protein geometry are used to calculate the 3-D structure of the protein in addition to the calculated distances by NOEs. Fig. 5A shows one NMR instrument.

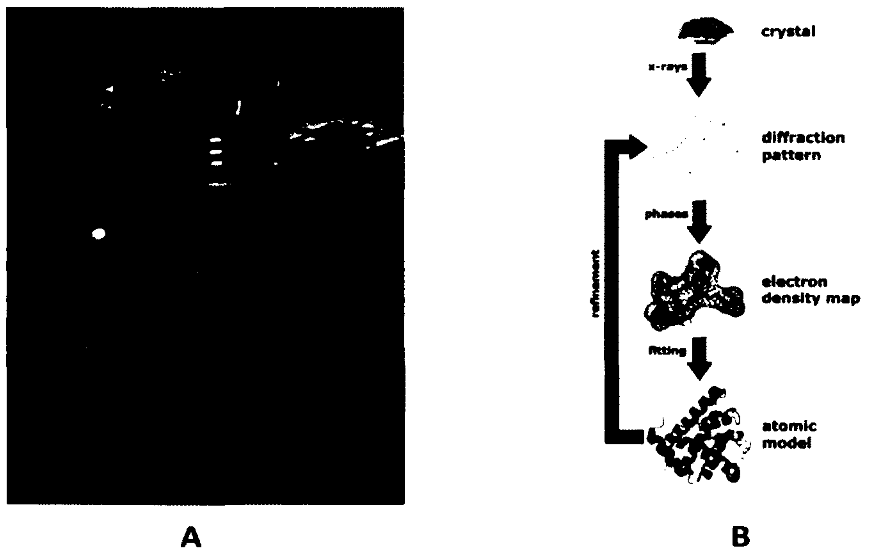


Figure 5. Two experimental methods to determine protein 3-D structure. (A) A 900MHz NMR instrument with a 21.2 T magnet at HWB-NMR, Birmingham, UK, Courtesy of Martin Saunders. (B) Workflow for solving the structure of a molecule by X-ray crystallography. Courtesy of Thomas Splettstoesser.

1.1.2 Protein Structure Prediction

Protein determination techniques are relatively expensive, time-consuming and not successful with all kind of proteins. Membrane protein is one example of those proteins unsuccessfully determined by experimental methods [23,24]. In particular, the success of X-ray crystallography is limited to the existence of suitable crystals of the protein, and unfortunately large proteins can't easily produce crystals. On the contrary, the sequencing of proteins is fast, simple and relatively less expensive. Due to rapid growing of the gap between number of known sequences and number of known 3-D structures determined, which is expected to keep growing because of genome projects worldwide, the need of computational methods that give some indication or prediction of protein structures/functions are become critically important [18]. The number of protein sequences available at the time of writing this dissertation is more than 31m[1] while number of structure determined and posted on Protein Data Bank[2] so far is less than 82,000. Furthermore, the sequence of amino acids, together with the physics of the intra-protein and the environment interactions, plays an important role in determination of protein structure [10,11,12,13,14]. Therefore, the prediction of protein native structure (tertiary structure) from its amino acids sequence (primary structure) has given more considerable attention [19]. It is becoming one of the most important goals in Bioinformatics and theoretical chemistry. The design of drugs and novel enzymes are two important examples on the applications of protein structure prediction in medicine.

Still, protein structure prediction is extremely a hard process for some proteins. The two most important difficulties in such a field are the calculation of a good energy function and finding the global minimum of this energy function. The search space of the problem in which prediction methods need to explore is astronomically large. In 1969, Cyrus Levinthal stated in what is known by "Levinthal's Paradox" that, due to the large number of degrees of freedom in the primary structure of the protein, the molecule has an astronomical number of possible conformations [25]. For example, if a protein of length 100 residues were to attain its correctly folded configuration by sequentially sampling all the possible conformations (3^{198} different conformations), it would require a time longer than the age of the universe to arrive at its correct native conformation. The huge search space could be

[1] The information is collected from the website http://www.uniprot.org/uniparc/
[2] From the website of Protein Data Bank www.pdb.org

pruned by comparative modeling or *ab initio* modeling. When the target primary structure is assumed to adopt a similar structure of another experimentally determined protein, comparative modeling would narrow the search space and guide the prediction method. Otherwise, *ab initio* modeling is used to predict the structure. The accuracy and performance of current prediction methods is assessed by CASP experiment (Critical Assessment of Techniques for Protein Structure Prediction) every two years [19,26,27,28,29].

1.1.2.1 *Ab Initio* Modeling

Ab initio modeling is a computational method aims at predicting and characterizing the structure/function of the protein using the information of primary structure, i.e., sequence of amino acids, as the only input. Due to the difficulty of the problem and the astronomical size of the search space, most of *ab initio* approaches use knowledge-based and physics-based potentials that may govern protein folding to guide the prediction process. The usage of such information is helpful to exploit important features regarding secondary structures, distant constraints, and conformational preferences taken from sequences. The majority of *ab initio* approaches focus on three aspects in this problem. First, suitable protein representation and corresponding protein conformation space in that representation. Second, an accurate energy function that is able to distinguish good conformations from bad ones and compatible with the representation. Third, efficient approach that is able to search the conformational search space to minimize the energy term [30]. Numerous sophisticated algorithms such as Monte Carlo, genetic algorithms, and molecular dynamics are used to search the conformational space.

ab initio modeling requires vast and intensive computations. For larger proteins, *ab initio* modeling needs an efficient algorithmically methods and very powerful computational resources such as supercomputers (i.e., IBM® Blue Gene or RIKEN MDGRAPE-3) or distributed systems (i.e., Stanford University project folding@home[3]). However, many representations have been proposed to simplify the protein structure than its full 3-D coordinates representation. The ultimate goal of such methods is to reduce the complexity of the model in which overcomes the sampling problems. These methods can be divided into two major classes: lattice and off-lattice models [31]. Lattice models are simple and the calculations of energy can be done fast and efficiently [32,33,34,35]. However, they are hard to represent geometric considerations (i.e., secondary structures) and most of these models exhibit various degrees of secondary structure bias which outweigh the advantages of this model [32,36,37,38]. Off-lattice models also tend to simplify the complexity with avoidance of the restrictions exhibited by lattice models. Some of these models limit the side chain to one center coordinate, or further to either C_β or C_α [39], or represent the side chain as spheres or ellipsoids [40], or recently to multiple geometrical shapes [41].

The most challenging aspect in *ab initio* prediction is the accurate energy function that distinguishes between good and bad conformations built through the process. The difference on energy between these conformations could be several kcal/mol in most cases which is equivalent to several atomic interactions [42,43]. The inaccuracy of force field may mislead the conformation sampling in which it may guide the energy minimization to a wrong conformation of global energy minimum that is totally different than the native one [23]. Thus, energy function should properly reflect the forces responsible for protein structure formation. Many energy functions have been developed; some based on the hydrophobic effect [44,45,46], pair potential interactions [35,47,48,49,50,51,52], or more complicated energy functions such as Rosetta energy function [53,54].

The fundamental idea of *ab initio* modeling was first proposed by a pioneering work of Anfinsen [10]. In his work, Anfinsen developed his thermodynamic hypothesis which states that the primary structure alone provides sufficient and adequate information for finding the native conformation of a protein. Later works have been

[3] The official website of folding@home project is http://folding.stanford.edu/English/Main

done by Bowie and Eisenberg [44] who assembled new tertiary structures using small fragments cut from other PDB proteins. Baker *et al.* [55] similarly developed ROSETTA which, in addition to its new developments [56,57], work well and made the use of fragments popular in the field. A good survey of what Rosetta can do can be found on [58]. One example of models based purely on knowledge is called TASSER which is proposed by Zhang and Skolnick [59], and I-TASSER, a newer version, developed by Wu *et al.* [60]. Oldziej *et al.* [39] developed UNRES which is a reduced physics-based model. Klepeis *et al.* [61,62], developed a novel four-stage *ab initio* approach called ASTRO-FOLD. TOUCHSTONE [63,64] is a threading-based algorithm of secondary and tertiary restraint prediction.

1.1.2.2 Comparative Modeling

In contrast to *ab initio* modeling, comparative modeling uses previously determined structures as templates. This modeling is seems to be effective due to the limited number of tertiary structures motifs available even though the number of proteins in the nature is credibly huge. Many proteins with good sequence similarity have similar functions and structures; when a query protein shares 30% sequence identity with a protein of known structure, comparative modeling can predict the structure to fairly good accuracy [65,66,67,68]. Most of comparative modeling consists of four steps [69,70]: First, finding a good template from already determined structures in protein data bank. Second, aligning query sequence with the template structure. Third, building the structural framework based on alignment by copying aligned regions. Fourth, filling up the gaps found on the framework. These four steps are actually performed in two steps. The first two steps are done in one step called threading (or fold recognition) [71,72]. Similarly, the last two steps are performed simultaneously in one step [30].

Although the Homology-based comparative modeling is the most successful methods for structure prediction to date [19,31,73], identifying the correct template and refine it closer to the native one is still an important condition. The appropriate template in the PDB is a crucial condition for the success of this modeling otherwise *ab initio* modeling should be used. On their recent study, Zhang and Skolnick [74] showed that high quality models could be built with average RMSD 2.25Å when the best template in the PDB used. Various algorithms have been developed for threading (i.e., identifying structure template) since its early invention in 1990s [71,72]. Examples of these techniques are: Profile-Profile Alignment (PPAs) [75,76,77,78], machine learning [79,80], Hidden Markov model [81,82], structural profile alignments [83] and many more. Furthermore, numerous refinement methods have been developed to put the templates closer to the native which is extremely hard question [30]. Examples of refinement techniques are: refinement techniques based on molecular dynamics [84], techniques based on AMBER [85] with Generalized Born (GB) [86] implicit salvation model potential [87], other techniques recently concluded based on knowledge of atomic contact potential [88], and many more. Some successful homology servers are available online such as spatial restraints-based server MODELLER [66,89,90], rigid-body assembly-base servers COMPOSER [91] and SWISS-MODEL [92], and other servers.

1.2 CRYO-ELECTRON MICROSCOPY

Cryo-electron Microscopy (CryoEM) is an advanced imaging technique that aims at visualizing and interpreting unstained nanostructures biological complexes such as viruses [93,94,95,96]. In contrast to X-ray Crystallography and NMR, CryoEM is able to visualize relatively larger molecules of atomic mass of 200 *kDa* or larger [97,98,99,100] (Fig. 7). CryoEM involves a process of freezing the sample in ethane slush to produce specimen's non-crystalline ice. These frozen specimens studied at very low temperature (i.e., below −238 °F) show a structure similar to the native state [97]. The advantage of freezing process of the biological sample is to view it without any distortion or artifacts such as redistribution of elements or removing away of substances and its ability to visualize different functional states [101,102]. A later averaging and processing of multiple images (i.e., thousands) leads to a relatively good resolution information (between 5 and 15Å). Unfortunately, at such a resolution, atoms positions are hard to be interpreted directly from the volumetric density map. However, Hong Zhou *et al.* [103] reported recently an image of a virus structure at a high resolution (3.3Å) which is enough to see atoms effectively. They used a single-particle CryoEM to report the structure of the primed, infectious subvirion particle of aquareo virus. The volumetric density map they have generated reveals side-chain densities of all types of amino acids (except glycine). It allows them to construct a full-atom model of the viral particle.

Since the first CryoEM volumetric density map (henceforth affectionately referred to as volumetric density maps) reported for hepatitis B virus in 1997 [104,105], many volumetric density maps of large protein complexes have been generated to low and/or intermediate resolution using CryoEM technique [95,103,105,106,107,108]. The Electron Microscopy Data Bank (EMDB) currently holds more than 1300 volumetric map entries in addition to more than 400 PDB entries of fitted coordinates (Fig. 6). The deposition rate of volumetric maps and fitted PDB models in 2008 and 2009 were around 150 and 40 per year [109].

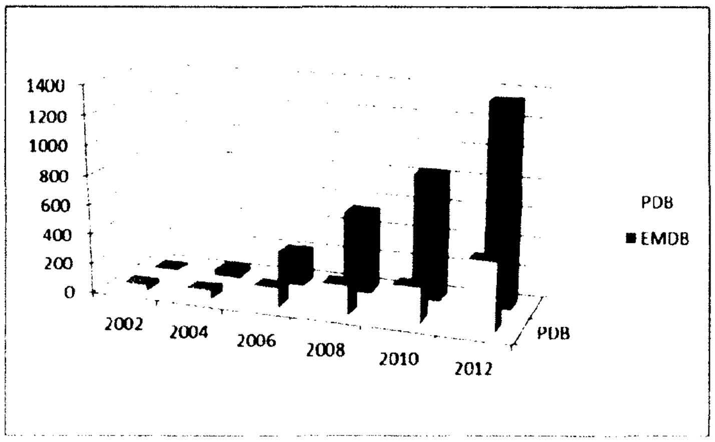


Figure 6. The growth of EMDB of CryoEM and fitted PDB models entries. The data is cumulative by year. Number of entries for year 2012 is through the month of May (May 9th).

At the medium resolution range such as 5-10Å, the volumetric density map is not resolved well enough to determine the atomic information of the protein. In contrast, protein structure prediction techniques (i.e., *ab initio* and comparative) have been proven to be capable of producing relatively good structural models for isolated proteins or domains [106]. Recent works has shown the ability of volumetric density maps to help in discriminating between models built by *ab initio* and/or comparative modeling and building final models as well [54,100,106,110,111,112,113]. Given an initial structural model obtained by either *ab initio* or comparative modeling, the volumetric density map is used to refine and fit the model structure to generate a high-resolution, all-atom protein models. Refinement process is done by heuristic methods, and a fitting scoring function measures how well the model fits into the volumetric density map to guide structure refinement process and identify mismatch regions between the model and the map.

The *de novo* structure prediction from the volumetric density maps is not the same as the general structure prediction problem. While the general structure prediction is to predict the atomic structure when the amino acid sequence of the protein is given, the *de novo* structure prediction from volumetric density maps is to predict the atomic structure when both the protein sequence and the volumetric density map are given. Instead of fitting those models already generated into volumetric map to refine the structure of the model at atomic-resolution, the prediction starts from volumetric density map to produce the atomic-structure of the protein.

At medium resolution (around 5 to 10Å), the location and orientation of the secondary structure such as helices and β-sheets can be computationally identified [114,115,116,117]. It is also possible to derive the β-sheets computationally [118]. Since the densities of the region connecting two consecutive secondary structures are not well resolved, the backbone structure of adjacent secondary structures elements is modeled in a separate step. Numerous recent methods have proven the effectiveness of predicting the structure of protein from low to medium resolution maps [52,119,120,121].

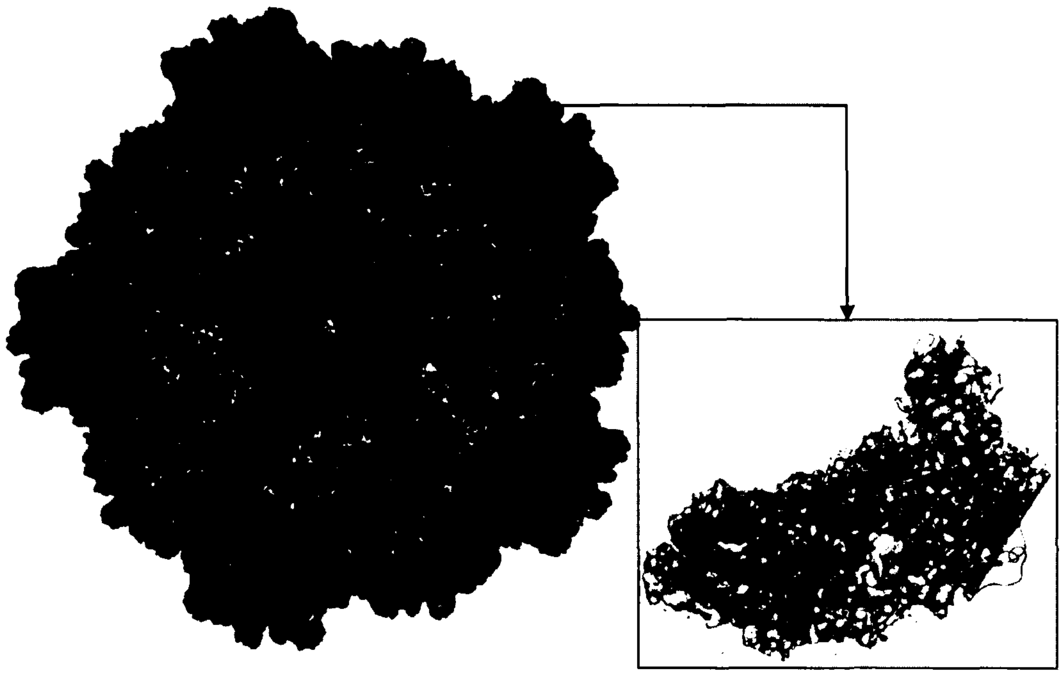


Figure 7. 3.88Å structure of Cytoplasmic Polyhedrosis virus (EMDB ID: 1508, PDB ID code: 3CNF) by CryoEM. Chimera [2] is used to visualize the CryoEM volume map.

1.3 SECONDARY STRUCTURE ELEMENTS EXTRACTION FROM CRYOEM

CryoEM is an experimental technique in which its capability to study large protein is advantageous on other experimental. Although the backbone of the complexes could not be derived directly from the volumetric map since the characteristics of amino acids are not well resolved at this resolution, secondary structure elements such as α-helices and β-sheets still can be detected. The location and orientation of these elements can be computationally identifies; α-helices can be detected as rods and β-sheets form plates areas.

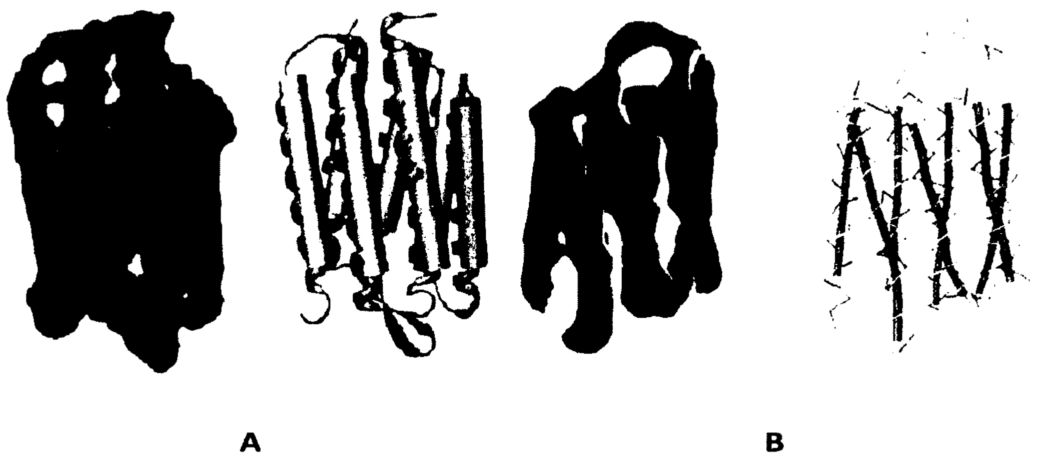


Figure 8. Two examples of helices extraction from volumetric density maps. (A) The prediction of helices of bacteriorhodopsin (PDB ID code: 1C3W) simulated at 8Å using EMAN suite [1] by HelixHunter, the helices are straight cylinders. (B) The prediction of helices of crystal structure of bacteriorhodopsin in the light-adapted state (PDB ID code is 1BM1) simulated at 10Å using EMAN suite [1] by HelixTracer, the helices are bent.

Many tools have been implemented to automatically extract secondary structure information from low to medium maps. HelixHunter [122] is a segmentation and feature extraction tool capable of identifying helix position, orientation and length using a five-dimensional cross-correlation search of a 3-D volumetric density map followed by feature extraction. EMatch [123] is a segmentation tool similar to HelixHunter. HelixTracer [115] differs from HelixHunter by the novel representation of α-helices. Helices are modeled as general cylinder-like shapes. Differently, HelixHunter represents the helices as straight cylinders with a 5Å diameter. In HelixTracer, the cylinder-like shapes is defined by a central axial line in which may not necessarily be a straight line. The central line is described by a set of control points represent the central axis of the predicted helix. This representation provides more flexibility and accuracy to the approaches that aim at building atomic-resolution structures. The advantage of HelixTracer over HelixHunter is on the segmentation method. The segmentation method used in HelixTracer relies less on threshold. Fig. 8 shows two examples of helices extraction from low resolution map using the two tools, HelixTracer and HelixHunter. Several other tools have been developed for automatic and manual secondary structure prediction from maps such as SSEhunter [116], Sheetminer [117], and Sheettracer [118]. HelixTracer and SSETracer are the tools being used through this study.

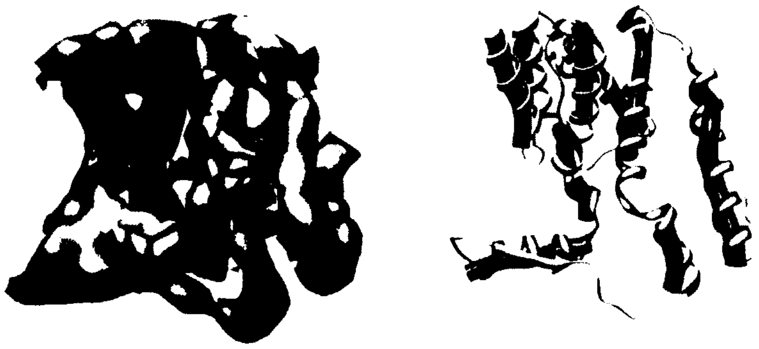


Figure 9. Helices extraction from the volumetric density map using SSETracer. Example of helices extracted from real CryoEM volume map (EMDB ID: 5100) at 6.8Å for the first 222 residues N-terminal of Scorpion Hemocyanin resting state (PDB ID: 3IXV). The extracted helices are bent.

Recently, a machine learning tool was developed in our group called SSETracer [124]. In SSETracer, some image processing concepts are used and the problem is translated to a multi-task learning problem and is then solved by using Support Vector

Machine (SVM). Each voxel in the volumetric map is classified into one of the three types of voxels: helix voxels, sheet voxels and background voxels. The feature extraction step in SSETracer characterizes each voxel based on the local geometrical features. Local gradient is often used to characterize the geometrical features and the local tensor used to define the local shape. Fig. 9 shows one example of helices extraction using SSETracer.

1.4 SECONDARY STRUCTURE ELEMENTS PREDICTION FROM THE PROTEIN SEQUENCE

Secondary structure prediction from protein sequences can be defined as the prediction of local secondary structure elements (i.e., positions on the sequence) of protein based on the given primary structure. The prediction of local secondary structure involves determination of the likelihood of the different segments on the sequence to different secondary structures types (i.e., helices, sheets, or turns). The success rate of the technique is determined by comparing its results to the results of Define Secondary Structure of Protein (DSSP) [125].

Secondary structure prediction has begun early in 1960s on single protein sequences and concentrated mainly on helix-like regions [11,12,13,14]. The accuracy of such methods was around 60-65% and they were relatively unable to predict beta sheets [126]. Recently, several prediction tools have been developed based on machine learning methods such as neural networks and support vector machines which are around 80% accurate in their prediction [127]. Numerous tools are now available for secondary structure prediction from protein sequences such as YASPIN [128], PHD [129] , JPRED [130], PSIPRED [131], Porter[132], and many other tools. Although the accuracy of secondary structure prediction tools is increasing, Dor *et al.* [127] stated that the theoretical upper accuracy of such methods is around 90%.

1.5 SUMMARY

Protein is one of essential organism parts on the planet. The word protein came from Greek word *prôtos* which means the most important. Proteins are the molecules carry out the vital function and make more than the half of dry weight in every cell. The function of proteins varies from acting as enzymes, cellular signaling (i.e., hemoglobin) and Molecular transport. Protein in nature folds into a unique and energetically favorable 3-D structure which is critical and unique to its biological function [4,5]. The unique conformation in which the protein folds into is called a native structure [9]. The sequence of amino acids build up the protein ultimately determine its native structure, in which corresponds to the favorable energy of the molecule [10].

The current methods of protein structure determination are not suitable for all kind of proteins such as membrane proteins. On the other hand, the sequencing of proteins is fast, simple and relatively less expensive. Thus, the gap between number of known sequences and determined structures is growing, which is expected to keep growing, and the need for computational methods to determine the structures of proteins is become critically important.

In contrast to traditional experimental techniques used to determine protein structures, CryoEM is a promising advanced image processing method for structure determination. Unlike X-ray crystallography, CryoEM is able to produce volumetric maps of proteins that are poorly soluble, large, and/or hard to crystallize. Unfortunately, at medium resolution, the volumetric maps generated by CryoEM are unable to determine the structure of protein at atomic-resolution. However, some features of the protein can be visually and computationally identified such as the location of secondary structure elements. The two protein structure prediction techniques (i.e., *ab initio* and comparative) have been proven to be capable of producing relatively good structural models for isolated proteins or domains [106].

CHAPTER 2

STATE OF THE ART

Relatively good structure models for isolated proteins or domains have been generated experimentally either by X-ray crystallography or NMR spectroscopy, or computationally either by *ab initio* or comparative modeling tools [100,106]. When a high-resolution atomic structure is available for small proteins or for a part of large proteins, fitting and refinements tools have shown the ability of deriving the atomic structure of a protein from CryoEM maps [54,100,110,111,112,113]. Given an initial structural model, the volumetric density map is used to refine and fit the model structure to generate a high-resolution, all-atom protein models. Refinement process uses a fitting scoring that measures how well the model fits into the volumetric map and identify mismatch regions between the model and the volumetric map.

2.1 RIGID FITTING

The techniques that attempt to fit the given atomic structure into the low or medium volumetric density map are called rigid fitting [100]. In rigid fitting techniques, keeping the structure being fit rigid, the ultimate goal is to minimize the fitting error with density map by finding the best correspondent position and orientation. Segger [133] is a latest example of rigid fitting techniques. It applies a watershed algorithm to segment the density map into regions corresponding to molecular components such as proteins and then fits them into the map based on the alignment of structures with segmented regions. A refinement on resulting alignments is accomplished locally to optimize the cross correlation score. Generally, in most rigid fitting techniques, a score function guides the exhaustive search over the search space. The most popular score function is the cross-correlation coefficient and its variants [122,134,135,136,137,138].

One difficulty arises when fit models generated for isolated proteins or domains by rigid techniques, is that the atomic structure is not the same as in the assembly. Different factors could cause this problem, for example when the structure is not solved to atomic resolution or existing of errors between the isolated protein and the assembly. Some errors caused by experimental methods [139] or some other errors caused by computational methods [65] such as the assignment of secondary structures to incorrect sequence regions which result in misplacing them in the 3-D space. To overcome this problem, a flexible fitting should be considered where the conformation of the structure being fit can be changed accordingly to improve the correspondence to density map.

2.2 FLEXIBLE FITTING

Several methods and tools recently developed to address the problem of rigid fitting. The early work of Volkmann *et al.* [140] has deployed flexibility fitting by breaking down the molecule to smaller rigid components and fitting them independently. Situs [141] uses a reduced representation for the molecule structure and the volumetric density map to undergo changes in the process of fitting. Another approach called vector quantization has been introduced that also uses a reduced representation [142,143]. NMFF [144], NORMA [145] and QEDM [146,147] use what is called normal mode analysis, where a linear combination of low-frequency normal mode is used to update the atomic structure while maximizing the correlation coefficient between the molecule structure and volumetric density map. A real space refinement method, RSRef [148], also proposed which optimizes the stereochemistry simultaneously while fitting the structure to the volumetric density map. Mod-EM [110] and Moulder-EM [111] combine comparative modeling of MODELLER [89] and structure refinement. The implementation of flexible fitting is done by fitting alternative comparative models according to the difference in sequence-structure alignments and different loops conformations. S-flexfit [149] and its substantial improvement [150], use the information posted in protein data bases like CATH to exploit the structural variability of protein domains within a given super family. Flex-EM [100] is a hierarchical approach that integrates rigid and flexible fitting of a component structure into the volumetric density map. The rigid fitting is done by a Monte Carlo search and then two refinement steps held based on simulated annealing and a scoring function. DEN approach [151] is a general geometry-based algorithm that combines constraints imposed by volumetric density maps and deformable elastic network (DEN). Jolley *et al.* [152] proposed a Monte Carlo simulation method with constrained geometric-based from FRODA [153]. In their work, Li and Frank [154] correlate the result from molecules dynamics simulations with volumetric density maps. MDFF [155] also performs molecular dynamics simulations to flexibly fit atomic structures into volumetric density maps. The simulations incorporate the volumetric density map as an external potential added to the molecular dynamics force field. Volumetric density maps have also been used to filter *ab initio* models [106]. Internal Coordinate Mechanics (ICM) [156] uses flexible fitting with *ab initio* prediction on transmembrane protein (relatively small proteins) to accomplish a high resolution protein structure refinement in simulated volumetric density maps. In a recent work of DiMaio *et al.* [113], unlike other methods which start with complete models, Rosetta structure prediction methodology [55,56,57] has been used to refine comparative models and low resolution C_α traces using volumetric density maps. A local measurement fitting score function has been used to identify incompatible regions for intensive rebuilding. The input comparative models in this method are relatively fit the map.

Inaccuracy in comparative modeling caused by target-template differences in the correctly aligned regions could not be addresses by flexible fitting. Moreover, On the absence of a high resolution structure correspondent to the map or part of it, which is mostly for all large proteins, it is impossible to fold the sequence to the map [119]. Therefore, for medium resolution volumetric maps, the location and orientation of the secondary structure elements such as helices and β-sheets can be computationally identified [115,116,117,122]. It is also possible to derive the β-strands computationally [118]. Since the densities of the region connecting two consecutive secondary structures are not well resolved, the backbone structure of adjacent secondary structures elements is modeled in a later step. Numerous recent methods have proven the effectiveness of predicting the structure of protein from medium resolution maps [52,119,120,121].

2.3 *DE NOVO* PREDICTION

The *de novo* prediction becomes very important when a high resolution structure is absent. A common step used is to extract the information of secondary structures from the map. Thus, many tools have been implemented to automatically extract secondary structure information from volumetric density maps. HelixHunter [122] and EMatch [133] are segmentation and feature extraction tools capable of identifying helix position, orientation and length. Both methods consider alpha-helices as continues, long, straight cylinders. HelixTracer [115] is a different segmentation and feature extraction tool in which it differs from both methods by the novel representation of α-helices, where helices are modeled as general cylinder-like shapes whereas HelixHunter represents the helices as straight cylinders with a 5Å diameter. The cylinder-like shape is defined by a central axial line called *spline*, which may not necessarily be a straight line, described by a set of control points represent the central axis of the predicted helix. Therefore, this representation provides more flexibility and accuracy to the approaches that aim at building atomic-resolution structures. The advantage of using HelixTracer over HelixHunter is on the segmentation method. The segmentation method used in HelixTracer is less threshold-dependent. HelixTracer has shown improved accuracy over the HelixHunter tool. Several other tools have been developed for automatic and manual secondary structure prediction from maps such as SSEhunter [116], Sheetminer [117], and Sheettracer[118]. In SSEhunter, a skeletonization algorithm has been used to identify secondary structures and suggest possible connections between them according to the density strength especially for short loops.

Contrary to the rigid and flexible fitting and refinement using density maps, *de novo* modeling of protein structures using volumetric density maps has started recently. An early effort of mapping between 3-D and 1-D structures using constraints obtained from density map is proposed [157]. The proposed approach builds a mapping library for helices between 3-D and 1-D structures by combining the information from protein secondary structure prediction and the information obtained from volumetric density maps. A parallel algorithm using dynamic distributed scheduling for load balancing is used to build a mapping tree. The algorithm was able to work on small to medium size proteins. Another work of Dal Palu *et al.* [158] is then proposed to work on larger proteins (up to proteins with 18 helices). A parallel constraint logic programming framework is developed to determine the 3-D structure of large complexes proteins. The framework is used to determine the association between parts of the primary sequence of the protein and α-helices extracted from volumetric density map of the large protein complex. Constraints include the length, relative position, and the connectivity of helices.

Wu *et al.* [120] proposed a procedure to determine the protein fold based on positions of secondary structure elements obtained from medium resolution volumetric density maps. The procedure they have proposed uses a knowledge-based geometry filter followed by an energetics-based evaluation uses a knowledge-based pair-wise potential function [159] to evaluate fold candidates. For each of the fold candidates passes the geometry screening filter, a coarse-grained model for secondary structure elements has been built, where each amino acid is represented by one atom C_α. Loop regions were built using an off-lattice Monte Carlo procedure [160,161]. Once the full C_α trace structure is built, a global optimization included the genetic algorithm used to rotate helices, and Monte Carlo relaxation was applied for loop regions, and Monte Carlo optimization for β-sheets regions. One drawback of their work was the time required to accomplish the search process. The time they have reported to find the solution for a small protein of three helices was more than 100 minutes. The time required to generate the entire set of candidates to evaluate using the knowledge-based geometry filter is protein size-dependent as well.

The correspondence between the set of helices on the volumetric density map and the helices on the sequence has been studies by Abeysinghe *et al.* [162]. They represented the problem as a subgraph-isomorphism between the primary chain (1-D) and the density map (3-D). The two shapes were modeled as attributed relational graphs. A constrained inexact graph matching problem has been solved by a heuristic function. On this work, the structure of the protein is not built; the only question answered is the correspondence between the two shapes based on geometrical features. SSEHunter [116] in addition to a skeletonization approach [163] to trace the volumetric density map were used to extract the geometric features from the volumetric density map. On some of proteins in the small data set, their geometrical approach failed to find the correct correspondence for some of proteins before the user is allowed to interactively add some constraints. The problem of wrong correspondence was due to the insufficient information extracted from the volumetric density map. On his latest work, Abeysinghe *et al.* [121] extended the method to be able to work on sheets and build the C_α traces for the proteins. The method suffers some limitations. The most important limitation is the computational cost. Thus, due to memory limitation the method was unable to work on proteins with more than 25 helices without a large number of specified user constraints. Moreover, the method showed a sensitivity to one type of β-sheet mis-prediction which is proven to be a challenge for current secondary structure extraction techniques [119]. Furthermore, on low resolution density, the success rates of the method are low due to the bad quality of geometry skeleton obtained.

Lu *et al.* [54] have incorporated the constraints extracted from low resolution volumetric density maps in *ab initio* proteins structure prediction. A two-stage approach to predict the backbone of a protein from a low resolution map is incorporated with Rosetta [55]. In the first stage, a small set of possible topologies will be predicted for the helical topology. Rosetta is used to produce 1000 of *ab initio* models purely based on sequence. The produced structure then screened for agreement with the helix topology derived from volumetric map [54]. In the second stage, the proposed approach searches for the conformations satisfying the constraints derived from the first stage. A modified Rosetta energy function is used to carry out the search process. The proposed approach does not use the entire volumetric density map at the stage of simulation. The volumetric density map is used for ranking the near native models generated from Rosetta package.

Lindert *et al.* [119,164] proposed a *de novo* folding approach of α-helical proteins guided by medium resolution volumetric density maps, EM-Fold. EM-Fold uses a Monte Carlo sampling method to build and refine protein fold into medium volumetric density maps. The first early step, like other tools, is to identify helices on the volumetric density map where EM-Fold considers helices as density rods. The predicted helices from the sequence then placed on the density rods using a Monte Carlo assembly algorithm. A Monte Carlo refinement process then used to improve the placement of structural α-helices. On a Later step, the loop and side chains are added by Rosetta's iterative side-chain repacking and backbone reconstruction protocols in which to generate a model at atomic resolution [113].

CHAPTER 3
METHODOLOGY OVERVIEW

The method proposed aims at predicting the atomic-resolution of a protein using its CryoEM volumetric density maps. The *de novo* structure prediction from the volumetric density maps differs from the general structure prediction problem. While the general structure prediction is to predict the atomic structure when the amino acid sequence of the protein is given, the *de novo* structure prediction from CryoEM maps is to predict the 3-D position of every atom when both the protein sequence and the volumetric density map are given.

We have decomposed the entire process into smaller components. Fig. 10 shows the three main components that form the entire system of prediction. Each component consists of smaller processes that produce intermediate structures or parts of structures. The first component is the preprocessing and data preparation (upper component in Fig. 10). The second component is the intensive component in which the prediction of secondary structure elements at atomic-resolution takes place. The third component (bottom in Fig. 10) is where the post processing and loop region modeling accomplished.

3.1 PREPROCESSING AND DATA PREPERATION

The first step in the method is to prepare the two inputs of the system, the secondary structure elements from the sequence (SSE-S) and positions of secondary structure elements in the 3-D space (SSE-D). SSE-S can be obtained from one of secondary structure tools [128,129,130,131]. The accuracy of secondary structure prediction plays a main role in the later prediction process. The poor predicted secondary structure may lead to a wrong prediction of atomic-resolution structure for the entire protein. Thus, to overcome this problem and decrease its negative impact, a consensus prediction over some tools also is possible. However, the true position of the SSEs-S is used in this work.

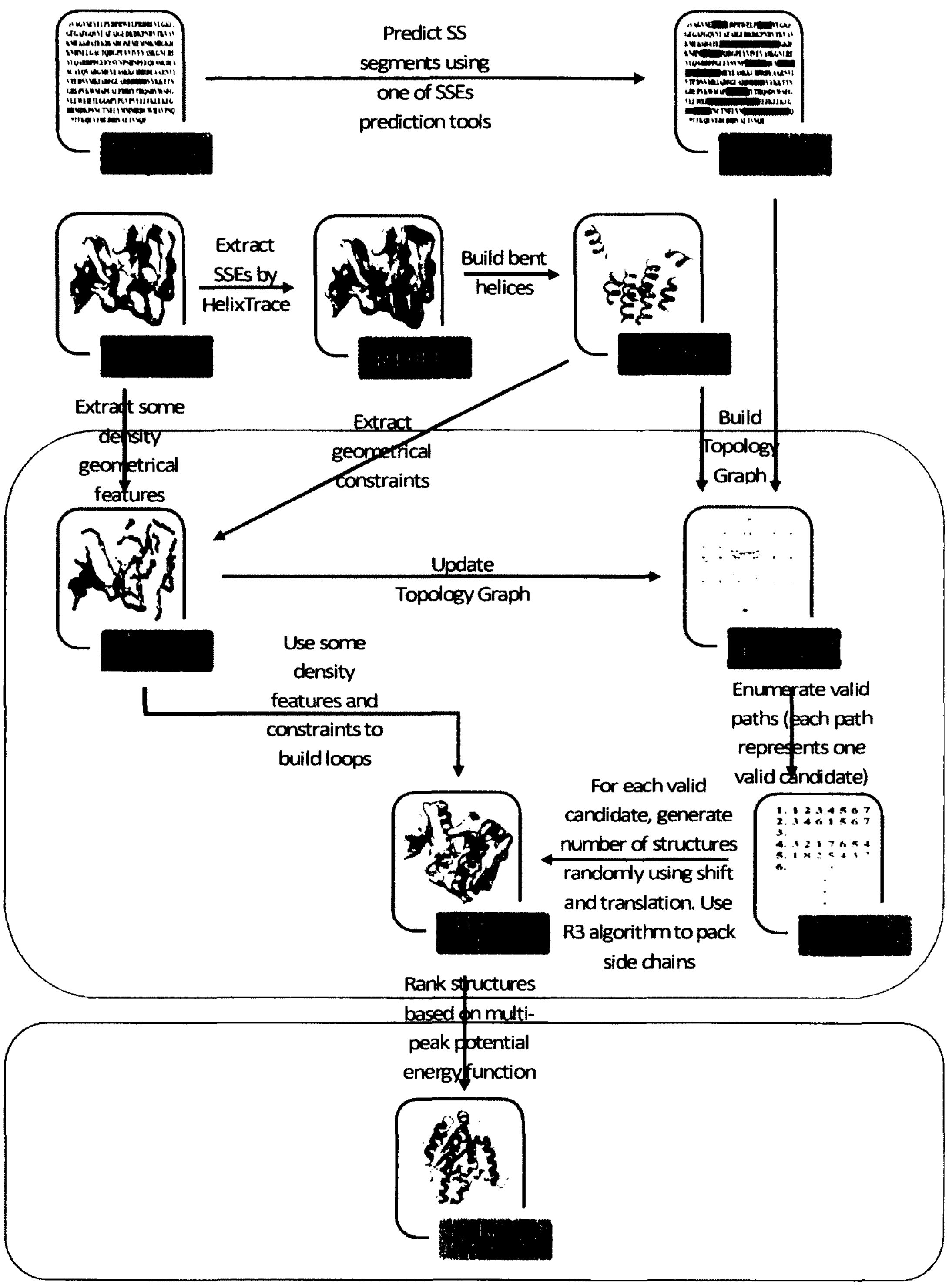

Predict SS segments using one of SSEs prediction tools
Extract SSEs by HelixTrace
Build bent helices
Extract some density geometrical features
Extract geometrical constraints
Build Topology Graph
Update Topology Graph
Use some density features and constraints to build loops
Enumerate valid paths (each path represents one valid candidate)
For each valid candidate, generate number of structures randomly using shift and translation. Use R3 algorithm to pack side chains
Rank structures based on multi-peak potential energy function

Figure 10. Overview of the proposed methodology.

The second input of the system, the positions of SSE-D, can be obtained from one of several tools available [128,129,130,132]. HelixTracer [115] and SSETracer [124] were used in this method as the tools to extract positions of secondary structure elements from volumetric density maps. The extracted elements are represented by a set of points of its axis, *spline*. The extracted information of secondary structures only contains their positions. The order of the SSEs-D along the protein sequence is not known. This question will be addressed in the next component process. The backbone atomic structure of each extracted SSE-D is built separately in this step. No connections or order is assumed. Some geometrical features also extracted from atomic-resolution structures built, such as the distances between every two ends of these structures.

3.2 TOPOLOGY PROBLEM

The second component is the major component. In this component we address the problem of ordering and direction of the SSE-D that can be obtained from HelixTracer and/or SSETracer along the SSE-S on the sequence that can be obtained from secondary structure prediction tools. Furthermore, in this component not only the order and direction of sticks will be answered, also an atomic-resolution for these sticks will be built. To address this problem, every pair of sticks and sequence elements has been converted to a node in a layered graph. The connections between these nodes are governed by the geometrical features collected from the previous component. The layered graph was built to enumerate all valid candidates for further evaluation and modeling. More details about this component will be discussed in Chapter 4.

3.3 LOOP MODELING

Loop modeling and prediction is as hard as predicting the entire structure of the protein. Most of problems encountered in protein prediction also encountered in loop modeling as well. The only difference is the size of the problem in both cases [165]. In loop modeling, as in protein modeling, the energy function used to test of native-like conformations is the essential discriminator between the qualities of different conformations. Moreover, to be able to predict a protein structure, your loop models predicted should be native-like and stable.

Recently, *ab initio* loop modeling has shown a considerable accuracy of predicting loops of length up to 12 residues [166,167,168,169,170]. In *ab initio* methods, loops are generated from random conformations. In contrast, Comparative modeling depends on databases to extract loop structures has also shown important progress [171]. A recent multi-method comparative approach was used to build loops of lengths up to 25 residues [172]. Most of loop modeling approaches that sample large number of conformations add side chains in later steps. Some of loop modeling approaches and tools are based on Random Tweak [173,174], such as LOOPY [166]. Random tweak is based on the computation of the Jacobian matrix of the first derivatives of error distance with respect to the torsion angles. It uses Lagrange multipliers to minimize the change of torsion angles while satisfying the constraints. In each iteration, the adjustment applies to all phi (ϕ) and psi (ψ) angles of the loop. It does not have the flexibility of imposing additional constraint on specific residues. Another drawback is that it is not free from mathematical singularities. LOOPY [166] avoids structure collision while applying loop closure. Loop closure problem, which we will discuss in Chapter 5, aims at filling a gap in between two fixed portions of structure (i.e., loop modeling). A self-organizing algorithm is used to generate clash-free loops of lengths between four and 12 residues [175]. The algorithm starts from random initial atomic coordinates followed by fast geometric matching of the conformationally rigid components of the constituent amino acids. RAPPER [176] avoid structure collision while sampling conformations using a predefined set of backbone torsion angles (i.e., ϕ and ψ). Rosetta [177] uses a Monte Carlo procedure to assemble the set of fragments sampled from database. PLOP [178] builds its fragments from a build-up procedure from N-terminal and C-terminal stems that meet in the middle. Sub-angstrom methods were introduced recently for relatively long loops (i.e., 12 residues). Mandell *et al.* [179] developed a kinematic closure that produces loops with sub-angstrom resolution. Li *et al.* [180,181] introduced a computational sampling approach to obtain reasonable loop backbone models. The sampling method is called Pareto Optimal Sampling (POS) which is derived from Pareto Optimality. The method uses multiple scoring functions to model the loop. POS efficiently tolerate insensitivity and inaccuracy in individual scoring function. Similarly, the method produces loops with sub-angstrom resolution. Each one of loop modeling approaches uses a score function to pick the most stable and energetically favorable conformations among the generated fragments.

3.4 CONTRIBUTION

In the course of the proposed system, the following original contributions have been made in this dissertation:

1. Efficient and fast graph-based approach to solve the problem of secondary structures and sequence elements assignment: we have developed a framework to perform the *de novo* structure prediction for secondary structures. On this framework, we have converted the problem of enumerating all the topological space. Instead of enumerating all possible topologies for validity evaluation in a separate step, we have built a graph in which only valid topologies to be enumerated. Two papers have been published on this field [182,183]. The proposed algorithm reduced the factorial term in the problem, $N!\,2^N$, to a polynomial term. When the number of sequence segments and secondary structure sticks detected on volumetric map are equal, the complexity of the algorithm to build the graph and find the best topology is $O(N^2 2^N)$. Moreover, we have introduced an alternative version of Yen's algorithm to enumerate the K-best possible topologies from the graph. The complexity of the proposed algorithm after building the graph is $O(KN^2)$. One paper is being written to be published on this field of study.
2. The graph representation proposed for the topology problem is general. The graph representation as well as the K-shortest paths algorithm developed can

be applied to many similar problems. For example it can be applied to inexact graph matching problems and to any assignment-like problem in the presence of constraints.

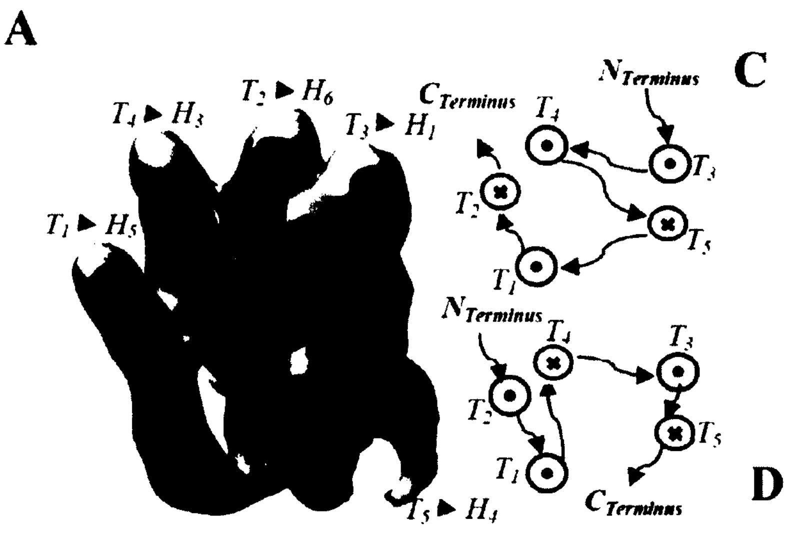
3. A novel approach to build a structure on the top of a skeleton trace: Recall that, in *HelixTracer* or *SSETracer*, the helix is defined by a central axial line called *spline*, which may not necessarily be a straight line, described by a set of control points represent axis of the predicted helix. One challenge is to build bent helix on the top of this axis. In a previous work [183], we have built straight helices using the first and last points in this axis. Currently, we are using a Cyclic Coordinate Descent method (CCD) (to be discussed in Chapter 5) to build such curved helices. CCD is an inverse kinematics approach to solve the problem of moving a robotic gripper to a specific position by changing joint angles and segment lengths.
4. A fast, accurate, and convergence-independent method for loop closure problem: we have addressed the problem of loop closure. We have proposed a FBCCD approach which is fast, accurate, and convergence-independent. More details about this approach can be found on Chapter 5. Two papers have been published on this work [184,185].
5. A system of three components for *de novo* prediction of protein structures instead of a number of components to solve problems separately: we now have an entire system for *de novo* structure prediction. One paper is being written for this work.
6. Parallel scheme for *de novo* structure prediction: we have built a simple dynamic master/slave parallel scheme to model the proteins. One of the future works is to parallelize the process of enumerating topologies from layered graph. Parallelizing layered graph will allow us to work on larger and/or multi-domain proteins. One paper was published in this field [183].

CHAPTER 4

TOPOLOGY PROBLEM

Prediction of secondary structure elements on the sequence, prediction of the atomic positions of secondary structures are essential steps in our system. These steps are separated and accomplished in the preprocessing component. The problem being addressed is the topology determination of the protein. The topology problem can be simply defined as the correspondence between the secondary structure elements on sequence (SSE-S) and those elements on 3-D (SSE-D) (i.e., order and direction). The problem can be represented and simplified by building a layered graph where each node represents one assignment between one element from SSE-S to one element from SSE-D.

The topology determination in this process is to identify the order and the direction of the SSE-D (i.e., sticks in Fig. 11) detected from the volumetric density map. The computational tools, such as HelixTracer and SSEhunter, provide the location of the SSEs in the volumetric density map, but they do not provide the topology information about the SSEs. For example, the true order is the one in which the protein sequence starts from stick T_3 then goes to T_4 (Fig. 11*C* and 11*A*). At the medium resolution, the order of the sticks is often ambiguous in the volumetric density map and has to be determined. The SSE-S provides an estimated location of the SSEs on the protein sequence. Therefore, the topology determination problem is also an assignment problem between the SSE-D and the SSE-S [51,52,112,183]. In the ideal case when there are N helices and M β-strands in the SSE-D and the SSE-S respectively, the total number of possible topologies is $N!\,2^N M!\,2^M$. This number is based on the fact that there are $N!$ different orders for the N helices and two directions to assign for each helix. The same rule is applied for β-strands. Due to the large topological space, topology modeling often involves two steps: the generation of a subset of the topologies that are more likely to include the true topology and the evaluation of each such topology by building the corresponding structures. The second step is computationally intensive and often involves the evaluation of the energy of the resulting structure [52,183,186].


A
$T_4 \blacktriangleright H_3$
$T_2 \blacktriangleright H_6$
$T_3 \blacktriangleright H_1$
$T_1 \blacktriangleright H_5$
$T_5 \blacktriangleright H_4$
$C_{Terminus}$
$N_{Terminus}$
T_4
T_3
T_2
T_5
T_1
C
$N_{Terminus}$
T_4
T_3
T_2
T_5
T_1
$C_{Terminus}$
D

B

[illegible] **SRKLMLDARENLKLLDRMNR** [illegible]
[illegible] **EMVEG** [illegible] **KDQAFPVLYEMLQQSFNLFYTEH** [illegible]
[illegible] **TLLEQLCTGLQQQLDHLDT** [illegible]
[illegible] **DPIVTVKKYFQGIYDYLQE** [illegible] **CAWEIVRVE**
MMRALTVSTTLQKRL [illegible]

Figure 11. Helical sticks and topologies of ovine interferon-tau (1B5L) protein. (A) The density map (grey) was simulated to 10Å resolution using protein 1B5L from the PDB. The helical SSE-D (T_1 to T_5) were detected using HelixTracer and viewed by Chimera. (B) The helices SSE-S are highlighted (bold) on the proteins sequence. Two alternative topologies are shown as diagrams in (C) correct and (D) wrong topology, in which the N to C direction for the loop (arrow) and for the stick (cross and dot) is labeled. The true assignment is labeled on the stick with H_1 being the first helical SSE-S.

4.1 TOPOLOGY AND THE ASSIGNMENT PROBLEM

The sticks in the 3-D space are detected using image processing tools such as HelixTracer. They are not associated with the protein sequence until we assign a segment of protein sequence for each of the sticks. The assignment problem can be described as the following. Let $H_1, H_2, \dots, H_M$ be a linear order of the segments located on the protein sequence, where H_1 is the first sequence segment and the H_M is the last sequence segment for the possible locations of helices (Fig. 11*B*). Let the set of sticks be $T_1, T_2, \dots, T_N$, and each stick T_i is labeled with the two end points S_i and S_i' where $i = 1, \dots, N$. Each topology is the result of a possible assignment between the sequence of $H_1, H_2, \dots, H_M$ to the set of sticks $T_1, T_2, \dots, T_N$. The topology refers to the linear order of the sticks and the directionality of each stick. For example, a topology (S_3, S_1', S_2') refers to an assignment of the three SSE-D (T_3, T_1, T_2) to three SSE-S (H_1, H_2, H_3).

The goal is to reduce the large topological space quickly to a small subset of possible topologies without the use of energy evaluation. This is likely to achieve because most of topologies might be invalid due to the geometrical constraints. The goal is to include the true topology in such a subset. One approach, the naïve approach, is to generate all the topologies in the entire solution space and then evaluate each one. This approach will be covered in Section 4.3. Another approach is to translate the problem into a graph matching problem aiming to find the optimal match of the two attributed related graphs. This approach was introduced by Abeysinghe *et al.* [162,187]. One graph was created from the SSE-S. The other graph was created for SSE-D. Each of them describes the connection relationship among the SSEs. However, this method requires that the true link between the SSE-D to be detected in the volumetric density map. In reality the true link may be missed due to the quality of the volumetric density map. Lindert *et al.* [186] used Monte Carlo method to generate the topologies in which the helical sticks were assembled in different orders and directions. This approach allows the sampling of the large topological space, particularly with the consideration of the errors in the data.

However, the random nature of the Monte Carlo method does not guarantee to find all the top ranked topologies. We propose a general framework to the topology determination problem using a weighted directed graph. The topology determination problem is then represented as a layered graph problem. In the layered graph (called topology graph) the topologies are represented as paths from the start node to the end node. The best topology is then represented by the shortest path in this graph. The shortest path can then be found using a depth first search, best first search or any other search algorithm. In this dissertation we will examine the approach of finding topologies from the graph using a depth first search in Section 4.4. More importantly, we will illustrate our dynamic programming algorithm to find the topology with the minimum cost in Section 4.5. This algorithm is, as expected, significantly faster than the naïve method and the depth first method. It allows us to find the topology with the minimum cost for large proteins. We also developed a method to enumerate the topologies with the top-K cost using the layered graph. This is the first dynamic programming approach to rank the topologies of the SSEs. The complexity of the proposed dynamic programming method is $O((D+1)^2 N^2 2^N)$, where D is the difference between number of SSE-S and number of SSE-D (M-N).

4.2 CONSTRAINTS

We use the distance and length constrains to screen for valid topologies. The distance constraint requires that the number of amino acids (in the loop) between two consecutive helical SSEs-S to be large enough to make the connection between two consecutive helical SSEs-D. The distance constraint was used to evaluate any two assigned pairs $< H_i, T_j >$ and $< H_k, H_l >$ where $i < k\ i, k = 1, \dots, M$ and $j \neq l, j, l = 1, \dots, N$. In the topology example $(S_3, S_1', S_2'), < H_2, S_1' >$ and $< H_3, S_2' >$ violate the distance constraint (Fig. 12). We made the judgment as the following. Given any two assigned pairs, we count the number of amino acids on the sequence that is expected to connect the two SSEs-D. Since the typical distance between the C_α atoms of the two amino acids is 3.8Å. The maximum expected distance for the loop that is in between two SSEs-S can be calculated. If the maximum expected loop distance is less than the corresponding distance measured in the 3-D space, a violation occurs. In order to simulate the inaccuracy from the actual secondary structure prediction, two times the allowable shift were added in this estimation. In this topology, there are only five amino acids on the loop between H_2 and H_3 (Fig. 12), therefore, the maximum expected loop distance span is about (5+2*2)*3.8Å=34.2Å, if two shift positions were allowed. Therefore five amino acids are not enough to make a connection from the point S_1 to the point S_2' which is 51Å (Fig. 12).

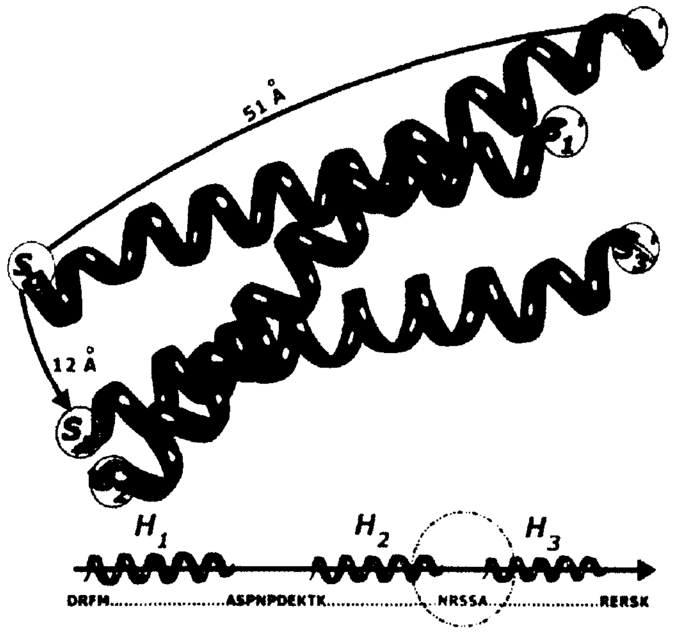


Figure 12. Secondary structure elements for the N-terminal domain of syntaxin (PDB ID code: 1BR0). The three SSEs-D and the three SSEs-S are shown. The amino acids in the loop between H_2 and H_3 were highlighted in a disk. The measured distance between point S_1 and S_2' was indicated.

In addition to the distance constraint, we also applied a length constraint to eliminate the situation when a SSE-D is assigned to a SSE-S when their length difference is too large. Based on experience, we required that 40% of the SSE-S length to be within the length of the SSE-D and 40% of the SSE-D length to be within the length of the SSE-S.

4.3 THE NAÏVE APPROACH: TOPOLOGY SCREENING

The early approach that we have used is the naïve approach [183]. In the naïve approach, we have generated all the topologies in the entire solution space and then eliminated the ones that are impossible or less likely to be true based on geometrical constraints illustrated in Section 4.2.

4.3.1 The Method

The input of the method includes two sources of information: the low resolution protein volumetric density map and the sequence of the protein. The volumetric density map was simulated from the native 3-D structure of the protein in the PDB to 10Å resolution using EMAN [1]. HelixTracer was used to detect the location of helical SSEs-D [115]. In order to map the SSEs-D to their corresponding SSEs-S, we generated all the $\binom{M}{N}N!\,2^N$ possible topologies of the SSEs-D, where M is the number of SSEs-S in the native protein and N is the number of helical SSEs-D [52]. To eliminate the unlikely topologies in the early stage of the process, the combination of geometrical constraints illustrated in Section 4.2 was conducted. For each possible backbone of the SSE-D stick, the distance, d, between the last C atom of a stick and the first N atom of the next stick is measured. We eliminated the topologies that satisfy $d > 3.8\,(n_{loop} + 2s)$ where n_{loop} is the number of amino acids on the loop connecting the two adjacent sticks and s is the maximum number of shift allowed in the sequence assignment. We used $s = 2$ for this work. This empirical rule was used to overcome the problem of errors (i.e., wrong position of the start and/or end amino acid) in the prediction of SSEs-S. The other rule we used to eliminate the bad topologies is to require an equivalent length detected from the stick and that from the sequence segment. A stick has an equivalent length as a sequence segment if their length difference is within 60% of the length of the stick. The length of a helix stick is the number of amino acids it contains estimated using a rise of 1.5Å per amino acid.

Since the secondary structures such as helices has more or less consistent backbone torsion angles, we generated a pool of possible backbone structures that share the same central helix axis. For each of the sticks, an initial backbone was constructed using the torsion angles ($\varphi = -60°, \psi = -50°$) to simulate a perfectly straight helix. We then generated an alternative structure by applying a rotation, θ, and a translation, t, to the initial backbone of the SSE-D around its helix axis. Since each topology determines an assignment between the SSEs-S and SSEs-D, it is possible to assemble the side chains to the backbone. To simulate the inaccuracy of the secondary structure prediction, we introduced a shift, s, for each sequence segment. $S = p_p - p_t$ where p_p is the index of the center amino acid of the predicted SSE-S and p_t the index of the center amino acid of the helix SSE-S in the native structure. Thus, for each topology, we constructed a pool of backbones, each of which can be represented by a set of parameters $(S_1, \theta_1, t_1), (S_2, \theta_2, t_2), \ldots, (S_N, \theta_N, t_N)$, when there are N SSEs-D in the volumetric density map. For each backbone constructed, we added the side chains based on a specific topology. The side chains were added using the rotamer library and the algorithm of R3 [188,189,190]. We developed a parallel simulated annealing process to optimize the all-atom structure for the sticks using a multi-well energy function previously developed [51]. A set of 55 processors were used in a master-slave dynamic load-balance implementation to perform the optimization. The master processor sends topology variables (the orders and the directions) and the set of parameters (S_i, θ_i) to each available processor. Each slave processor executes a simulated annealing process on the given topology. Fig. 13 depicts the method of the Naïve approach.

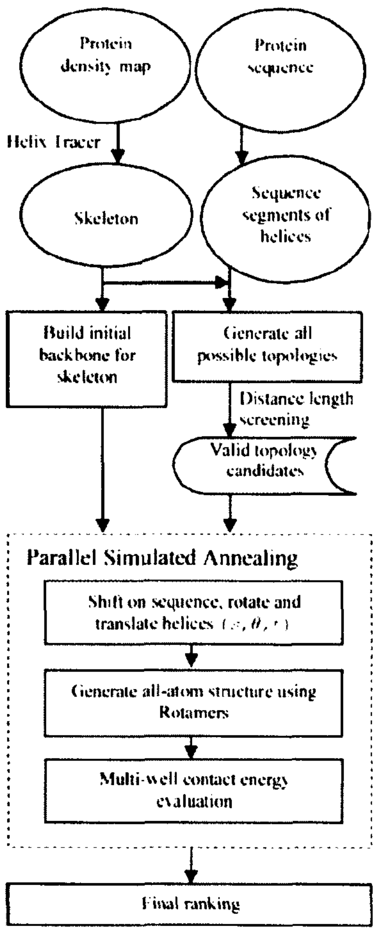

Protein density map
Protein sequence
Helix Tracer
Skeleton
Sequence segments of helices
Build initial backbone for skeleton
Generate all possible topologies
Distance length screening
Valid topology candidates
Parallel Simulated Annealing
Shift on sequence, rotate and translate helices (z, θ, r)
Generate all-atom structure using Rotamers
Multi-well contact energy evaluation
Final ranking

Figure 13. Secondary structure prediction using the naïve approach.

4.3.2 Results

Given the protein density map at 5-10Å resolution and its primary structure, our method generates a list of possible 3-D structures for the helices of the protein. Fig. 14 shows an example of the predicted structure for the helical SSEs-D detected from the 10Å resolution protein volumetric density map. In this case, HelixTracer detected five of the six helices in this protein (1B5L, Table 1, the 34th protein). In theory, there are totally $\binom{6}{5}5!\,2^5$ = 23,040 different topologies, with each one representing a specific order and direction of the SSEs-D sticks [51,52]. After distance and length screening there were 438 valid topologies (Table 1, row 34, column 7). For each valid topology, 500 structures were generated using simulated annealing to sample the freedom from $(S_1, \theta_1), (S_2, \theta_2), \ldots, (S_5, \theta_5)$. The translation along the helix axis was set to zero for simplicity. The predicted structures were sorted by the effective multi-well contact energy formed by the helices [51]. The highest ranked structure with the correct topology (red in Fig. 14) is the 759th out of 219,000 structures (Table 1, row 34, column 8). It has a backbone Root-Mean-Square-Deviation (RMSD) of 5.44Å from the native protein. The RMSD was calculated for the helix portion of the chain that was constructed by our program. Note that this method predicts the helix portion of the chain without building the loops; the predicted structure does not have the information about the loops. The two adjacent helices were simply connected with a straight line between the last C atom of the first helix and the first N atom of the next helix (Fig. 14). The amino acid names were shown for one of the five helices (Fig. 14). For viewing clarity, certain constructed side chains were shown for that helix. It can be seen that the sequence segment, the direction of the assignment are correct for this helix when the predicted helix is compared to its native peer. We noticed that the perfect helix model has introduced error in the predicted structure, since helices are often not perfectly straight and contain slightly different dihedral angles (data not shown).

To test the performance of our method, we generated 35 density maps at 10Å resolution [1] using the native structures from the PDB. The proteins were randomly selected among the proteins that have three to seven helices (Table 1, column 4). The total number of possible topologies $\binom{M}{N} N!\, 2^N$ is shown in the 6th column. It appears that the distance and the length screening are generally effective to reduce the number of topologies (column 6 and 7). However, this reduction is protein dependent. For some proteins, it only reduces less than 10% of the topologies (1DXS, row 17), and for other proteins, it reduces more than 80% (1JW2, 20th row). This is expected since the distance screening can only reduce the topologies in which the loops appear to be short in amino acid sequence but long in the density map and not the other way around. The structures were ranked by the contact energy formed by the constructed helices and not including the loops. The highest rank of the structure that has the correct topology is listed in column 9 (Table 1). Our previous study has shown that if the backbone coordinates are fixed, the correct topology can generally be located at the top 25% of the list that is ranked by the effective contact energy [51]. In this study we relaxed the requirement of fixing the backbone coordinates and built the possible backbones from the central helical axis. This involves the sampling of the rotation and translation freedom about the helix axis. Our simulated annealing test in this work suggests that a near-native helical structure can be found within the top 1% of the structures generated (Table 1, column 11).

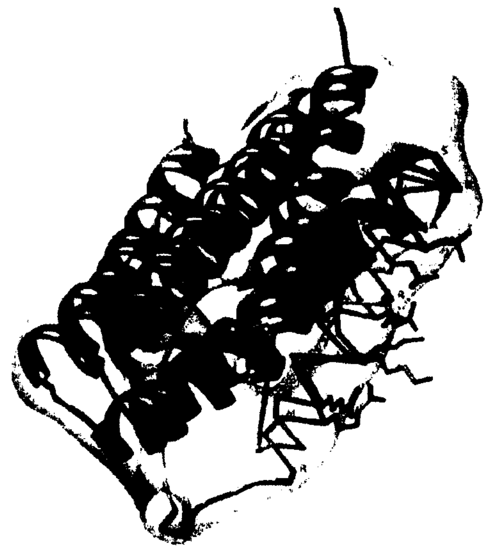


Figure 14. The highest ranked structure with the correct topology for 1B5L (PDB ID code). The native structure (grey ribbon) and the predicted structure (red ribbon) were superimposed on the protein density map. In the predicted structure, the connection between the two helices is simply drawn as a straight line that is smoothed by the ribbon representation. The amino acid labels and the side chains are shown for one of the five helices. The dotted line (grey) represents the missing loop in the native structure.

Table 1. Structure prediction of the helical SSEs-D sticks using the naïve

No	ID	#AA[a]	#hlces[b]	#sticks[c]	#Possible Topologies[d]	#Valid Topologies[e]	#Generated structures[f]	Rank[g]	RMSD[h]	Prct[i]
1	1DP3	55	3	3	48	6	3,000	10	4.78	0.33%
2	1A2T	149	3	3	48	32	16,000	15	3.72	0.09%
3	1AIL	73	3	3	48	16	8,000	3	3.96	0.04%
4	1BUU	168	3	3	48	16	8,000	27	4.67	0.34%
5	1IRE	81	3	3	48	18	9,000	10	4.55	0.11%
6	1BRO	120	3	3	48	4	2,000	4	11.17	0.20%
7	1B67	68	3	3	48	14	7,000	17	4.03	0.24%
8	1AYI	87	4	3	192	48	24,000	93	3.8	0.39%
9	1BEA	127	4	3	192	160	80,000	572	4.75	0.72%
10	1GXG	85	4	3	192	40	20,000	12	2.8	0.06%
11	1NO1	126	4	3	192	120	60,000	42	4.24	0.07%
12	2EZH	75	4	3	192	104	52,000	47	4.2	0.09%
13	1A3C	181	4	3	192	48	24,000	110	3.57	0.46%
14	1A32	88	4	3	192	40	20,000	9	3.69	0.05%
15	1PZ4	116	4	3	192	72	36,000	147	7.3	0.41%
16	1LIH	164	4	3	192	124	62,000	114	5.48	0.18%
17	1DXS	80	5	3	480	450	225,000	30	3.51	0.01%
18	1SUO	159	5	3	480	96	48,000	60	6.69	0.13%
19	1BO9	73	6	3	960	438	219,000	35	3.43	0.02%
20	1JW2	72	4	4	384	66	33,000	31	4.26	0.09%
21	1I2T	61	4	4	384	64	32,000	21	5.58	0.07%
22	1CCD	77	4	4	384	20	10,000	3	4.75	0.03%
23	2PSR	100	5	4	1,920	468	234,000	339	4.75	0.14%
24	1A7D	118	6	4	5,760	139	69,500	144	5.33	0.21%
25	2LIS	136	6	4	5,760	144	72,000	8	4.84	0.01%
26	1ALU	186	6	4	5,760	419	209,500	288	7.22	0.14%
27	1HXI	121	6	4	5,760	400	200,000	17	3.78	0.01%
28	1JMW	146	6	4	5,760	304	152,000	129	5.25	0.08%
29	1AA2	108	7	4	13,440	768	384,000	3,599	4.15	0.94%
30	1BVC	153	8	4	26,880	1215	607,500	8	5.59	0.00%
31	1BZ4	144	5	5	3,840	16	8,000	31	4.91	0.39%
32	1AEP	161	5	5	3,840	157	78,500	204	5.3	0.26%
33	1DUS	194	6	5	23,040	3840	1,920,000	2,179	4.44	0.11%
34	1B5L	172	6	5	23,040	438	219,000	759	5.44	0.35%
35	1FLP	142	7	6	322,560	7734	3,867,000	4,707	4.65	0.12%

a: the number of amino acids in the protein.

b: the number of helices in the protein.

c: the number of sticks detected by HelixTracer.

d: the number of all possible topologies.

e: the number of valid topologies after applying distance and length screening.

f: the number of structures generated for all valid topologies.

g: the highest rank of the structure that has the correct topology.

h: the Root Mean Square Deviation (RMSD) of C_α atoms of the structure that has the highest rank with the correct topology.

i: the percentage of the highest rank among all generated structures.

Since our method predicts the structure for the helical sticks without building the entire chain, we explored the possibility of applying it to large proteins at multiple local regions. We performed a test on two proteins that have 290 and 322 amino acids respectively (Table 2). For each protein, we generated their volumetric density map at 10Å resolution and used the HelixTracer to detect the SSEs-D sticks. We selected two local regions with closely associated sticks and wanted to see how well our program can predict a near native structure for the local regions without building the entire chain of the protein. Each local region consists of four helical sticks. The structures constructed for each local region were ranked by their effective contact energy. The highest ranked structure that has the correct topology is at the 10,448th of the 6,973,800 pool of structures generated for the first local region (1A0P_G1, Table 2, 2nd row). The structure for region G1 has a backbone RMSD of 3.96Å when it is compared with its native peer (Fig. 15 and Table 2). It is ranked at the top 0.15% in the pool of structures for this region. The two local regions we selected have no common SSE-D, although they may have in principle. We simply combined the ranked list of structures for the first local region (G1) with that for the second local region (G2). Since each list is developed independently from the other, the conflicting assignments need to be eliminated when the two lists are combined. A conflicting assignment is such that the same SSE-S is assigned to both a stick in the first region and a stick in the second region. After this screening, the highest ranked structure with the correct topology (red ribbon in Fig. 15) for eight sticks was ranked the 3,741,775th of a pool of 5.9E+13 structures, about the top 0% of the list.

Our exploratory test about the local regions of large proteins used minimum rules to eliminate the impossible structures. We expect that more geometrical rules can be used to enhance the ranking of the near-native structure. This work suggests that a near-native structure for the helical skeletons can be found near the top of the list ranked by the effective contact energy. In order to generate a few most likely structures, additional evaluation is needed involving statistical analysis of the likely structures, refinement of the structures and using additional information from the density map.

Table 2. Structure prediction of the local regions in two large proteins.

ID	#AA[a]	#hlces[b]	#sticks[c]	#Possible Topologies[d]	#Valid Topologies[e]	#Generated structures[f]	Rank[g]	RMSD[h]	Prct[i]
1AOP_G1		14	4	384,384	69,738	6,973,800	10,448	3.96	0.15%
1AOP_G2		14	4	384,384	84,733	8,473,300	14,673	4.11	0.17%
1AOP	290	14	8			5.90E+13	3,741,775		0%
1WQ3_G1		20	4	1,860,480	184,255	18,425,500	40,485	6.47	0.22%
1WQ3_G2		20	4	1,860,480	280,708	28,070,800	18,412	4.65	0.07%
1WQ3	322	20	8			5.17E+14	32,104,299		0%

a: the number of amino acids in the protein.
b: the number of helices in the protein.
c: the number of sticks used for structure prediction in the region.
d: the number of all possible topologies in the region.
e: the number of valid topologies after applying distance and length screening.
f: the number of structures generated in the region or the total number of the structures evaluated for two regions.
g: the highest rank of the structure with the correct topology among the generated structures.
h: the RMSD of the highest ranked structure with the correct topology among the generated structures.
i: the percentage of the rank for the structure with the highest rank and the correct topology.

The results of the naïve approach shows that it simply becomes very hard to enumerate all the possible topologies, and then screen each of them for validity especially for the large proteins. To be able to work on large proteins, we will introduce a graph representation of the enumeration problem when the length and the distance constraints are present. We will also show the structures simulated from the valid topologies that are generated from the constraint graph.

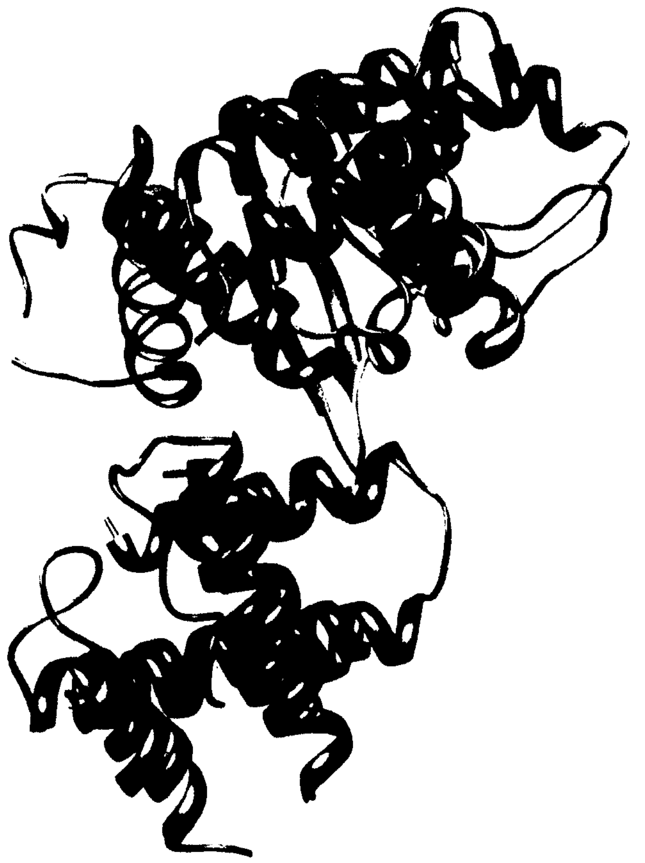


Figure 15. The predicted structure for two regions in 1A0P (PDB ID code). The native structure (grey) and the highest ranked structure with the correct topology (red) are shown.

4.4 THE TOPOLOGY GRAPH

In this Section, we propose a general framework to the topology determination problem using a weighted directed graph. The topology determination problem is then represented as a problem of enumerating constraints paths in a layered graph.

4.4.1 Protein Topology and the Topology Graph

Let $(H_1, H_2, \ldots, H_M)$ be a sequence of the SSEs located on the protein sequence (SSEs-S). Due to the linear nature of the protein sequence, the order of the sequence segments $H_i, i = 1, \dots, M$ is fixed. Let $\{S_1, S_2, \dots, S_N\}$ be the set of sticks detected from CryoEM volumetric density map (SSEs-D). In principle, the number of SSEs-S can be different from SSEs-D due to the errors detecting the sticks and the errors estimating the sequence elements. For simplicity, we only allow $M \geq N$ in our method. The topology determination problem can be described as a problem to find a permutation γ of $\{1, 2, \dots, N\}$ such that assigning H_i to $S_{\gamma(i)}, i = 1, \dots, N$ minimize the assignment score. In the assignment, each H_i is assigned to $S_{\gamma(i)}$ in one of the two opposite directions.

It has been observed that various constraints related to two adjacent SSEs can be used to reduce the number of possible topologies significantly [120,162,183,186]. One such constraint comes from the length of the loop that connects the two SSEs-S. The other constraint we use in this method is the variance in length between secondary structure elements. The assignment between SSEs is not possible if the variance in length, in terms of number of amino acids, between the two elements (SSE-S and SSE-D) is more than %60. See Section 4.2 for more details about constraints. To estimate the number of amino acids in SSE-D, we simply divide the length of the stick by 1.5 which represents the rise of a helix in real. Other constraints such as the geometrical constraints and the connectivity constraints have also been used [120]. The nature of such constraints is that it involves the assignment of two SSEs. Such constraints involving two SSEs can be naturally represented by an edge in a graph. In Section 4.7 we will explain the method of assigning weights to the edges of the graph.

We use a weighted directed graph $G_{TOPO} = (V, E, w)$ to represent the topology problem. V has two special nodes $START$ and END and $N \times N \times 2$ "regular" nodes. More precisely,

$$V = \{(i, j, t) | 1 \leq i \leq M, 1 \leq j \leq N, t \in \{0,1\}\} \cup \{START, END\}$$

$$E = \left\{ \begin{array}{c} \left\{\left(((i,j,t)(i',j',t')) \middle| \begin{array}{c} 1 \le i \le M-1, i < i' \le \min(M, i+M-N+1) \\ 1 \le j \ne j' \le N, t, t' \in \{0,1\} \end{array}\right)\right\} \cup \\ \left\{\left(((START)(i,j,t)) \middle| 1 \le i \le M-N+1, 1 \le j \le N, t \in \{0,1\}\right)\right\} \cup \\ \left\{((i,j,t)(END) \middle| N \le i \le M, 1 \le j \le N, t \in \{0,1\})\right\} \end{array} \right\}$$

Let $w(i,j,t)(i',j',t') \ge 0 \in \mathbb{R}$ be a real number associated with the edge $e(i,j,t)(i',j',t') \in E$. Each element in SSE-S can be assigned to each element in SSE-D in two different directions. In G_{TOPO}, each possible assignment between SSE-S and SSE-D is represented by two nodes, one node for each direction. A node (i,j,t) represents an assignment of H_i to S_j in t direction (Fig. 16B). An edge from node (i,j,t) to (i',j',t') represents the assignment of $H_{i'}$ to $S_{j'}$ in direction t' after the assignment of H_i to S_j in direction t. In reality, the detection of SSEs-D is not error-free due to noisiness in the CryoEM volumetric density map. Thus, to consider the error of detecting helices from the map which leads to missing helices, we add an edge between nodes in row i and rows $i+1, i+2, \dots, \min(M, i+M-N+1)$. The maximum number of rows can be skipped each time is equal to the number of helices missed (i.e., the difference between SSEs-S and SSEs-D). Initially, if the two assignments involved in an edge do not satisfy the constraints described in Section 4.2, the weight of the edge was assigned to ∞, indicating that the edge is not feasible. Otherwise, the weight of the feasible edges was assigned to initial weight w_{init}. The initial weight was set to $|3.8 * loopLength(H_i, H_{i'}) - Dist_{map}((j,t),(j',t')) + \varepsilon|$, where $loopLength(H_i, H_{i'})$ is the number of amino acids in the loop between the two SSEs-S, H_i and $H_{i'}$. $Dist_{map}((j,t),(j',t'))$ is the Euclidian distance from the ending point of S_j in t direction to the beginning point of $S_{j'}$ in t' direction, and ε was used to be zero if $Dist_{map}$ is less than seven and 50 otherwise. The reason we used ε=0, if the two sticks are less than 7Å apart, is the fact that if the sticks are less than 7Å apart they are more likely to be connected. In a later step, we will use the CryoEM volumetric density map to re-assign weights of the edges with initial weight w_{init}. The weights of special edges from $START$ and to END nodes should always remain zero.

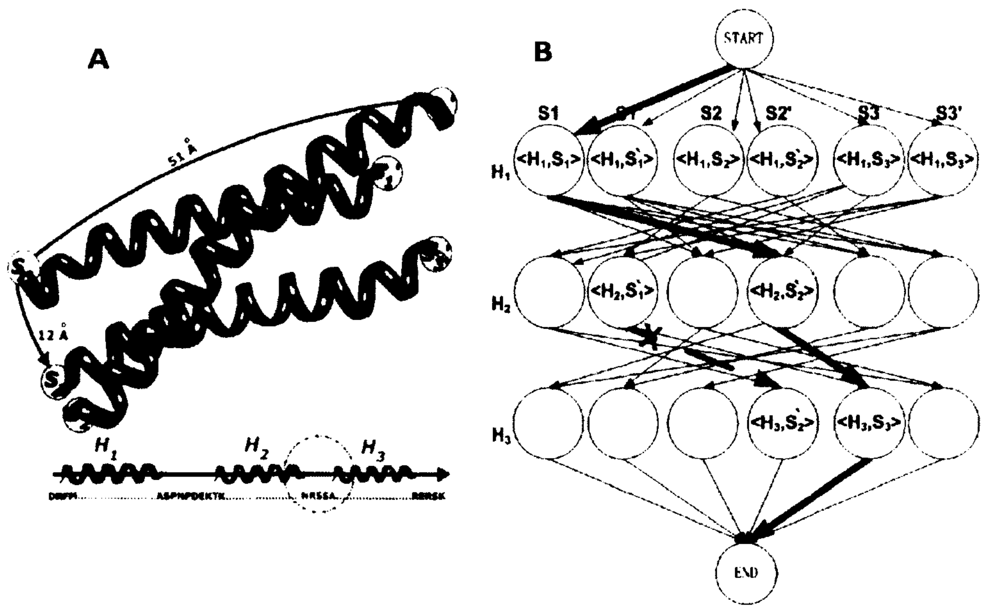


Figure 16. The constraint graph built for the N-terminal domain of syntaxin (PDB ID code: 1BR0). The three SSEs-D and the three SSEs-S were shown in (A). The amino acids in the loop between H_2 and H_3 were highlighted in a disk. The measured distance between point S_1 and S_2' was indicated in (A). The constraint graph was drawn in (B) with an invalid edge labeled with X and colored in red. The path that corresponds to the true topology is highlighted in green.

Fig. 16*B* illustrates the topology graph. A node $v = (i, j, t)$ is located on the *i*th row, *j*th column and *t*th position inside the box drawn in a dashed line. The graph is a directed graph in which each edge points downwards. An edge represents a valid relationship between two SSEs-S. Since each SSE-D can be only assigned to one SSE-S, there is no edge between the nodes in the same column, and similarly there is no edge between the nodes in the same row. When *M*=*N* each edge links between two adjacent rows. When $M > N$ (Fig. 17), an edge links between non-consecutive rows, where the maximum number of rows can be skipped is $M - N$. Special edges are drawn from node $START$ to each node in the first $M - N + 1$ rows and similarly from each node at the last $M - N + 1$ rows to node END. The weight on the special edges is zero, and the other weights are non-negative. However, only non-special edges and edges with weights not equal to ∞ are shown in Fig. 17 for simplicity.

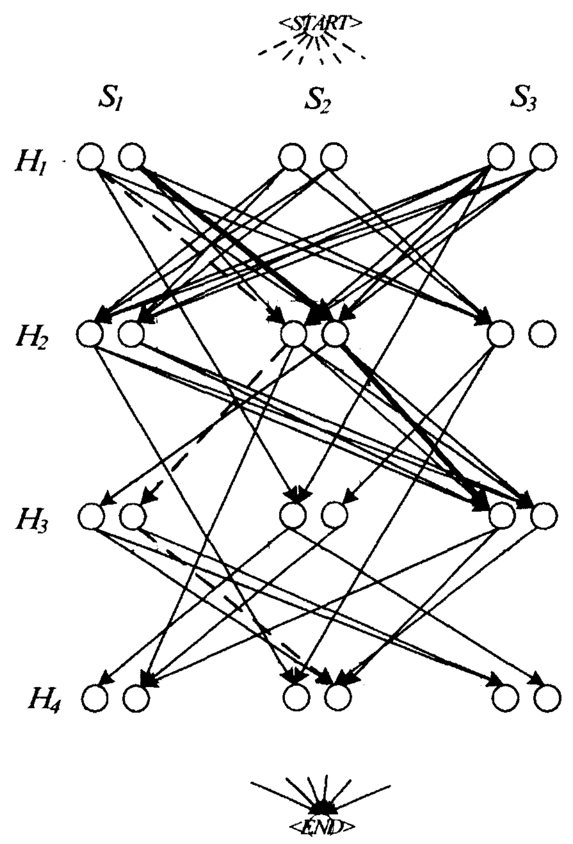


Figure 17. A topology graph with number of SSEs-D is less than number of SSEs-S. The dashed red line is an example of one invalid path. The green line is one example of a possible valid path.

4.4.2 Constraints in Finding Valid Paths

A path of G_{TOPO} begins at $START$ node and ends at END node. The problem of enumerating all valid topologies becomes the problem of enumerating all valid paths. Not every path is a valid path. For example, those paths that visited the same column more than once are not valid paths, since each stick cannot be assigned to multiple SSEs-S. A valid path needs to satisfy the following constraints:

(1) The path begins at $START$ node and stops at END node.

(2) A valid path should visit all columns, each column exactly once.

More precisely, a valid path is a sequence of nodes $(< START >, < i_1, j_1, t_1 >, < i_2, j_2, t_2 >, \dots, < i_N, j_N, t_N, >, < END >)$ where $\{i_1, i_2, \dots, i_N\} \in \{1,2, \dots, M\}$, $\{j_1, j_2, \dots, j_N\} = \{1, 2, \dots, N\}$ and $\{t_1,\ t_2, \dots, t_N\} = \{0,1\}$. The number of nodes in the valid path, exclude $START$ and END nodes, should be exactly equal to number of columns (N). An example of a valid path is shown in green thick lines and a non-valid path is shown in red dashed lines in Fig. 17. With the formulation of the topology graph and the valid path, a valid topology corresponds to a valid path from $START$ to END. The optimal path is the path that has the minimum cost. In this work, the cost of a path is simply the sum of the weights along the path, $c(p) = \sum_p w$.

4.5 DEPTH FIRST SEARCH APPROACH

The topology graph was built to represent the pair-wise relationship between any two assigned pairs of two SSEs-D and two SSEs-S. Given the representation of the topology graph, the problem of enumerating all valid topologies becomes the problem of enumerating all valid paths. A valid path needs to satisfy the constraints in Subsection 4.4.2. In order to enumerate all the valid paths, we performed a depth-first search in the topology graph.

4.5.1 Construction of the Protein Chain

Since a valid topology determines the assignment between the sequence of SSEs-S $H_1, H_2, \dots, H_M$ and the set of SSEs-D $\{S_1, S_2, \dots, S_N\}$, the backbone of the protein and hence the side chains of the protein can be simulated. For each valid topology, five hundred protein conformations were constructed [183]. This was done by randomly sampling the freedom of translation and shift parameters. The translation is the movement along the central axis of the helical SSE-D. The maximum translation is set to 1.5Å, about the rise of an amino acid in a helix. The shift is the amino acid position shift from the true position of the helix on the protein sequence. The shift parameter was introduced to partially simulate the error of the secondary structure prediction. The range of the shift is [-2,2], which refers to the inaccurate shift of two position to the left and right of the true position of the amino acid. The conformations were evaluated and sorted using the multi-well energy function previously used for naïve approach [51].

4.5.2 Results

We used a data set consisting of 25 proteins to evaluate the enumeration and the accuracy of the protein conformations predicted. This data set was randomly selected from the helical proteins in the PDB with the requirement of having three to seven helices. No large proteins were selected for this approach because of the intensive time it takes to evaluate the structures after finding all valid topologies. We first generated the protein volumetric density map to 10Å resolution using EMAN [1] for the 25 proteins. HelixTracer [115] was then used to detect the helical SSEs-D from each of the volumetric density maps. The SSEs-D for the helices were obtained from the true structure as a test. We built a topology graph for each protein and generated all the valid paths. For each valid path, we construct 500 conformations each consisting of the coordinate of the heavy atoms in the backbone and side chains of the helices. The conformations were sorted based on their effective energy. Table 3 shows the highest rank (column 9) of the constructed structure with the true topology. For example, in the case of 1BZ4 (Table 3, row 23), the highest ranked conformation with the correct topology is at the 70th (the top 0.5% of all the 14,000 structures generated) of the list that was sorted by the energy. We compared the accuracy of the constructed helices with the helices in the native proteins using the Root-Mean-Square-Deviation (RMSD) between the corresponding C_α atoms on the helices. The RMSD between this conformation and the true structure is 3.732Å. The best ranked structure with the correct topology was aligned with the native structure for protein 1BRO (PDB ID code) in Fig. 18. For clear viewing, the side chains of one of the three helices are shown. The RMSD for this structure is 3.808Å (Table 3, row 5).

In addition to the 25 proteins in Table 3, we built the layered graph for four larger proteins with nine to 14 helices in Table 4. It is expected that the total number of possible topologies increase quickly as the number of helices increase. However, the number of valid paths enumerated from the graph is a very small percentage of the total possible number. In the case of 2H7O (Table 4), there are only 755,142 valid paths of 371,960 million total possible topologies, a number that was impossible for us to generate in our previous approach (naïve approach, Section 4.3). The number of valid topologies varies from protein to protein. For some proteins, building the topology graph may not be needed, since there is not a big difference between the total number of the possible topologies and the number of valid topologies. However, we noticed that as the number of helices increase, the benefit of the graph become evident since it does not have to enumerate all the possible topologies.

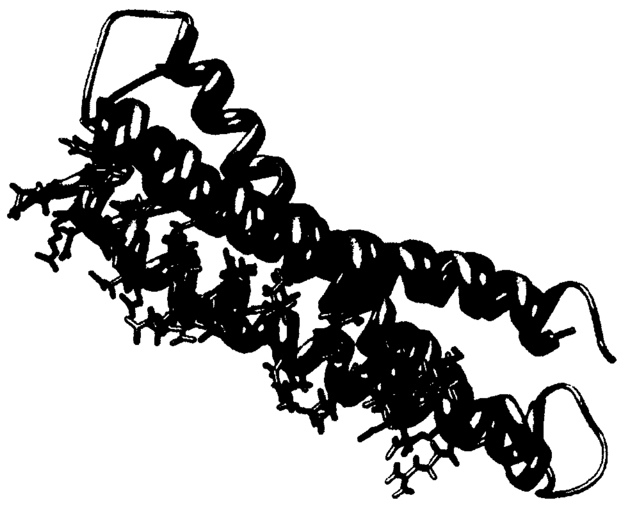


Figure 18. The predicted structure with the correct topology for protein 1BRO (PDB ID code). The native structure of 1BRO in gray is superimposed with the highest ranked predicted structure (in orange) with the correct topology of the skeletons. The side chains are only shown for one of the three helices for clear viewing. Note that the energy evaluation does not rely on the loop. The helices were directly connected without the loop information.

Table 3. The highest rank of the structure with the correct topology using DFS.

No	ID	#AA[a]	#hlces[b]	#sticks[c]	#Possible Topologies[d]	#Valid paths[e]	#structures[f]	$Rank_{bestE}$[g]	$RMSD_{bestE}$[h]	Prcng[i]	Time[j]
1	1DP3	55	3	3	48	4	2.000	8	5.031	0.4%	309
2	1A2T	149	3	3	48	48	24,000	1	6.085	0.004%	280
3	1AIL	73	3	3	48	18	9,000	6	3.544	0.06%	251
4	1IRE	81	3	3	48	36	18,000	19	4.077	0.11%	292
5	1BRO	120	3	3	48	14	7,000	7	3.808	0.1%	687
6	1B67	68	3	3	48	16	8,000	1	3.651	0.01%	147
7	1AYI	87	4	3	192	108	54,000	17	4.361	0.03%	311
8	1BEA	127	4	3	192	156	78,000	175	3.586	0.2%	400
9	1GXG	85	4	3	192	131	56,500	206	3.278	0.36%	531
10	2EZH	75	4	3	192	128	64,000	74	2.927	0.12%	529
11	1A3C	181	4	3	192	96	48,000	461	3.274	0.9%	201
12	1PZ4	116	4	3	192	84	42,000	33	5.126	0.07%	275
13	1LIH	164	4	3	192	168	84,000	72	4.519	0.09%	495
14	1BO9	73	6	3	960	420	210,000	229	2.514	0.11%	402
15	1JW2	72	4	4	384	122	61,000	581	3.542	0.95%	493
16	1I2T	61	4	4	384	136	68,000	108	3.279	0.16%	490
17	1CCD	77	4	4	384	59	29,500	8	4.279	0.03%	393
18	2PSR	100	5	4	1,920	572	286,000	300	5.499	0.1%	518
19	1A7D	118	6	4	5,760	175	87,500	203	3.250	0.23%	676
20	2LIS	136	6	4	5,760	224	112,000	3	3.944	0.003%	493
21	1JMW	146	6	4	5,760	380	190,000	1,174	5.68	0.62%	781
22	1BVC	153	8	4	26,880	3,573	1,786,500	182	3.827	0.01%	1161
23	1BZ4	144	5	5	3,840	28	14,000	70	3.732	0.5%	879
24	1AEP	161	5	5	3,840	624	312,000	755	4.870	0.24%	1287
25	1B5L	172	6	5	23,040	816	408,000	1,436	4.374	0.35%	1163

a: the number of amino acids in the protein.
b: the number of helices in the protein.
c: the number of sticks detected by HelixTracer.
d: the number of all possible topologies.
e: the number of all valid topologies (valid paths) after applying distance and length .screening
f: the number of generated structures for all topologies (500 each).
g: the highest rank of the structure with the native topology of the skeletons.
h: the root mean square deviation (RMSD) of Cα atoms of the structure in column g.
i: the percentage of the highest ranked structure in g among all generated structures.
j: the time in millisecond needed to build, process, and traverse (find all valid paths) the graph .

We included a column of the time consumption for the enumeration in Table 3. The value is the total time from building the graph, processing the graph and the traversal for all the valid paths. For example, it only takes 1161 milliseconds to generate all the valid paths for 1BVC (row 22). It appears that the time spent on building the topology graph which is about $O(MN^2)$ pays off during the later step of the enumeration. Although we did not include the detailed comparison between the time consumption using the current topology graph and those from the naïve enumeration, the time saving is significant. Such comparison will be shown later (Subsection 4.6.4, Table 5). We were not able to generate all the valid paths for proteins with more than seven helices using parallel computers in the naïve approach [183]. Now we can enumerate all the valid paths of proteins with 14 helical SSEs-S and nine SSEs-D sticks (Table 4) using one CPU. We expect this is particularly true when the β-strands are considered in the graph which increases the total number of the possible topologies quickly.

Table 4. Enumeration from the constraint graph of large proteins.

No	ID	#AA[a]	#hlces[b]	#sticks[c]	#Possible Topologies[d]	#Valid paths[e]	Percentage[f]
1	1NG6	148	9	7	23m	3,668	0.02%
2	1OFC	304	13	8	13,284m	33,094,704	0.25%
3	1ZA0	275	13	8	13,284m	11,632,336	0.09%
4	2H7O	303	14	9	371,960m	755,142	0.0002%

a: the number of amino acids in the protein.
b: the number of helices in the protein.
c: the number of sticks detected by HelixTracer.
d: the number of all possible topologies (rounded to millions).
e: the number of valid paths (topologies) after applying distance and length screening.
f: the percentage of number of valid topologies to the all possible topologies after applying screening.

4.6 DYNAMIC PROGRAMMING ALGORITHM

We will illustrate our dynamic programming algorithm to find the topology with the minimum cost. This algorithm is, as expected, significantly faster than the naïve and the depth first approaches. It allows us to find the topology with the minimum cost for large proteins. We also developed a method to enumerate the K-minimum cost topologies using the topology graph (Subsection 4.6.3). This is the first dynamic programming approach to rank the topologies of the SSEs. The complexity of the proposed dynamic programming algorithm is $O((D+1)^2N^2 2^N)$. Where D is $M-N$, M is the number of SSE-S and N is the number of SSE-D. In this dynamic programming algorithm, the topology graph introduced in Section 4.4 is used. The geometrical constraints illustrated in Section 4.2 are applied in order to build the graph.

4.6.1 The Complexity of the Problem

In this Section we will proof the complexity of the problem. For the simplicity, we will use the case that number of SSEs-D is equal to the number of SSEs-S. The solution space for the problem of assigning N SSEs-D to N SSEs-S segments is $N!\,2^N$. It is often desired to reduce the solution space to a set of small number of highly ranked possible topologies, and the correct topology is contained in the set. We investigated the complexity of the reduction problem that involves the constraint from a pair of nodes. We will show that finding the set of the top-K ranked topologies is NP hard by showing that finding the top-ranked topology in the topology graph is already NP hard. The dynamic programming algorithm we are introducing will improve the shortest path search from $N!\,2^N$ time, as in the naïve approach, to $O((D+1)^2N^2 2^N)$ time.

Note that the problem of finding the shortest valid path in the topology graph is equivalent to finding the shortest valid path from a node in row 1 to a node in row N. We know that the following variant of travelling salesman problem is NP hard: Given a complete weighted graph $G = (V, E, c)$ find the minimum cost Hamiltonian path in G. We call this variant as MCHP. We provide a polynomial-time reduction from MCHP to our problem. Intuitively, this reduction is possible because the constraint of not revisiting a column in the shortest path in G_{TOPO} is similar to the constraint of not revisiting a vertex in a Hamiltonian path.

Claim. The problem of finding the shortest valid path from a node in row 1 to a node in row N in the topology graph G_{TOPO} is NP hard.

Proof. Consider an instance of MCHP, $\mathrm{G}_{\mathrm{MCHP}} = (\mathrm{V}, \mathrm{E}, \mathrm{c})$ of N nodes, with $c(i,j)$ being the cost of travelling from node i to node j. We construct an instance of $\mathrm{G}_{TOPO} = (\mathrm{V}^{'}, \mathrm{E}^{'}, \mathrm{w})$ as follows.

$V^{'} = \{\, (i,j,t) : i \in V \; and \; 1 \le j \le N, 0 \le t \le 1 \,\}$

$E' = \left\{ \left(((k,i,t),(k+1,j,t')), ((k,j,t),(k+1,i,t')) \right) \middle| \begin{matrix} (i,j) \in E \; and \; 1 \le i,j \le N \\ 1 \le k \le N-1, 0 \le t,t' \le 1 \end{matrix} \right\}$

$w\left((k,i,t),(k+1,j,t^{'})\right) = w\left((k,j,t),(k+1,i,t^{'})\right) = c(i,j) \; for \; 1 \le i,j \le N, 1 \le k \le N-1, 0 \le t, t^{'} \le 1$

Intuitively, for each vertex i of MCHP, we create a row of nodes in G_{TOP}. For each edge (i,j) of MCHP, we create the links, using the weight $c(i,j)$, between the nodes at the i^{th} column and the j^{th} column in two consecutive rows. Now consider the path $(1,j_1,t_1),(2,j_2,t_2) \ldots (N,j_N,t_N)$ in $G_{TOPO} = (V^{'},E^{'},w)$ and the path $j_1, j_2, \ldots, j_N$ in G_{MCHP}. Noticing that both paths have the same cost, if $(1,j_1,t_1),(2,j_2,t_2) \ldots (N,j_N,t_N)$ is the shortest valid path in G_{TOP}, then $j_1, j_2, \ldots, j_N$ is a minimum cost Hamiltonian path in G_{MCHP} ■

4.6.2 Finding the Shortest Path Satisfying the Constraints

Finding the shortest path is a long studied problem because it is general and fundamental in many areas of computer and non-computer sciences. The problem has been extensively studied in different applications such as networking, robotics, operations research, and Plant and facility playout. Many other applications can be found in the study of [191]. Several algorithms can be found in the literature that find the shortest path for a given graph, which can be defined as the path between two vertices (i.e., source and destination), in which the sum of edges weights is minimum. One well known, and largely used example, is Dijkstra algorithm [192]. Dijkstra algorithm finds the shortest path in a single source and destination graph with positive edge weights. Another example is, Bellman-Ford [193,194] algorithm which solves the problem of finding the shortest path in a single source destination graphs with the existence of negative edge weights. In contrast, Floyd-Warshall [195,196] and Johnson's [197] algorithms for example solve the problem on a graph between every pair of vertices. However, these generic shortest path algorithms cannot be applied directly to this problem due to the constraints that need to be satisfied by a valid path. In this Subsection, we give a dynamic programming algorithm to find the shortest valid path.

The valid paths represent the valid topologies, and ideally, the valid path with the minimum cost represents the true topology of the protein. To find the shortest valid path, our method maintains of, at each node $v = (i, j, t)$, the shortest path for each possible set of columns can be visited. Let $S = \{1,2, \ldots, N\}$ is the set of all columns in the graph, and let $U^{(i)}$ is the set of subsets of S with number of elements equals to x. Where $max(1, i-(M-N)) \leq x \leq min(i, N)$. $U^{(i)}$ represents all columns can be visited to form a valid path to node $v = (i, j, t)$. That it, for each $v \in V$ at level i, we maintain all subsets of columns could be visited from $START$ to v to form a valid path. Let $U \subset U^{(i)}, j \in U$ with U representing the set of columns visited in a path. The algorithm only saves the shortest path among $|U|!$ possible different paths for the set of columns in U. For example, there are maximum 3! =6 different paths that visit the three columns when $U = \{1,3,5\}$, regardless of the order of the visits. The different cardinalities of U in $U^{(i)}$ at level i represent the different possible order of the node along the path. For example, there are two possible orders for nodes at level $i = 2$ along valid paths (Fig. 19*B*). One possible path may have nodes at level two visited directly from $< START >$ node. Another possible path may have the nodes at level two visited after nodes at level one visited. Let $f(v, U)$ be the minimum cost of the path for reaching v by using the elements of U as columns.

$$f(v,U) = f\big((i,j,t),U\big)$$
$$= \begin{cases} 0 & v =< START > \\ w(< START >, v) = 0 & 1 \leq i \leq M-N+1, U = \{j\} \\ \min\limits_{j' \in U\{j\}, t' \in \{0,1\}} \left[f\left((i',j',t'), U\{j\}\right) + w\left((i',j',t'),(i,j,t)\right)\right] & \\ & i \in [2,M], \max(1, \mathrm{i}-(\mathrm{M}-\mathrm{N})-1) \leq \mathrm{i}' < i, j \in U \\ \infty & \text{otherwise} \end{cases}$$

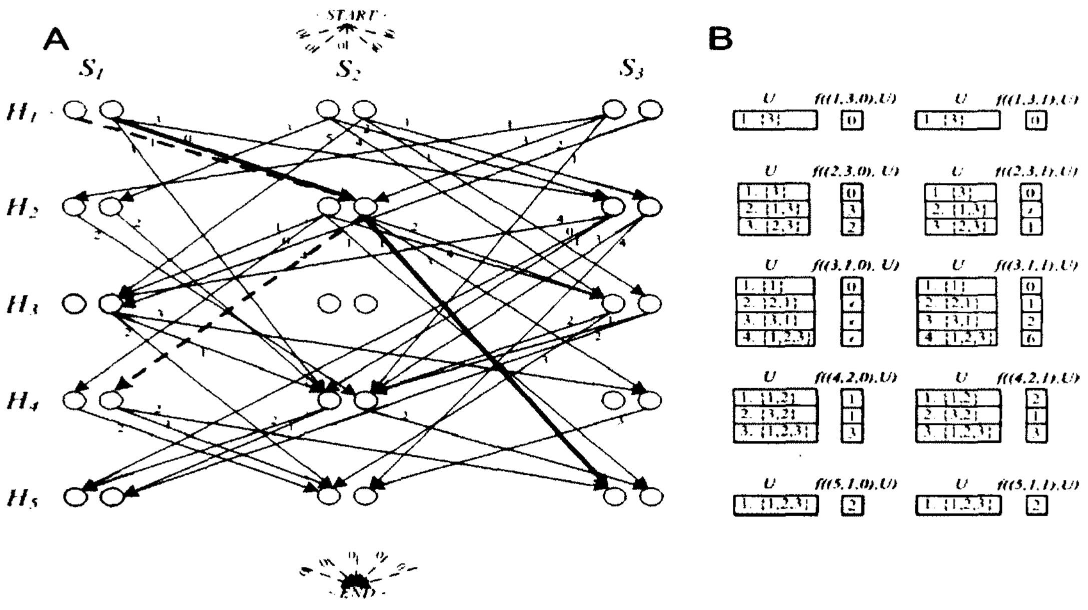


Figure 19. The topology graph built for NS1 protein from Influenza A virus (PDB ID 1AIL). (A) The weights were restricted to integers to save the space in drawing. (B) The example of U and *f*(*v*, *U*) for the nodes (1,3,0), (1,3,1), (2,3,0), (2,3,1), (3,1,0), (3,1,1), (4,2,0), (4,2,1), (5,1,0) and (5,1,1). The shortest path is shown in the thick green lines and an example of an invalid path is shown in the red dashed lines.

Fig. 19*B* shows an example of $f(v, U)$ for some nodes (i.e., dark boxes in Fig. 19*A*). At each level i, there are $|U^{(i)}|$ subsets represent all possible sets of columns can be visited. Only subsets that have *j* as an element are shown. Other subsets do not have *j* as an element, trivially, have a value of $f(v, U)$ equals to ∞. At node (2, 3, 0), there are six instances of U with two different cardinalities. The subsets $U_1 = \{3\}$, $U_2 = \{1,3\}$ and $U_3 = \{2,3\}$ are only shown. The minimum cost of reaching (2, 3, 0) using column 1 and column 3 is 3. The minimum cost of reaching it using column 2 and column 3 is 2. Finally, the minimum cost of reaching the node directly from node $START$ is trivially zero. The pseudo code of algorithm 1 that used in order to find the shortest valid path is given in Fig. 20.

Notation:

- For a given set $S = \{1,2,3,\dots,N\}$, we define $U^{(i)}$ as the set of the subsets of S with cardinality of x, where $\max(1, i-(M-N)) \le x \le \min(i, N)$, $1 \le i \le M$, and $M \ge N$.
- We define $U_k^{(i)}$ as the k^{th} subset of $U^{(i)}$.
- $v_{(j,t)}^{i}$ is the j^{th} column with t direction in the i^{th} layer.
- $f\left((i,j,t), U_k^{(i)}\right) \leftrightarrow f(v_{(j,t)}^{i}, U_k^{(i)})$.

Algorithm 1 (CalculateSets):

```
$\boldsymbol{input}$ : Weight array $w$ (no edge $\Rightarrow$ $\infty$ weight) and V
$\boldsymbol{output}$ : Minimum path cost $min_{cost}$
$S \leftarrow \{1,2,3,\dots,N\}$
$f(*, U_k^{(i)}) \leftarrow 0, |U_k^{(i)}| = 1, 1 \le i \le M-N+1, 1 \le k \le |U^{(i)}|$
$f(*, U_k^{(i)}) \leftarrow \infty, |U_k^{(i)}| \ne 1, 2 \le i \le M, 1 \le k \le |U^{(i)}|$
$\boldsymbol{for}\ i \leftarrow 2\ to\ M\ \boldsymbol{do}$
  $\boldsymbol{for}\ k \leftarrow 1\ to\ |U^{(i)}|\ \boldsymbol{do}$
    $\boldsymbol{for}\ each\ p \in U_k^{(i)}, |U_k^{(i)}| > 1\ and\ t \leftarrow 0\ to\ 1\ \boldsymbol{do}$
      $U' \leftarrow U_k^{(i)} \backslash p$
      $\boldsymbol{for}\ each\ q\ \in U'\ and\ t' \leftarrow 0\ to\ 1\ \boldsymbol{do}$
        $f(v_{(p,t)}^{i}, U_k^{(i)}) = \min_{\max(1, i-(M-N)-1) \le i' < i} \left\{ f\left(v_{(q,t')}^{i'}, U'\right) + w\left(v_{(q,t')}^{i'}, v_{(p,t)}^{i}\right), f(v_{(p,t)}^{i}, U_k^{(i)}) \right\}$
      $\boldsymbol{endfor}$
    $\boldsymbol{endfor}$
  $\boldsymbol{endfor}$
$\boldsymbol{endfor}$
$min_{cost} = \min\{f(v_{(j,t)}^{i}, S) : N \le i \le M, 1 \le j \le N, 0 \le t \le 1\}$
```

Figure 20. The pseudo code of Algorithm 1 (CalculateSet). Algorithm 1 is used to build the subsets of possible paths for each node. The algorithm is also used to simply find the shortest path at the end.

If we assume that the access of any entry in the table $f(v, U)$ is in constant time, the time to find the shortest valid path is $O((D+1)^2 N^2 2^N)$. The dynamic programming approach reduced the $N!$ component in the naïve approach $\binom{M}{N}(N!\,2^N)$ to $(D+1)^2 N^2$, where $D = M - N$. However, the nature of the problem is still NP hard. The following is the analysis of the run time required for the algorithm to calculate all subsets:

$$Time = \sum_{i=2}^{M} \left[(i - \max(1, i - (M - N) - 1)) * \sum_{k=\max(1,i-(M-N))}^{\min(i,N)} 4 * k * (k - 1) * \binom{N}{k} \right]$$

$$Time < \sum_{i=2}^{M} \left[(M - N + 1) * \sum_{k=\max(1,i-(M-N))}^{\min(i,N)} 4 * k * (k - 1) * \binom{N}{k} \right]$$

Let $D = M - N$ and $X_k = 4 * k * (k - 1) * \binom{N}{k}$, then

$$time < \sum_{i=2}^{M} \left[\{D + 1\} * \sum_{k=\max(1,i-D)}^{\min(i,N)} X_k \right]$$

We claim that

$$\sum_{i=2}^{M} \left[\{D + 1\} * \sum_{k=max(1,i-D)}^{min(i,N)} X_k \right] = (D + 1)^2 * \sum_{k=2}^{N} X_k$$

Thus

$$time < (D + 1)^2 * \sum_{k=2}^{N} \left(4 * k * (k - 1) * \binom{N}{k} \right)$$

Now, after differentiating the Binomial series

$$(x + 1)^N = \sum_{k=0}^{N} \binom{N}{k} x^k$$

With respect to x twice

$$N * (x + 1)^{N-1} = \sum_{k=1}^{N} k * \binom{N}{k} x^{k-1}$$

$$N * (N - 1) * (x + 1)^{N-2} = \sum_{k=2}^{N} k * (k - 1) \binom{N}{k} x^{k-2}$$

And then substituting $x = 1$, we get

$$N * (N - 1) * 2^{N-2} = N^2 * 2^{N-2} - N * 2^{N-2} = \sum_{k=2}^{N} \left(k * (k - 1) * \binom{N}{k} \right)$$

Therefore,

$$time < (D + 1)^2 * 4 * N^2 \ * 2^{N-2} = (D + 1)^2 * N^2 \ * 2^N.$$

The proof of the claim is by induction on D. For $D = 0$; that is, $M = N$. Then

$$\sum_{i=2}^{M}\left[\{0+1\} * \sum_{k=\max(1,i-0)}^{\min(i,N)} X_k\right] =$$

$$\sum_{i=2}^{M}\left[\sum_{k=\max(1,i)}^{\min(i,N)} X_k\right] = \sum_{i=2}^{M}\left[\sum_{k=i}^{i} X_k\right] = \sum_{i=2}^{M} X_i = (1)^2 * \sum_{k=2}^{N} X_k$$

And so the base for the induction is proven.

Now, assume that the claim holds for $D = q$. This means that

$$\sum_{i=2}^{M}\left[\{q+1\} * \sum_{k=\max(1,i-q)}^{\min(i,N)} X_k\right] = (q+1)^2 * \sum_{k=2}^{N} X_k$$

Divide both sides by $(q+1)$ to get

$$\sum_{i=2}^{M}\left[\sum_{k=\max(1,i-q)}^{\min(i,N)} X_k\right] = (q+1) * \sum_{k=2}^{N} X_k$$

We need to show that the claim holds for $D = q+1$. From the behavior of the domain of i (from 2 to M) we note that for $i = 2$ to $d+1$ the starting value of k always equals to 2 (since when $k = 1, X_k = 0$) and from $i = d+2$ to M the value of k goes from 2 to N.

$$\sum_{k=\max(1,i-(q+1))}^{\min(i,N)} X_k = \begin{cases} \displaystyle\sum_{k=\max(1,i-q)}^{\min(i,N)} X_k & for \quad 2 \le i \le q+2\,; \\ \left(\displaystyle\sum_{k=\max(1,i-q)}^{\min(i,N)} X_k\right) + X_{i-(q+1)} & for \quad q+3 \le i \le M. \end{cases}$$

Hence,

$$\sum_{i=2}^{M}\left[\{q+2\} * \sum_{k=\max(1,i-(q+1))}^{\min(i,N)} X_k\right] = (q+2)\left\{\sum_{i=2}^{M}\left[\sum_{k=\max(1,i-(q+1))}^{\min(i,N)} X_k\right]\right\}$$

$$= (q+2)\left\{\sum_{i=2}^{q+2}\left[\sum_{k=\max(1,i-q)}^{\min(i,N)} X_k\right] + \sum_{i=q+3}^{M}\left[\sum_{k=\max(1,i-q)}^{\min(i,N)} X_k\right] + \sum_{i=q+3}^{M}\left[X_{i-(q+1)}\right]\right\}$$

$$= (q+2)\left\{\sum_{i=2}^{M}\left[\sum_{k=\max(1,i-q)}^{\min(i,N)} X_k\right] + \sum_{i=q+3}^{M}\left[X_{i-(q+1)}\right]\right\}$$

Then, using the inductive hypothesis as stated above for the first summation , re-indexing the second summation (by letting $k = i - (q+1)$ and $M = N + (q+1)$) and combining, we get

$$\sum_{i=2}^{M}\left[\{q+2\} * \sum_{k=max(1,i-(q+1))}^{min(i,N)} X_k\right] = (q+2)\left\{\left[(q+1) * \sum_{k=2}^{N} X_k\right] + \sum_{k=2}^{N} X_k\right\}$$

$$= (q+2)^2 * \sum_{k=2}^{N} X_k$$

And this completes the proof.

4.6.3 *K*-Shortest Valid Paths Satisfy the Constraints

The prediction of secondary structures from both sequence and CryoEM volumetric maps may contain some noise. For instance, the current accuracy of the prediction of SSEs-S is around 80% which means that the prediction will have noise in some cases. Similarly, the prediction of SSEs-D might contain some noise such as in the position of sticks, positive false prediction, or the length of sticks. The shortest path may not be the true topology, although the true topology often has near minimum cost. On the other hand, enumerating all paths and then evaluating them is not practical for large proteins as well (as shown in Section 4.5).

The K shortest path problem is a generalization of the shortest path problem where not only the shortest but the first K paths $(p^1, p^2, \dots, p^K), K \geq 1$ is determined in non-decreasing order of the cost between two vertices. The cost of each path p^i is smaller than the cost of path p^{i+1}. Two types of paths were examined in the literature, simple path where no repeated vertex is allowed and non-simple path in which repeated vertices are allowed. Finding the simple K-shortest paths makes the problem remarkably harder. The early attempt to solve the problem of finding K-shortest paths is done by Hoffman and Pavley [198] which with other attempts led to an exponential run time [199]. Many papers have examined the problem since then [200,201,202,203,204,205]. The fastest algorithm known to date is the generalization algorithm of Lawler [206] for the algorithm first proposed by Yen [202] which can be implemented in $O(kn(m + nlogm))$ worst case run time, where n is the number of nodes and m is the number of edges. Although the asymptotic worst-case bound for Yen's algorithm is still unbeaten [203], several algorithms have been proposed to improve the expected run time of Yen's algorithm with the same worst-case bound [204,205].

The traditional algorithms of finding K-shortest paths can't directly be applied to the problem being proposed in this work. Unfortunately, these kinds of algorithms are not suitable for domain-specific graphs such as the topology graph in this work. One challenge in the topology graph is the nature of the shortest path in which the next node to visit is a function of the nodes visited so far. To find the shortest path or the top-K paths, one possible approach is to apply a simple and fast algorithm (i.e., Dijkstra) to find the first shortest path between node $START$ and node END that verifies the constraint. This solution is infeasible because, interestingly, the probability of finding a path verifies the constraint decreases considerably with N. The number of valid paths $(N!\,2^N)$ compared to the total number of possible paths $(2N(2N-2)^{N-1})$ in a fully connected graph is very small. For example, for a $N = M = 5$ fully connected graph, the probability of finding one path verifies the constraint is 0.09375, and for a graph with $N = M = 10$ the probability decreases to 0.0009375 (the total number of possible paths is approximately 4 X 10^{12}).

4.6.3.1 Problem Definition and Notation

For the topology graph we define a valid path as a sequence of nodes of the form $p = (START = v_0, v_1, v_2, \dots, v_N, v_{N+1} = END)$. The cost of the path p is simply the sum of edges weights along the path and denoted by $c(p)$.

Let $p_{i,j}$ denotes a valid path between nodes v_i and v_j. Let $\mathrm{P}_{i,j}$ denotes the set of all valid paths in G_{TOPO} between the pair of nodes v_i and v_j, and $\mathrm{P}_{i,j}^{K} =$

$\{p_{i,j}^1, p_{i,j}^2, \dots, p_{i,j}^K\} \in \mathrm{P}_{i,j}$ be the set of K-shortest paths over the set of paths between the pair of nodes, v_i and v_j. For simplicity, we will use P to denote for the set of all valid paths between $START$ and END nodes and P^K to denote the K-shortest paths between $START$ and END nodes. The concatenation of two paths $p_{x,y}$ and $p_{w,z}$ is denoted by $p_{x,y} \diamond p_{w,z}$ and forms the path from node v_x to node $v_z, p_{x,z}$. For any node $v \in V$, let $H(v)$ denotes the set of edges whose head nodes is v.

In the problem of K-shortest path, for a given graph G_{TOPO} and the two given nodes, $START$ and END, the goal is to find the set P^K such that:

1. $c(p^k) \leq c(p^{k+1}), 1 \leq k < K$.
2. $c(p^K) < c(q), q \in \mathrm{P} - \mathrm{P}^K$.
3. p^{k+1} is determined immediately after p^k.

In the proposed algorithm, we define the k^{th} shortest path as $p^k = < START = v_0^k, v_1^k, \dots, v_{N+1}^k = END >, k \leq K$. We define the set of columns in G_{TOPO} that being visited by a path p by $U(p)$.

4.6.3.2 The Reverse-Pseudo-Tree

Our implementation of K-shortest paths is based on a generalization of Yen's algorithm (a deviation algorithm) [204], an alternative with improved expected running time. We replaced the generic shortest path algorithm used (i.e., Dijkstra) with our constrained shortest path algorithm that was described in the previous Section 4.6.2. Most of the deviation algorithms known in the literature that find the K-shortest paths between a pair of nodes are based on the construction of what is called a "*pseudo-tree*". The *pseudo-tree* contains repeated nodes in which makes it not a tree as it is defined. However, because the same repeated node belongs to two different paths in the tree, all nodes are considered different in the tree. In K-shortest paths problem, the K-shortest paths form a tree. Therefore, the goal of any path ranking algorithm is to build the tree of K-shortest paths. To do so, a *pseudo-tree* is built and determined for top-K paths.

In the proposed algorithm in this work, we introduce the *"reverse-pseudo-tree"*, the tree of paths from $START$ node to END node. If a simple labeling algorithm is used on *"reverse-pseudo-tree"*, the direction of edges can be reversed, and the END node can be considered the root of the tree. Fig. 21C shows a *"reverse-pseudo-tree"* for the first three shortest paths for G_{TOPO} in Fig. 19A.

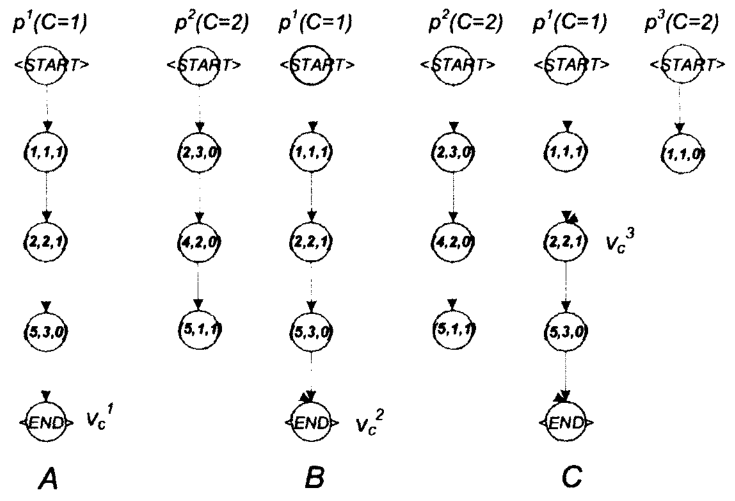


Figure 21. The formation of the "reverse-pseudo-tree" of P^3. (A) The shortest path. (B) The second shortest path coincides with p^1 at the END node. (C) The third shortest path coincides with p^1 at node <2,2,1> and with p^2 at END node. The coinciding node for each path is shown.

We can use algorithm 2 shown in Fig. 22 to build the corresponding *"reverse-pseudo-tree"*, denoted by $\mathcal{T}_K$, for K-shortest paths. When constructing the tree, the path $p^1 \in P$ simply forms a tree with only one path. To add the path $p^k \in P - \{p^1, p^2, \dots, p^{k-1}\}$ to the *"reverse-pseudo-tree"* of $k-1$ paths $\mathcal{T}_{k-1}$, the branch p^k_{START,v_k} denotes the subpath from $START$ node to the node v_k is added. For example, Fig. 21 shows the *"reverse-pseudo-tree"*, $\mathcal{T}_3$, built for the topology graph, G_{TOPO}, in Fig. 19A. The tree can be built by three consecutive calls to algorithm 2. The initial tree should be sent is an empty tree. To build $\mathcal{T}_1$, $\mathcal{T}_0 = \emptyset$ is sent. The value of other parameters at the end of executing the algorithm as follow: $edge = (5,3,0)(END), tail = < 5,3,0 >$, $p = << 5,3,0 >, END >, v_c^1 = END, p_{START,v_1}^1 = p^1$. The piece code of the loop will not be executed since $p \notin \mathcal{T}_0$. See Fig. 21*A*. To build $\mathcal{T}_2$, $\mathcal{T}_1$ should be sent to the algorithm. The *"reverse-pseudo-tree"* $\mathcal{T}_1$ contains one path, p^1. The values obtained for different parameters as follow: $edge = (5,1,1)(END), tail = < 5,1,1 >$, $p = << 5,1,1 >, END >, v_c^2 = END, p_{START,v_2}^2 = p^2$. The loop will not be executed in this call as well because $p \notin \mathcal{T}_1$. See Fig. 21*B*. For the last call to add p^3, the values are: $edge = (5,3,0)(END), tail = < 5,3,0 >$, $p = << 5,3,0 >, END >$. Since $p \in \mathcal{T}_2$ this time the loop should be executed. Then $edge = (2,2,1)(5,3,0), tail = < 2,2,1 >, and\ p = << 2,2,1 >, END >$. Again $p \in \mathcal{T}_2$ and the loop should be executed for the second time until we obtain values: $edge = (1,1,0)(2,2,1), tail = < 1,1,0 >, and\ p = << 1,1,0 >, END >$. Now $p \notin \mathcal{T}_2$, then the inner loop should not continue. The values of other parameters will be as follow: $v_c^3 =< 2,2,1 >, p_{START,v_3}^3 =< START, < 1,1,0 >, < 2,2,1 > >$. See Fig. 21*C*.

The node v_c^k determined by algorithm 2 is called the coinciding node. The path $p^k \in \mathrm{P} - \mathrm{P}^{k-1}$ coincides with each path $p \in \{p^1, p^2, \dots, p^{k-1}\}$ at some node and never deviates. The closest node to $START$ node among other nodes where p^k coincides with each path $p \in \{p^1, p^2, \dots, p^{k-1}\}$ is called the coinciding node. Therefore, the subpath from coinciding node to END node, $p_{v_c^k,END}^k$, is the longest subpath, in terms of number of nodes, that the path p^k could share with any path $p \in \{p^1, p^2, \dots, p^{k-1}\}$. For example, the initial coinciding node for p^1 in the example discussed in Fig. 21 is END node. p^2 coincides with p^1 at the last node, thus, the coinciding node for p^2 is END node. Finally, p^3 coincides with p^2 at END node and with p^1 from END node back until node $< 2,2,1 >$, thus, the coinciding node for p^3 is $< 2,2,1 >$.

Notation:

- Let p^k is the k^{th} shortest path.
- $\mathcal{T}_k$ is the reverse-pseudo tree of the k^{th} shortest path.

Algorithm 2 (TreeConstruction):

Constructing the reverse-pseudo-tree for the k^{th} shortest path.

```
input: $p^k, \mathcal{T}_{(k-1)}$
ouput: $\mathcal{T}_k$, and $v_c^k$

edge ← last edge of $p^k$
tail ← tail node of edge
$p$ ← subpath of $p^k$ from tail to END
while $p \in \mathcal{T}_{(k-1)}$ do
    edge ← edge preceeding edge in $p^k$
    tail ← tail node of edge
    $p$ ← subpath of $p^k$ from tail to END
end
$v_c^k$ ← head node of edge
$p^k_{START,v_c^k}$ ← subpath of $p^k$ from START to $v_c^k$
$\mathcal{T}_k \leftarrow \mathcal{T}_{(k-1)} \cup p^k_{START,v_c^k}$        // $p_k = p^k_{START,v_c^k} \diamond p^k_{v_c^k,END}$ is a path of $\mathcal{T}_k$ now
```

Figure 22. The pseudo code of algorithm 2 (TreeConstruction). Algorithm 2 can be used to prove that the first K-shortest paths forms a reverse-pseudo-tree.

4.6.3.3 Finding the K-Shortest Paths

For the deviation algorithm be able to find the K-shortest paths, a set X of pairs is maintained. The pair of information we store in X are a candidate path and its corresponding coinciding node. The set X is initialized with the shortest path p^1 and its coinciding node v_c^1 (i.e., END). The shortest path can be determined by algorithm 1 introduced in the previous Section where $c(p^1) \leq c(q), q \in \mathrm{P} - \{p^1\}$. At each iteration k to find path p^k, the pair of lowest cost path in X is picked up and new candidate paths are generated and re-stored in X. The process will repeatedly continue until the K-shortest paths are determined. Moreover, the process of generating new candidates includes the process of deleting some edges in order to prevent generating duplicates paths.

Notation:

- Let $p^k = < START = v_0^k, v_1^k, \ldots, v_{N+1}^k = END >$ is the k^{th} shortest path.
- X : A set contains candidate paths for K-shortest paths and their coinciding nodes.
- T : A list of shortest paths calculated so far.
- $p_{i,j}^k$: The path from node v_i^k to node v_j^k in the k^{th} shortest path.
- $U(p_{i,j}^k)$: The subset of columns visited in the path $p_{i,j}^k$.
- $H_{\mathcal{T}_k}(v)$: The set of edges whose head node is v.

Algorithm 3 (TopologiesEnumerationK): Finding K-shortest paths

```
$input$: $G_{TOPO}, K$
$output$: $T(the\ list\ of\ K - Shortest\ paths)$

$S \leftarrow \{1,2,3 \ldots, N\}$
$T \leftarrow \emptyset$
$k \leftarrow 1$
$p \leftarrow shortest\ path\ in\ G_{TOPO}$
$v_c^k \leftarrow END$
$X \leftarrow \{(p, v_c^k)\}$
while $(X \neq \emptyset\ and\ k \leq K)$ do
    $p^k \leftarrow shortest\ path\ in\ X$
    $X \leftarrow X - \{(p^k, v_c^k)\}$
    $T \leftarrow T \cup \{p^k\}$
    remove edges $H_{\mathcal{T}_{(k-1)}}(v_c^k)$ from $G_{TOPO}$
    $j \leftarrow 1$
    for each $v_i^k \in \{v_2^k, v_3^k, \ldots, v_c^k\}$
        remove edge $e_{(v_{i-1}^k, v_i^k)}$
        $U' \leftarrow S \backslash U(p_{i,N}^k)$
        $p_{START,i} \leftarrow$ the shortest path from START to $v_i^k$ over the set of coloums in $U'$
                        /*The path verifies $\min_{v' \in V}\left(f(v', U') + w(v', v_i^k)\right)$ */
        $q_{jpk} \leftarrow p_{START,i} \diamond p_{v_{i+1}^k, END}$
        $v_c^{q_{jpk}} \leftarrow v_i^k$
        $X \leftarrow X \cup \{(q_{jpk}, v_c^{q_{jpk}})\}$
        $j \leftarrow j + 1$
    endfor
    restore deleted edges to $G_{TOPO}$
    $k \leftarrow k + 1$
endWhile
```

Figure 23. The pseudo code of algorithm 3 (TopologiesEnumerationK). Algorithm 3 is used to find the K-shortest paths of the topology graph.

At each iteration k, to find path p^k, the pair element of shortest path in X is picked. The path is selected as path p^k and new candidate paths are generated. To generate new candidates for path $p^l, c(p^l) > c(p^k), l > k$, every node of p^k is analyzed from coinciding node v_c^k until the node v_2^k. For each node $v_i^k \in p_{2,v_c^k}^k$, starting from v_c^k, a specific shortest path $q \in \mathrm{P} - \{\mathcal{T}_k U X\}$ has to be computed. The shortest path q is actually the concatenation of the shortest subpath from $START$ to v_i^k, $p_{START,i}$, and $p_{i,END}^k$, where the edge $(v_{(i-1)}^k, v_i^k)$ is not in $p_{START,i}$ and $p_{START,i}$ has not been generated. In order to ensure that candidate path $p_{START,i}$ has not been generated before, the G_{TOPO} graph should be modified by removing some edges. Let $H_{T_{(k-1)}}(v_c^k)$ be the set of edges whose head nodes is the coinciding node v_c^k. Therefore, the set of edges in $H_{T_{(k-1)}}(v_c^k)$ and the edge $(v_{(i-1)}^k, v_i^k)$ should be removed from G_{TOPO}. Finally the set of edges deleted should be re-stored after all candidate paths have been generated from path p^k.

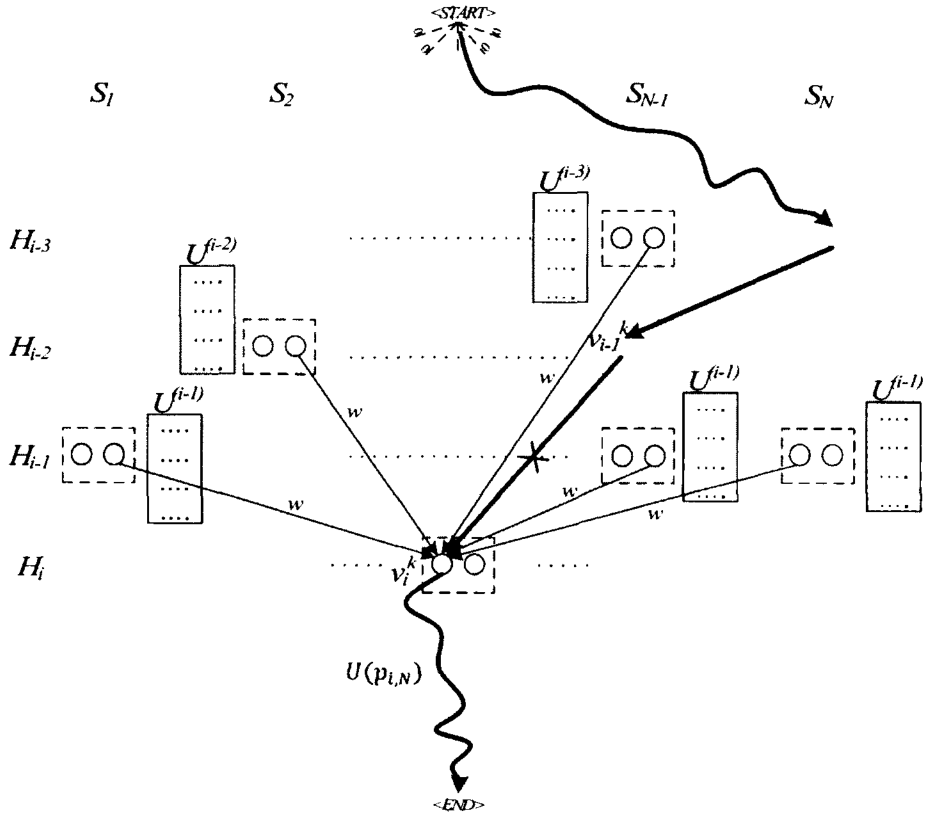


Figure 24. The process of finding the shortest path to node v_i^k by TopoDP. The process is to find the minimum cost of a path that visits the set of columns in $S - U(p_{i,N})$. The path p^k is the path colored in green.

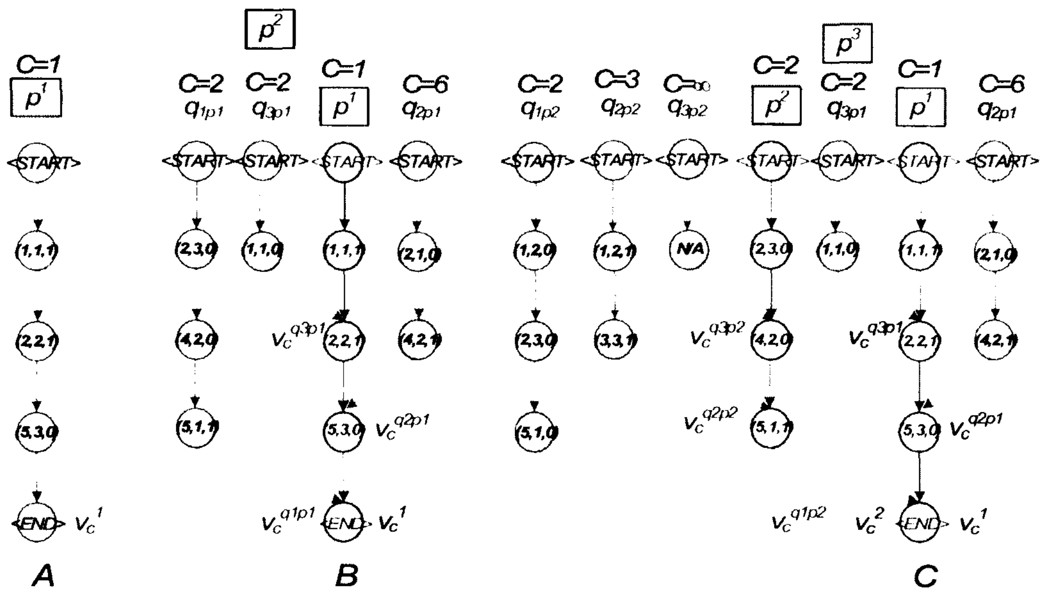


Figure 25. The first two iterations to find P^3 for the topology graph in Fig. 19A. The set of paths in $\mathcal{T}$ and X are shown. Paths in red are those paths will be saved in $\mathcal{T}$, paths in black are paths will be stored in X. (A) The shortest path in the graph. The coinciding node is initialized to END node. (B) Shows the candidate paths generated from p^1. The coinciding node is also shown. (C) The generated candidate paths from p^1 and p^2.

The constraint of the graph (i.e., no column may appear twice in the path) should be maintained throughout the process of generating candidate paths. Thus, at each iteration k and for each node $v_i^k \in p_{2,v_c^k}^k$, the shortest subpath $p_{START,i}$ being calculated should maintain the constraint. Consequently, the subset of columns being visited in $p_{START,i}$ should not contain any element of $U(p_{i,N}^k), U(p_{1,i}^k) = \{1,2,\dots,N\} - U(p_{(i+1),N}^k)$. The algorithm simply uses the subsets and their costs calculated in algorithm 1 to find $p_{START,i}$. In other words, the algorithm searches for the intended path in a set of nodes that are tails of edges in $H_{\mathcal{T}_k}(v_i^k)$. Any node v in this set that verifies $min(f(v, U(p_{START,i}^k)) + w(v, v_i^k))$ is chosen and the path from $START$ to v is picked as the shortest subpath. Note that the set of edges

$H_{\mathcal{T}(k-1)}(v_c^k)$ *and* $(v_{(i-1)}^k, v_i^k)$ are deleted at this time. The pseudo code of the algorithm used to find the K-shortest paths is illustrated in Fig. 23. Fig. 24 illustrates the process of finding the shortest path from $START$ node to node v_i^k.

In Fig. 25, we show the *"reverse-pseudo-tree"* of $X \cup \mathcal{T}_k$, $k < 3$, in the first two iterations of algorithm 3. In iteration number one, $\mathcal{T}_1$ has only path p^1, and the three paths $\{q_{1p1}, q_{2p1}, q_{3p1}\}$ will be generated and added to X starting from the coinciding node $v_c^1 = END\ to\ v_2^1 =< 2,2,1 >$. See Fig. 25*A*. At iteration number two, path $p^2 = q_{1p1}$ will be picked as the shortest path in X where $c(q_{1p1}) = 2$. Starting from the coinciding node $v_c^2 = END$ back to $v_2^2 =< 4,2,0 >$, three paths are generated and added to X, then $X = \{q_{2p1}, q_{3p1}, q_{1p2}, q_{2p2}, q_{3p2}\}$. See Fig. 25*B*. At iteration number three (not shown in Fig. 25), the next shortest path p^3 is either q_{3p1} or q_{1p2}, and their coinciding nodes are $v_c^3 =< 2,2,1 >$ and $v_c^3 = END$ respectively.

4.6.4 Results

To investigate the performance of the dynamic programming approach, we first tested if the shortest valid path identified by our method is indeed the valid path with the minimum cost. We sorted the cost of all the valid paths and found that the shortest path identified by the dynamic programming approach is indeed the top-1 ranked (Table 5, column 5) among all the valid paths. Note that the shortest valid path may not be the path of the true topology, due to the potential error in the weight. Recall that the weight used so far to be used in the topology graph is simply the absolute difference between the length of the loop can be found between the two SSEs-S represented in the rows and the Euclidean distance between the end of the two SSEs-D represented by the two columns. However, we will explain a more advanced approach in Section 4.7 to update this weight. The new approach uses some features extracted from the volumetric density map.

We notice that the valid path of the true topology often has near minimum cost (to be discussed with Table 7). We then compared current approach with two previous approaches: the naïve and the depth first search. In the naïve approach, each of the entire $N!\,2^N$ topologies were evaluated to search for the one with the minimum cost. As expected, the naïve and the depth first search approaches took significantly longer time than the dynamic programming approach in the large proteins (Table 5). For example, the time to find the shortest valid path is 1,410.49 seconds (Table 5, row 8) for the naïve approach, 8.753 seconds for the depth first, and 0.014 seconds for the dynamic programming approach. In this case, the protein has nine actual helical SSEs-S on the sequence and seven helical SSEs-D sticks detected using HelixTracer. For this protein, our dynamic programming approach is 100,000 faster than the naïve method. It is expected that the difference in performance is even more for larger proteins. The time in Table 5 includes the time to build the graph and the search for the shortest path. All the tests in this work were run on a generic PC - Dell Optiplex 980 machine at 2.8 GHz and 8 GB of memory.

Table 5. The test of the shortest valid path using the three approaches.

Num.	Protein ID	#True helices[a]	#Sticks[b]	Rank shortest[c]	Dynamic[d]	Depth first[e]	Naive[f]
1	1SU0	5	3	1	0.002	0.014	0.125
2	1BO9	6	3	1	0.001	0.013	0.498
3	1JW2	4	4	1	0.001	0.008	0.209
4	1A7D	6	4	1	0.002	0.018	0.944
5	1AA2	7	4	1	0.009	0.068	1.268
6	1DUS	6	5	1	0.004	0.186	8.038
7	1FLP	7	6	1	0.010	1.224	9.452
8	1NG6	9	7	1	0.014	8.753	1,410.49

a: the number of helices in the native protein.
b: the number of helices detected by HelixTracer.
c: the rank of the shortest valid path using the dynamic programming approach.
d: the time (in seconds) to find the shortest valid path using the dynamic programming algorithm. It includes the time to build all subsets at each node.
e: the time (in seconds) to find the shortest valid path using the depth first method.
f: the time (in seconds) to find the shortest valid path using the naïve method.

Table 6 shows the performance and the memory usage for large proteins. We were not able to work with proteins with more than seven helices using the naïve approach due to the large number of topologies to be evaluated [52,183]. Now we are able to work with proteins with 33 actual helices on the protein sequence and 18 detected sticks (Table 6, row 15). For this protein, it took 66.07 seconds to build the graph, 0.57 seconds to find the shortest valid path and it used 1.11 GB memory. We also listed the time it takes to get the top 100 shortest paths (Table 6, column 7). Notice that the search time is generally much shorter than the time to build the graph. However, the graph is only needed to build once for the search of top-*K* paths. This makes our approach practically effective to obtain the top-ranked valid topologies. Although most proteins do not have as many as 33 helices, the total number of helices and β-strands can be over 20 in a medium sized protein.

Table 6. Run time and memory usage for the dynamic programming algorithm.

Num.	ProteinID	#Helices[a]	#Sticks[b]	BUILDtime[c]	1st time[d]	Top100[e]	Memory[f]
1	1B5L	6	5	0.008	0.000	0.005	0.16
2	1FLP	7	6	0.010	0.000	0.007	0.18
3	1NG6	9	7	0.014	0.000	0.011	0.25
4	1ZA0	13	8	0.291	0.001	0.039	0.72
5	2H7O	14	9	0.270	0.003	0.075	0.83
6	3ACW	17	10	1.200	0.001	0.481	2.68
7	3L9T	14	11	1.000	0.000	0.337	2.44
8	2XB5	13	8	0.276	0.000	0.065	0.64
9	3ODS	21	12	1.800	0.003	1.000	7.44
10	1A4S	20	12	1.400	0.003	1.200	8.61
11	2PFT	27	14	24.09	0.020	4.200	39.99
12	2X79	24	17	28.40	0.024	5.040	211.1
13	2OEV	26	18	35.02	0.028	6.050	531.0
14	2XVV	33	17	52.04	0.030	10.10	515.64
15	2XSI	33	18	66.07	0.570	11.06	1,110.3

a: the number of helices in the native protein.
b: the number of helices detected by HelixTracer.
c: the time (in seconds) to build all subsets in the graph.
d: the time (in seconds) to find the shortest valid path .
e: the time (in seconds) to find the shortest 100 paths .
f: the memory (in MB) to store all subsets and paths.

Table 7. The rank of the true topology among all valid topologies.

Num.	Protein ID	# Helices[a]	#Sticks[b]	Rank[c]
1	2PSR	5	4	16
2	1AEP	5	5	42
3	1B5L	6	5	34
4	1FLP	7	6	1
5	1BVC	8	4	50
6	1NG6	9	7	97

a: the number of helices in the native protein.

b: the number of helices detected by HelixTracer.

c: the rank of the true topology ranked by our top-K method.

Since we used a simple criterion to assign the weight for an edge, error is expected in the weights. We wanted to see if the true topology is near the top of the solution space using the current weighting strategy. We applied our top-*K* deviation algorithm to identify the top-*K* shortest paths and see where the true topology is ranked. We tested six proteins with less than eight sticks detected in the volumetric density map. The rank of the true topology is between one and 97 for these proteins. For the largest protein (1NG6, Table 7, row 6), the true topology is ranked the 97th out of $\binom{9}{7}7!\,2^7 \approx 23\,million$ possible topologies in the entire solution space. This suggests that the simple weight could be fairly effective in eliminating most of the possible topologies for these proteins. Note that the weight in our method employs minimal constraints. In fact, it does not involve sophisticated analysis of the volumetric density map and only reflects the fact that the end-to-end distance of the sticks is comparable to the length of the loop connecting them. It is expected that more accurate weights of the edge can improve the ranking of the true topology even more.

The current weight mainly represents the difference between the estimated length of the loop connecting the two sticks and the estimated distance between them in 3-D space. In order for the true topology to be ranked near the top of the list, the predicted helices based on the protein sequence have to be accurate enough. Two situations will affect the ranking the most. One is when a long helix is wrongly predicted as two short helices that are connected by a short loop. The other is when two short helices are predicted as one long helix. In order to identify the true topology, the search needs to consider the possible errors from both the secondary structure prediction and those from the detection in 3-D space.

The current implementation is limited in the ranking of the helices, although our dynamic programming approach is general for either helices or β-strands. In order to extend the current approach to β-strands, the location of the β-strands needs to be estimated. The β-sheets are generally not as accurately detected as the helices in the intermediate resolution volumetric density maps. It is expected that the β-strands will be estimated with many possible alternatives. It is still a challenging problem to rank the topology with both α-helices and β-sheets without the knowledge of a template.

4.7 UPDATE TOPOLOGY GRAPH USING VOLUMETRIC MAPS

The topology graph was initially built with two kinds of edges, special and non-special edges. For special edges, edges connect the two special nodes $START$ and END with other nodes; the weight was initialized, and should always remain, to zero. On the other hand, for non-special edges, the weights of edges connect nodes represent secondary structure elements, were initialized to either ∞ or w_{init}. Non-special edges that do not satisfy constraints in Section 4.2 are initialized to ∞. Non-special edges satisfy constraints were initialized to w_{init}. In this Section, we explain how we use CryoEM volumetric maps to update the weight of non-special edges that satisfy the constraint.

Electron CryoEM is an attractive advanced image processing method for structure determination. Unlike experimental methods such as X-ray crystallography, CryoEM is able to produce volumetric maps of proteins that are poorly soluble, large and/or hard to crystallize. Furthermore, it studies the proteins in their native environment. Unfortunately, the volumetric maps generated by current advances in CryoEM technique produces protein maps at about 5-10Å resolution in which it is unable to determine the atomic-structure of the protein. However, some features of the protein can be visually and computationally identified such as the location of secondary structures and some connections between them. Therefore, recent work has shown the ability of CryoEM volumetric maps to help in *de novo* modeling of protein structures.

In recent work of Abeysinghe *et al.* [163], they have used a sort of thinning and pruning algorithms to produce a skeleton for the CryoEM volumetric map (Fig. 26A). Gorgon [187] is one tool, and the only tool up to our knowledge, that generates the skeleton of the CryoEM volumetric map and uses it to predict the topology of the secondary structures. The skeleton obtained is used to extract the geometric features from the volumetric map and to guide the process of topology prediction as well. In Gorgon, the topology problem is represented as a subgraph-isomorphism between the sequence (1-D) and the CryoEM volumetric map (3-D). The two shapes were modeled as attributed relational graphs. A constrained inexact graph matching problem has been solved by a heuristic search. On some of proteins, will be shown in the comparison Section below, Gorgon fails to find the correct correspondence. One problem that prevents Gorgon from finding the correct topology is the gaps found in the skeleton. In Gorgon, an edge drawn between the two secondary structure ends in density graph if and only if there is a trace on the skeleton that connects these two ends, or the Euclidean distance between the two ends is less than a threshold $\epsilon = 0.15d$ where d is the size of the volumetric. In the presence of these gaps, Gorgon cannot find the trace resulting in wrong topology prediction. However, a manual sketching can be used to avoid such a problem in a method depends completely on the user to locate the traces of the skeleton.

As reported, even if the problem of gaps is not considered, the method suffers some limitations [121]. The most important limitation is the computational cost (i.e., time and memory usage). Thus, due to memory limitation the method was unable to work on proteins have more than 25 helices without a large number of user-specified constraints [121]. Furthermore, on low resolution CryoEM volumetric maps, the success rates of the method are low due to the pad quality of geometry skeleton obtained. Fig. 26*A* shows an example of a skeleton produced by Gorgon for the real CryoEM volumetric map at 6.8Å resolution (EMDB ID 5100 [207]) and the corresponding Protein *"Scorpion Hemocyanin resting state"* (PDB ID 3IXV). The blue line represents the skeleton produced by Gorgon and red cylinders represent the detected secondary structures from CryoEM volumetric map. The black box shows one region where Gorgon fails to find the continuous skeleton.

In this work, we use the skeleton produced by Gorgon to update the weights of non-special edges in topology graph. To resolve the problem of gaps in the skeleton and to avoid the manual sketching, we developed an automatic algorithm to find loop traces between ends of secondary structures on CryoEM volumetric maps with existence of gaps.

Gorgon produces the skeleton that is represented as a list of voxel points. Each voxel point has a coordinate value for the location of the voxel at the CryoEM volumetric map. The main idea of the introduced algorithm is to translate each voxel point at the skeleton to a node in undirected graph. The voxel points (nodes) at the end of secondary structures are also marked. The edges between any two nodes in the undirected graph depend on the distance between the two original voxel points. If the distance is less than 3.0Å, the two nodes are considered neighbors and an edge created to connect them. The weight of the edge is equal to the distance between the two corresponding voxel points. Bron-Kerbosch algorithm [208] was applied to the graph to find the cliques of at least of size three. The purpose of finding the cliques is to find the crowded regions on the graph. The set of nodes represents the clique found are replaced with one central node (geometrical central of all voxels form the clique). The depth first search (DFS) is used to find the paths between every two nodes marked as SSE ends, called complete paths. Moreover, it is used to find the incomplete paths between every node marked as SSE end with any other node in the graph that represents a dead end. For example, in Fig. 26*B*, there are three complete paths from node P and one incomplete path $< P, R, S >$. All paths found for each SSE end is saved in a list $endList_j^t$, where $t \in \{0,1\}$ and $1 \leq j \leq N$. The variable t represents which end on the stick the paths starts from. The length of each path is simply the summation of weights of edges along the path.

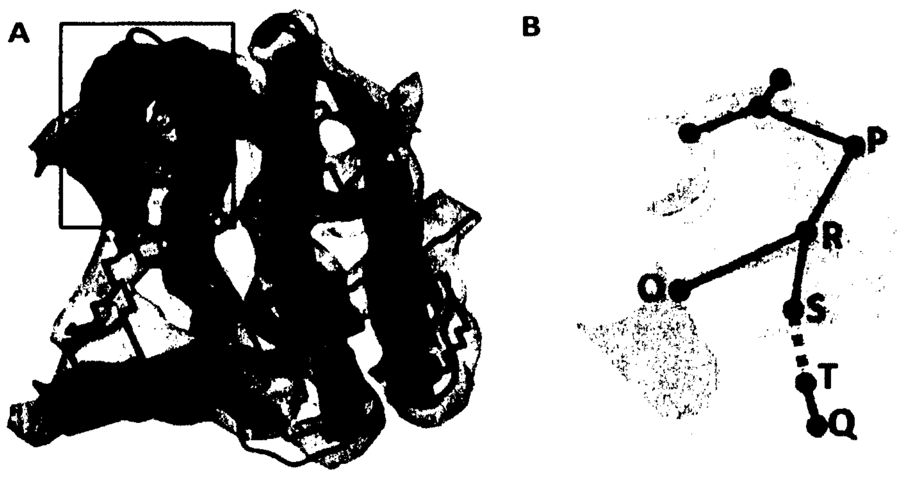


Figure 26. The skeleton detected from the CryoEM maps using Gorgon v2.1. (A) Shows an example of the skeleton detected for Scorpion Hemocyanin resting state (PDB ID code: 3IXV) using Gorgon at 6.8 Å resolution. In (A) we can see the gap in the skeleton (inside the black box). The SSEs-D sticks are shown in red cylinders and the actual protein structure is shown as well. (B) Depicts the gap found in the skeleton and the some voxels found in the skeleton to represent nodes in the skeleton graph. The gap is shown in the dotted line connect S and T.

The updating process of edges weights takes place once all lists are built for all SSE ends. For each edge $((i,j,t),(i',j',t')) = w_{init}$ of G_{TOPO}, we find the complete path in $endList_j^{t^c}$ or the two incomplete paths in $endList_j^{t^c}$ and $endList_{j'}^{t'}$ that best fit the number of amino acids on the sequence between H_i and $H_{i'}$. t^c is the complement of t denotes the other end of the stick j. We simply search for a complete or incomplete path with a length that best fit the estimated length of the loop on the sequence. The estimated length of the loop on the sequence is calculated by multiplying the number of amino acids by 3.8. In both cases, complete or incomplete, the length of the path should not exceed the estimated length of the loop plus *e=5Å*. Verifying complete paths against loop is very simple. We trivially compare the two lengths. For incomplete paths, we try all combination of incomplete paths between the two lists that are at most 15Å apart in authentic maps or at most 10Å apart in synthesis maps. The length of the new path produced from the two incomplete paths is the summation of incomplete paths, one from each list, and the gap between them. For example, in Fig. 26*B*, the two incomplete paths $< P, R, S >$ and $< T, Q >$ form one complete path $< P, R, S, T, Q >$. The absolute difference between the length of the loop on the sequence and the best path from the list(s) is w_{trace}. If no proper path (complete or incomplete) was found on the CryoEM volumetric map, the w_{trace} is set to ∞. Finally, the new weight of the edge of G_{TOPO} is the minimum between w_{init} and w_{trace}.

4.7.1 TOPODP vs. Gorgon

To test the performance of the dynamic programming approach (TopoDP), we tested the algorithm against the current version of Gorgon (v 2.1-windows-32bit) [187]. In Gorgon, a heuristic algorithm is used to match between two graphs, one graph for SSEs-S (called sequence graph) and another one for SSEs-D (called volume graph). Each secondary structure (particularly helix) in volume graph is represented by two vertices with a link connects them. The links between two SSEs vertices are created based on skeleton. A link connects two SSEs vertices is created if a continuous trace can be found on skeleton that connects the two ends of corresponding SSEs-D, otherwise no connection is established. A Max Euclidian Loop Distance parameter (ϵ) can be set to create a link between any two SSEs vertices if the trace on skeleton is missing or non-continuous. Consequently, a link is established between any two SSEs vertices if the corresponding SSEs-D ends are (ϵ)Å or shorter apart. In the course of this experiment, parameter ϵ is set to be 15Å. Other than Max Euclidian Loop Distance, default parameters values are used.

In Gorgon, as well as in TopoDP, the quality of the skeleton plays a major role in the process of prediction. However, the negative impact of the skeleton on TopoDP is less (will be shown in Table 8). The way it deals with gap and the best match process implemented in TopoDP makes it robust to medium quality skeletons. Two types of skeletons were used when predict the topology, grayscale and binary skeletons. The method used to generate the binary skeleton is composed of two algorithms: iterative thinning and skeleton pruning [163]. On the other hand, the grayscale skeleton is generated by deploying a segmentation-free algorithm [209]. The algorithm mainly employs the idea of structure tensor in addition to feature extraction. In contrast to binary skeleton, grayscale skeleton does not suffer from threshold dependency and does not need a segmentation process. Thereupon, the produced skeleton is less biased to human intervention. In consequence, the quality of grayscale skeletons is enhanced in relative to binary skeletons. Binary and grayscale skeletons were generated for each protein in the data set.

The data set used in the experiment consists of 14 volumetric maps at 10Å resolution of which 12 were are simulated from actual protein models found in Protein Data Bank (PDB) and two were authentic CryoEM volumetric maps from the EMDB (EMDB ID 5100 and 5030 at 6.8Å and 6.4Å respectively). For both softwares, we have used the actual position of secondary structure on the sequence obtained from the model in PDB. However, a secondary structure prediction tool could be used, but we intended to avoid the negative impact of wrong secondary structure prediction on topology prediction. The helices on CryoEM volumetric maps were detected using SSETracer [124]. The correctness evaluation of the two methods was carried out by comparing the produced topologies with the correct topology of each protein obtained from PDB. All the tests in this dissertation were run on a generic PC - Dell Optiplex 980 machine at 2.8 GHz and 8 GB of memory.

The results of the 14 CryoEM volumetric maps are listed in Table 8. Table 8 shows the number of helices in each protein (SSE-S), the number of detected helices from CryoEM volumetric map (SSE-D), and the results of the two methods on the two types of skeletons. The memory usage monitored in Table 8 represents the memory needed to run the heuristic algorithm by Gorgon to rank the top 35 topologies and does not count the memory used to save graphs and data structures. In contrast, it represents the memory used to build Topo graph and its sets in addition to the data structure used to save traces found on skeleton. Furthermore, Table 8 shows the time needed to rank the topologies and not counts the time needed to build graphs for Gorgon. Nevertheless, it counts the time needed to build Topology graph and its sets for TopoDP.

Table 8 shows the performance and the memory usage for relatively large proteins. The comparison with Gorgon focuses on the memory needed, the time to accomplish the task and the accuracy of the two prediction algorithms. We were not able to work with proteins with more than seven helices in our earlier work [52,183] due to the large number of topologies to be evaluated. Now, for instance, we are able to work with proteins with 20 actual helices on the protein sequence and 20 detected sticks (3HJL, Table 8, row 9). For this protein, TopoDP takes 1.6 seconds to build the graph, trace the CryoEM volumetric map and to find the top 35 proteins. Additionally, it used 240.9 MB of memory to save all sets, all traces, and the 35-shortest paths. Notice that the search time for top 35 topologies is generally much shorter than the time to build the graph and to trace the CryoEM volumetric map. However, the graph is only needed to be built once for the search of top-K paths. This makes our approach particularly effective to obtain the top-ranked valid topologies. Although most proteins do not have as many as 20 helices, the total number of helices and β-strands can be over 20 in a medium sized protein. On the contrary, Gorgon has failed to find any correspondence for this protein using the two types of skeletons. Thus, the memory, time, and accuracy are not available.

As depicted in Table 8, Gorgon failed to predict the correct topology within the top 35 topologies in 12 cases when binary skeleton is used. Furthermore, it fails in 10 cases when a grayscale skeleton is used. This shows that grayscale skeleton is improved over the binary one with more continuous traces. The two cases in which Gorgon could successfully predict the correct topology are rows 2 and 4 in Table 8 (binary skeleton). The skeletons of the two proteins are either gap-free or the Euclidean distance between the two sticks' ends is shorter than ϵ. However, for some cases, even though the skeleton is gap-free, Gorgon failed to find the correspondent. It fails with most of relatively big proteins. Generally, the time and space Gorgon needs to carry out the prediction are greater than the amount of time and space TopoDP needs. The big difference in time and memory usage is clear in big proteins. Some entries in Table 8 show such big difference. For instance, the amount of memory used by Gorgon was 100 times larger than the amount of memory used by TopoDP for protein 3LTJ when binary skeleton is used (Table 8, row 4). Moreover, TopoDP is faster to accomplish the prediction process. This can be seen in the amount of time needed to rank the 35 topologies in 1Z1L. the amount of time used by TopoDP is 10 times faster than the amount of time needed by Gorgon (Table 8, row 6).

Gorgon is very sensitive to the quality of the skeleton used in the process. As mentioned earlier, bad skeletons negatively affects the quality of the prediction. When a low quality skeleton is used, more gaps expected to present. Unfortunately, Gorgon fails to predict the true topology successfully if a small gap present in the skeleton. The existence of a gap means that no continuous trace can be found between the two ends of SSEs-D. The problem is clear for the relatively small proteins. For instance, Gorgon fails to find the true topology of five proteins among the first seven proteins in Table 8. When an enhanced quality of skeleton used, the prediction is improved. This can be seen when a grayscale skeleton is used for the first seven proteins. Gorgon could predict the true topologies of four proteins out of the same seven proteins. However, this is not the only problem Gorgon suffers. On contrary, TopoDP has overcome the problem. In TopoDP, the gaps shorter than 10Å for synthesis volumetric maps and 15Å authentic maps are treated when incomplete paths are used to find the best fit trace (see Section 4.7 for more details). Thus, TopoDP could successfully predict all true topologies of the first seven proteins using both types of skeletons. However, the gap threshold was increased from 10 to 15Å for one of the proteins (3ACW, Table 8, row 5) when a grayscale skeleton is used.

Table 8. Improved accuracy, space and time for topology identification.

No.	ID[a]	#AA	#hlces[b]	#sticks[c]	Topo-DP				Gorgon			
					Binary		GrayScale		Binary		GrayScale	
					space/time[d]	Rank[e]	space/time[d]	Rank[e]	space/time[d]	Rank[e]	space/time[d]	Rank[e]
1	1FLP	142	7	7	0.004/<=2	1	0.004/<=2	1	0.41/<=2	N/A	0.64/<=2	1
2	1NG6	148	9	7	0.004/<=2	2	0.004/<=2	2	0.33/<=2	3	0.19/<=2	2
3	2XB5	207	13	9	0.03/<=2	11	0.01/<=2	1[d]	1.44/<=2	N/A	1.11/<=2	N/A
4	3LTJ	201	16	12	1.6/<=2	2	1.69/<=2	2	165.3/9.9	2	270.3/16.2	2
5	3ACW	293	17	14	9.7/<=2	32	9.75/<=2	34[d]	16.6/2.7	N/A	9.2/<=2	N/A
6	1Z1L	345	23	14	18.5/2.3	11	18.59/2.4	1	>1289.5/24	N/A	>934.6/42.6	N/A
7	3ODS	415	21	16	42.3/2.8	12	42.34/2.9	2	275.4/10.7	N/A	377.4/15.2	23
8	1HZ4	373	21	19	273.1/14.0	2	273.00/14.7	3	>959.3/68.9	N/A	458.8/40.3	N/A
9	3HJL	329	20	20	240.9/<=2	1	236.9/<=2	1	N/A	N/A	N/A	N/A
10	2OEV	705	26	20	1209.6/48.3	N/A	1209.5/74.4	N/A	>1288.0/34	N/A	>950.9/39.4	N/A
11	2XVV	585	33	20	1224.7/120	21	1225.1/126	4	>1315.6/25	N/A	>1312.9/276	N/A
12	2XSI	585	33	19	1381.9/141	13	1382.5/168	N/A	>1341.1/23	N/A	>1286.7/272	N/A
13	3IXV_A	222	14	10	0.21/<=2	1[d]	0.004/4.2	1[d]	116.8/5.8	N/A	>922.0/30.3	N/A
14	3FIN_R	117	4	4	0.004/<=2	1[d]	0.004/<=2	4[d]	0.55/<=2	N/A	0.48/<=2	N/A

a: the PDB ID of the protein.

b: the number of actual helices in the protein.

c: the number of detected helices from CryoEM map.

d: the space (in MB) and time (in Sec.) needed to rank top 35 topologies. The sign > means that the task could not be completed.

e: the rank of the true topology within top 35 topologies. N/A means the true topology could not be ranked within top 35 topologies.

In addition to the gap problem, Gorgon appears to be less capable in handling large proteins. Recall that the problem being addressed is exponential. The A-star heuristic search algorithm used by Gorgon is known to suffer from memory limitation.

Conversely, the dynamic approach used in TopoDP effectively saves the memory. Moreover, the performance of TopoDP can be improved (in terms of time and memory usage) by a factor of 10. Recall that links in Topology graph have no correspondent traces on skeleton are maintained with a weight equals to w_{init}. Therefore, deleting these links could save memory and time when build the sets dynamically. However, this step may negatively affect the accuracy of finding the true topology. For instance, this process was successfully applied to protein 2XSI (Table 8, row 12) and the time needed to find the 35-shortest path was 14 seconds. Again, the true topology was ranked 12^{th}. On the other hand, it failed to rank the true topology within the top 35 topologies when applied to protein 1HZ4 (Table 8, row 8). However, the time was improved by a factor of 10 to rank the top 35 topologies. Thus, a careful process on such links might be used to improve the performance of TopoDP and maintain the accuracy.

The current weight mainly represents the best trace (complete and incomplete paths) can be found for the loop between the two sticks detected from CryoEM volumetric map. In order for the true topology to be ranked near the top of the list, the predicted helices based on the protein sequence have to be accurate enough. Two situations will affect the ranking the most. One is when a long helix is wrongly predicted as two short helices that are connected by a short loop. The other is when two short helices are predicted as one long helix. In order to identify the true topology, the search needs to consider the possible errors from both the secondary structure prediction and those from the detection in 3-D space.

The current implementation only applies to the ranking of the helices, although our dynamic programming approach is general for either helices or β-strands. In order to extend the current approach to β-strands, the location of the β-strands needs to be estimated. The β-sheets are generally not as accurately detected as the helices in the intermediate resolution density maps. It is expected that the β-strands will be estimated with many possible alternatives. It is still a challenging problem to rank the topology with both α-helices and β-sheets without the knowledge of a template.

4.8 PARALLEL SOLUTION

In the work of the naïve approach [183], we were not able to enumerate and build the structure of the entire set of possible topologies for proteins have more than seven helices using parallel computers evaluate (i.e., build 500 full-atoms conformations for each candidate and ranking them according to the multi-well energy potential). Moreover, the total number of possible topologies is expected to increase very quickly as the number of helices increased.

In the work of the naïve (Section 4.3) and depth first search (Section 4.5) approaches, the work of parallel method used is depicted in Fig. 27. On this work we have used a simple dynamic master/slave scheme. The master processor was responsible of generating the initial structure of SSEs-D sticks and finding the valid topologies among all possible topologies by either the naïve or DFS approaches. A free slave processor asks the master for any available valid topology. The answer the slave processor is expecting is one of two, the next available valid topology or a flag to indicate that no more valid topologies left. When a slave processor receives a valid topology, it generates random 500 full-atoms structures by shifting each SSE-S up to two positions along the sequence and/or translate (i.e., move) each helical SSE-D stick one rise along its axis. After generating the initial conformation of the valid topology, the slave uses our own implementation of R3 algorithm [190] to add side chains. Finally, the multi-well energy [51] measures the stability of the structure is used to evaluate and rank the 500 structures. When an end-of-job flag received, the slave sends back the information of the best structures (i.e., have been sorted according to the contact energy) it has generated to the master processor. At the end of the job, and after all slave processors send their results to the master, it re-ranks them according to their energy values and pick up best structures to be topologies candidates.

For current approach, using topology graph, the time saving over the previous approach is significant. The time required to build the initial graph is $O(MN^2)$ and the time required to build all subsets to find the top-K paths is $O((D+1)^2N^22^N)$ which can be done using a single core processor for medium size proteins in comparison to the time needed to traverse a huge graph for big proteins. However, for larger proteins we still need to develop a parallel approach to traverse and enumerate top-K paths.

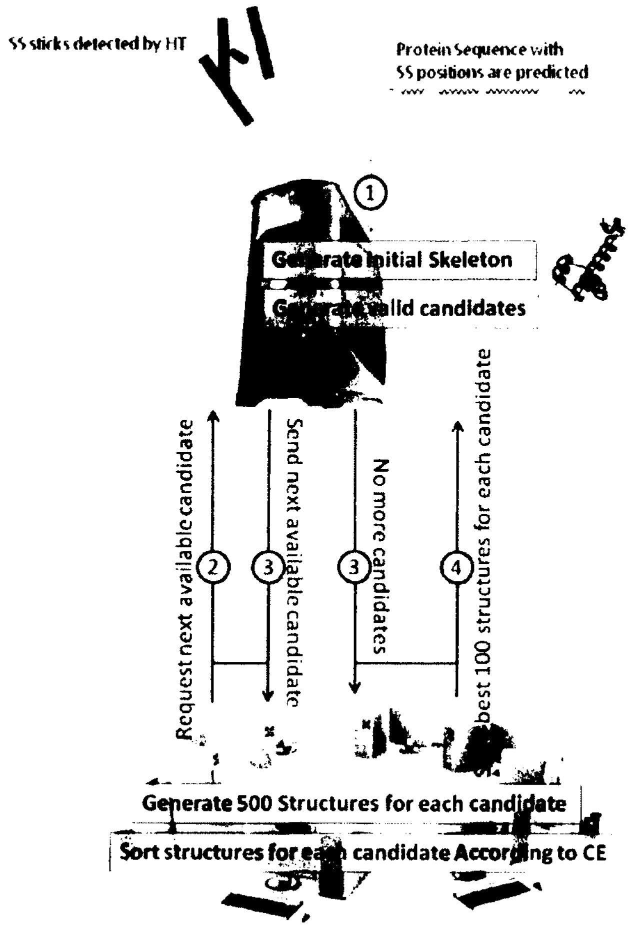

SS sticks detected by HT
Protein Sequence with SS positions are predicted
1
Generate Initial Skeleton
Generate valid candidates
Request next available candidate
2
Send next available candidate
3
No more candidates
3
best 100 structures for each candidate
4
Generate 500 Structures for each candidate
Sort structures for each candidate According to CE

Figure 27. The parallel approach to enumerate and evaluate all possible topologies.

One approach to parallelize graph traversing and valid paths enumerating is to distribute the load balance equally over the entire set of processors. In such a case, the speed up factor is maximized. One challenge of such an implementation is the difficulty to know in advance the number of valid paths outgoing from certain node.

CHAPTER 5

LOOP CLOSURE FOR LOOP MODELING

Loop closure problem arises in nearly all loop prediction problems. Loop closure is a problem of generating a loop whose N and C terminal residues satisfy the constrained locations predefined by the two ends of the chain to be connected by the loop (Fig. 28). The N-Anchor refers to the last (the C-terminal) residue of the first portion of the chain and the C-Anchor refers to the first (N-terminal) residue of the second portion of the chain (Fig. 28*B*). The position and orientation of these two anchors are expected to remain the same during the process of loop closure. This problem is encountered in the last component of proposed system when the loop regions need to be modeled to fill the gaps in the atomic-resolution structure for secondary structures generated in component number two.

To address the loop closure problem, both analytical methods and optimization methods have been proposed. Wedemeyer and Scheraga have solved the problem for three consecutive residues through spherical geometry and polynomial equations [210]. Other solutions to this problem can be found in [211,212,213]. Recent work has extended the solution to any three residues that may not be consecutives [214]. It has been proved that the loop closure problem with six degrees of freedom has at most 16 possible solutions, whereas the number of possible solutions is infinite for the problem with more than six degrees of freedom [215,216,217]. A sub-angstrom method was introduced to solve the problem analytically for loops of up to 12 residues [179].

The longest loop that has been constructed analytically has nine bonds of freedom using a geometrical screening through the solution space [218]. Optimization approach has been used for loops with more than six degrees of freedom. These methods search for an approximate solution by iteratively changing the backbone torsion angles until the desired distance that is between the end of the loop and the anchor is reached. Two such well-known methods include random tweak [173,174], and cyclic coordinate descent [219]. A self-organizing algorithm is used to generate clash-free loops of lengths between four and 12 residues [175]. The algorithm starts from random initial atomic coordinates followed by fast geometric matching of the conformationally rigid components of the constituent amino acids.

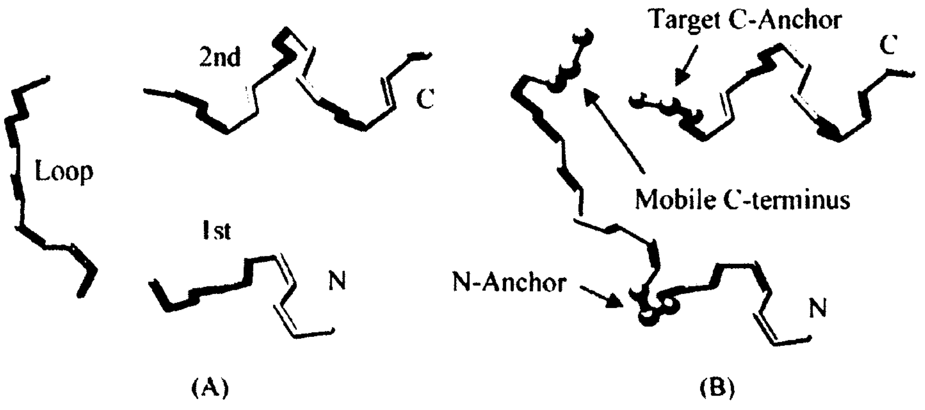


Figure 28. The loop closure problem. The 1st portion and the 2nd portion of the chain are to be connected by the loop (A). The *N*-Anchor and the C-Anchor amino acids, represented by their N, C_α and C backbone atoms (spheres in (B)), are expected to be fixed during the process of loop closure. The copy of the N-Anchor on the loop is superimposed with the N-Anchor of the 1st portion in (B). The mobile C-terminus that consists of a copy of the C-Anchor residue is expected to superimpose with the Target C-Anchor during the process of loop closure.

Cyclic coordinate descent (CCD) aims at closing the loop by adjusting a single torsion angle at a time. It applies the idea of inverse kinematics in robotics. In robotics, inverse kinematics algorithms were proposed to solve the problem of moving a robotic gripper to a specific position by changing joint angles and segment lengths [220]. A biological specialized inverse kinematics tool was initially designed as early as 1970 by Go and Scheraga [215]. Many exact inverse kinematics solvers have been proposed [210,214]. CCD is easy to implement and computationally inexpensive and is adopted by Rosetta docking and other variants such as FCCD [221,222]. For more literature details in this problem we refer the reader to our previous work on this problem [184,185] and [223].

5.1 INTRODUCTION

CCD is an iterative procedure to drive the *C* terminal end of the loop to the destination at the *C*-Anchor. The process stops when the maximum number of cycles is reached or when the loop is converged to the target. A loop is considered converged if RMSD between the N, C_α and C atoms of the moving C-terminus and their corresponding atoms at the C-anchor is within an accepted error [219]. For easy reference, let us call the RMSD the distance error from destination. The threshold of the distance error is 0.08Å in the CCD method. Once the random loops converge, the second portion of the chain is fairly accurately positioned. From our previous experience of an implementation of CCD [184], we observed that the C-terminus of the loop often reaches a neighborhood of the destination in a small number of iterations. However, it takes significantly more number of cycles to converge from the neighborhood to the destination. For example, for a loop of length four (1qnr, Table 9), it takes eight cycles for the mobile C-terminus of the loop to reach from 4.7Å to 1.1Å of distance error from the destination (data not shown). However, it takes 418 cycles to reach from 1.1Å to 0.08Å of distance error from destination.

Although the remaining distance error can be small (i.e., up to 0.08Å), the second portion of the chain has to be moved to connect to the loop in order to generate a continuous chain. In the above case, if the last 418 cycles are omitted, the second portion of the protein chain will not be accurately placed due to the remaining distance error at the end of the iterations. The proposed method aims at developing an effective loop closure method that is not completely dependent on the convergence of the loop. Instead of spending the majority cycles of CCD for the loop to converge, our method uses a small number of cycles to lock the moving end of the loop to a neighborhood of the target. It then directly adjusts the accuracy for the second portion of the chain using the iterations of backward walk. Our method generally needs smaller number of cycles to close gap than the original CCD method, yet produces more accurate second portion of the chain.

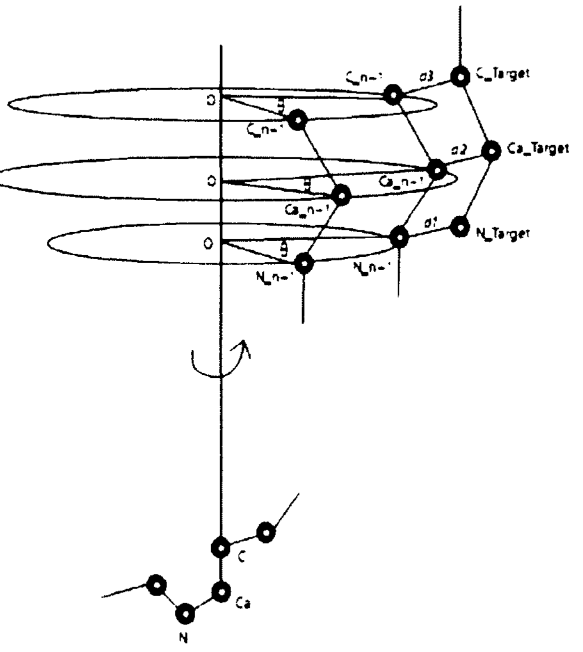


Figure 29. Torsion angle update. The N, C_α, and C atom of the C-Anchor are labelled as N_Target, C_α_Target, and C_Target respectively. The N, C_α, and C atom of the residue $n + 1$ on the loop are labelled as N_$n + 1$, C_α_$n + 1$, and C_$n + 1$ respectively. A torsion angle is updated with ϑ that minimizes the S in formula (1). The three distances in S, $d1$, $d2$ and $d3$, are labeled.

5.2 FORWARD-BACKWARD CYCLIC COORDINATE DESCENT (FBCCD)

FBCCD is composed of two major steps: the forward iterations and the backward iterations. The target in the forward walk is the C-Anchor residue represented by the

N, C_α, and C atoms. The target in the backward walk is composed of three points determined by the second portion of the protein. The idea is to use the forward walk to move the C-terminus of the loop quickly to the neighborhood of the destination and to use the backward walk to refine the loop based on the accuracy of the second portion of the chain.

5.2.1 Forward Walk

After the N-anchor and C-anchor are added to the initial loop, the forward walk starts by overlapping the N-anchor on the loop with the N-anchor at the first portion of the chain (Fig. 28*B*). In each cycle of the forward walk, the torsion angles of the loop are adjusted sequentially from the ψ angle of residue 0 till the φ angle of residue n + 1. The way of updating a torsion angle is the same as used in CCD [219]. Briefly, each torsion angle is updated so that the sum of squared distances, S in equation (1), is minimized (Fig. 29).

$$S = d_1^2 + d_2^2 + d_3^2 \dots\dots\dots\dots\dots (1)$$

Where d_1, d_2, and d_3 are the distances between the moving C-terminus and the C- anchor for the N, C_α and C atom respectively (Fig. 29). More details about the calculation of the update can be found in [185,219].

Each cycle in the forward walk starts from the N terminal of the loop. The forward walk stops either when the moving C-terminus converges to the fixed C-anchor or the maximum number of cycles is reached. A loop is considered converged if the distance error from the destination is within 0.08Å [219]. The maximum number of iterations used for the results to be shown in Table 9 is 100 and 2000 for the results in Table 10. In order to explore the possibility of improving the accuracy and convergence rate, two versions of the forward walk were implemented. The first implementation uses the concept of greedy in selecting torsion angles for adjustment during the initial cycles of the walk. In each greedy cycle, only one torsion angle is adjusted instead of all the torsion angles that are adjusted in a non-greedy cycle. The torsion angle that can move the mobile C-terminus closest to the fixed C-Anchor is chosen from all the torsion angles on the loop. In the greedy version of the implementation, the first 10 cycles were greedy and the rest 20 forward cycles were non-greedy for the results in Table 9. For the results in Table 10, the first ten cycles were greedy, and the rest 790 forward cycles were non-greedy. The second version of implementation does not use greedy cycles, and is as described in the original CCD method. Although the comparison results between the greedy and non-greedy versions are not included in this work, we find that the greedy version does not show significant advantage.

5.2.2 Backward Walk

Depending on the maximum number of iterations used in the forward walk, the C-terminus of the loop may or may not converge to the C-Anchor. Even for a converged loop, there is a gap of up to 0.08Å distance error between the C-terminus of the loop and the C-Anchor at the end of the forward walk.

After the forward walk finishes, the backward walk starts. To generate a continuous chain, the backward walk begins by connecting the second portion of the chain to the loop. Recall that residue $n + 1$ on the loop is a copy of the C-Anchor, the amount of translation and rotation for the second portion of the chain can be determined to superimpose the two amino acids, represented by the N, C_α and C atoms. The backward walk modifies the torsion angles sequentially starting from residue $n + 1$. The target of the backward walk involves three points determined by the second portion of the chain. The three points of the target include the two distal ends of the central axis of the helix and the last C atom of the helix (Fig. 30). The central helix axis can be approximated by connecting the two geometrical centers, one calculated from the first three C_α atoms and the other calculated from the last three C_α atoms of the helix. Although this is a rough estimation of the central axis of a helix, it does not seem to affect the accuracy of our method, since any three points from the second portion of the chain may be used. Each torsion angle is updated so that the distance between the three movable points and the three target points is minimized. The backward walk stops either when the maximum number of iterations is reached or when the RMSD is <0.001Å.

Similarly to the forward walk, backward walk also implemented two options. One option is to have a small number of greedy cycles for the initial iterations. The other option is not to use greedy cycles during the backward walk.

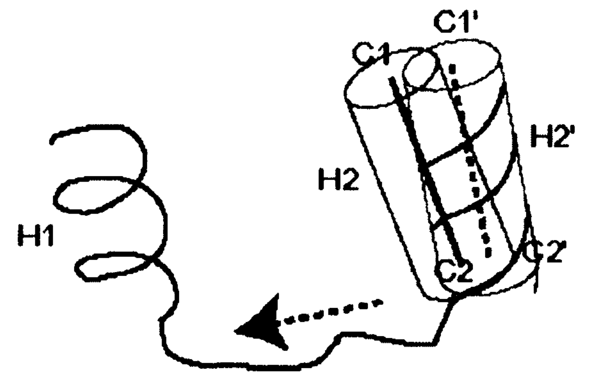


Figure 30. Backward walk of FBCCD. The torsion angles on the loop are updated from the last to the first (direction indicated with an arrow). *H1* and *H2* represent two helices, *H1* represents the first portion of the chain and is fixed during the process of loop closure. *H2* immediately follows the loop and is affected by the remaining distance error at the end of the forward walk. The three target points of the backward walk include the two distal ends (*C1* and *C2*) of the central axis of *H2* and the last C atom of *H2*. *C1*, *C2*, and *C* are moved to *C1'*, *C2'* and *C'* respectively, when *H2* is connected to the loop. *d1* is the distance between C1 and *C1'*. *d2* is the distance between C2 and *C2'*. *d3* is the distance between C and *C'*, the last *C* atom on the shifted helix2.

5.2.3 Implementation of CCD

To compare FBCCD with CCD, we implemented the CCD method according to the details in the CCD paper [219]. The differences between our implementation and the one in the CCD paper mostly lie in the initialization of the random loop, the maximum cycles allowed and the number of random loops generated for each tested loop. The CCD method in the paper uses the random torsion angles from the existing structures of the PDB to build an initial random loop.

Our implementation of the CCD uses random torsion angles from a feasible range ($\phi \in [-175,-40]$ and $\psi \in [-60,175]$). The different sources of random torsion angles should not affect the comparison result between the FBCCD and CCD, since FBCCD uses the same feasible range as the initial torsion angles. The way to update the torsion angles, the threshold for convergence of the loop are the same as in the CCD paper and the same for the forward walk of the FBCCD.

The maximum number of cycles in our implementation of the CCD is different from that was used in the CCD paper. The maximum number of the iterations is 5000 in the CCD paper. Since this work explores for a faster method, two cases of maximum iterations (100 and 2000) were used for CCD in order to compare with FBCCD in the same situation. The number of random loops generated for each tested loop is 300 in our implementation of CCD. The same number was used for FBCCD. The number of random loops was 100 and 5000 in two tests respectively in the CCD paper.

5.3 FBCCD EVALUATION AND RESULTS

Ten loops of length four, eight and 12 respectively were randomly selected from the data set of two papers [219,224]. For each loop tested, 300 random loops were generated initially using random angles of ϕ within [-175,-40] and ψ within [-60,175]. The torsion angles on the loop are iteratively modified using the forward and backward walk. The forward walk stops when the maximum number of iterations is reached or when the loop converges. For our implementation of CCD as well as the forward walk, a loop is converged if the distance error from the destination is less than 0.08Å. The destination for the forward walk in FBCCD and for the CCD is the C-Anchor residue. The destination for the backward walk includes *C1*, *C2* and *C*, a set of three points on the second portion of the chain (Fig. 30). The backward walk stops when the maximum number of iterations is reached or the distance error from the destination is less than 0.001Å.

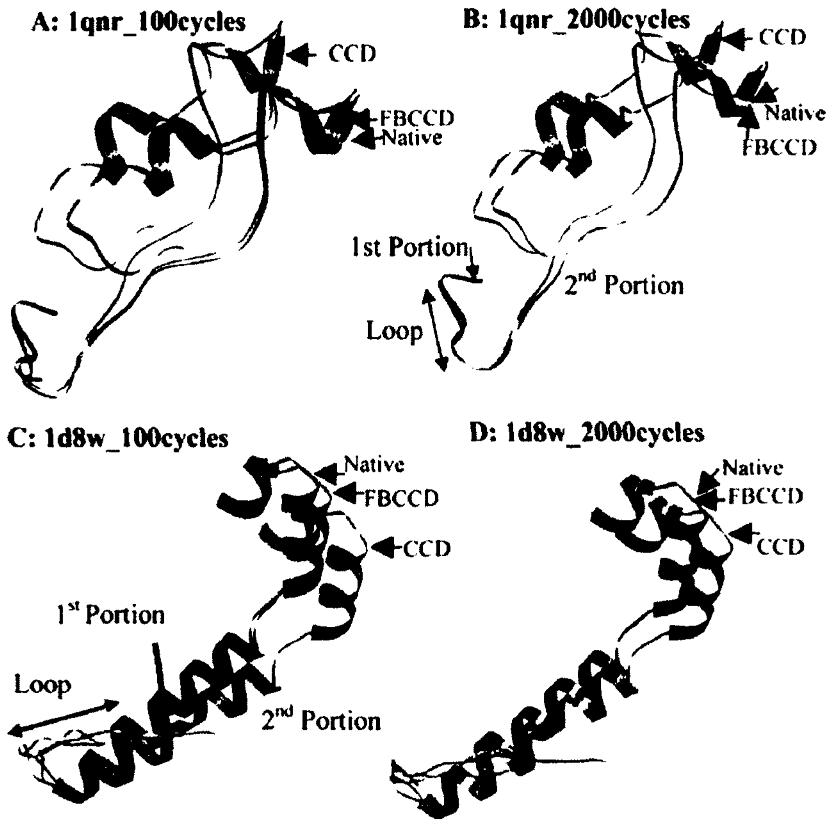


Figure 31. Structure fragments after 100 and 2000 iterations using FBCCD and CCD. The PDB ID code for the tested fragments is 1qnr in (A) and (B) and 1d8w in (C) and (D). The native fragment (in red) is superimposed on the fragment with the best loop (see also Tables 9 and 10) obtained using FBCCD (in cyan) and the fragment with the best loop obtained using CCD (in purple) after 100 iterations (A) and after 2000 cycles in (B). Similar superposition is shown for 1d8w fragments in (C) and (D). The first portion, the second portion of the fragment and the loop are indicated with arrows.

Table 9. Comparison between FBCCD and CCD after 100 cycles.

	CCD					FBCCD with greedy				
loop	Loop Best[a]	2nd portion[b]	Loop Avg[c]	Loop Max[d]	Conv.[e]	Loop Best[a]	2nd portion[b]	Loop Avg[c]	Loop Max[d]	Conv.[e]
					Length 4					
1i0hA_123-126	1.16	5.62	3.44	7.97	6%	1.14	0.02	2.94	7.162	0%
1qnrA_195-198	0.81	5.99	3.31	5.89	3%	0.49	0.74	3.31	7.721	0%
1tca_95-98	1.94	4.6	3.39	5.29	2%	1.46	1.19	3.04	4.899	0%
1ej0A_74-77	1.84	10.8	3.46	5.20	2%	0.93	2.19	3.39	5.844	0%
1fkf_42-45	0.80	2.59	3.42	5.69	5%	0.44	0.34	3.12	6.140	0%
1qopA_44-47	1.30	2.11	2.84	4.80	1%	0.37	0.26	2.85	5.337	0%
1aaj_82-85	1.27	14.9	3.42	5.31	28%	1.33	0.46	3.97	6.503	0%
1ads_99-102	1.50	3.39	3.14	5.22	12%	0.52	2.06	3.46	8.135	0%
1cbs_21-24	0.90	4.05	2.43	5.14	12%	0.54	0.17	2.42	5.306	0%
1nfp_37-40	0.32	2.21	2.14	4.36	8%	0.99	0.23	2.70	4.928	0%
Average	1.18	5.62	3.10	5.49	8%	0.82	0.77	3.12	6.197	0%
					Length 8					
1cruA_85-92	2.16	4.43	6.42	11.7	59%	3.16	0.41	8.33	13.91	1%
1ctqA_144-151	2.13	3.21	6.45	11.1	70%	2.32	0.39	6.35	11.52	0%
1i0hA_145-152	2.88	15.3	5.37	10.0	19%	2.92	2.14	5.25	10.50	1%
1gk8A_122-129	1.66	13.1	4.70	7.40	19%	1.91	0.50	4.79	7.770	0%
1ixh_106-113	2.05	13.8	5.46	9.62	20%	2.18	0.42	5.50	10.46	0%
1d8wA_334-	2.37	7.81	5.94	10.5	5%	2.58	0.80	7.45	12.16	0%
1ds1A_20-27	1.69	6.22	5.12	9.62	51%	1.49	3.41	5.29	10.62	1%
1cbs_55-62	3.22	8.59	7.79	10.7	30%	2.61	3.06	7.97	11.53	0%
1ddt_127-134	2.69	14.7	7.58	11.5	41%	2.39	2.17	7.23	13.80	0%
1bt1_50-57	2.53	17.9	5.57	10.1	31%	2.97	0.98	5.78	10.55	0%
Average	2.34	10.5	6.04	10.2	34%	2.45	1.43	6.39	11.28	0%
					Length 12					
1q1wA_31-42	3.96	12.5	9.26	15.8	28%	4.48	0.11	10.6	17.11	6%
1ctm_9-20	3.84	1.76	9.04	14.9	71%	3.62	0.91	8.51	15.45	6%
1eguA_508-519	2.87	2.88	6.30	10.1	24%	2.86	0.09	6.21	11.47	0%
1ede_150-161	3.91	2.07	8.17	14.8	63%	3.89	0.16	9.35	15.87	0%
1d8wA_46-57	9.05	8.94	12.2	17.1	62%	9.07	0.29	12.8	17.71	0%
1ds1A_291-302	3.26	20.3	9.80	16.7	39%	3.81	0.76	11.8	18.44	0%
1f74A_11-22	4.29	6.84	9.22	16.0	61%	4.86	0.47	10.7	16.75	32%
1qopA_178-189	6.96	1.08	13.6	18.3	82%	6.13	1.08	12.9	19.83	14%
154l_153-164	3.61	10.3	9.67	15.4	54%	2.85	0.32	8.18	16.88	0%
1msc_9-20	4.16	11.7	12.4	17.3	54%	5.02	4.78	13.0	18.40	6%
Average	4.59	7.84	9.9	15.7	54%	4.66	0.90	10.4	16.79	6%

a: the RMSD of the best loop among 300 loops generated for each tested loop.
b: the RMSD of the second portion in the structure fragment with the best loop .
c: the average RMSD for all of the 300 loops.
d: the Maximum RMSD among all 300 sample loops.
e: the percentage of the convergent loops; For FBCCD, after the 30 cycles of forward walk.

Table 10. Comparison between FBCCD and CCD after 2000 cycles.

Loop	CCD					FBCCD with greedy				
	Loop Best[a]	2nd portion[b]	Conv.[c]	Loop Conv[d]	2nd portion[e]	Loop Best[a]	2nd portion[b]	Conv.[c]	Loop Conv[d]	2nd portion[e]
Length 4										
1i0hA_123-126	1.04	7.2	58%	1.24	0.93	1.03	0.01	21%	1.04	0.00
1qnrA_195-198	0.49	3.2	91%	0.61	2.52	0.38	0.02	7%	0.76	0.00
1tca_95-98	1.49	1.6	78%	1.81	1.84	1.21	0.11	1%	2.15	0.00
1ej0A_74-77	1.41	1.5	59%	1.41	1.51	1.09	0.03	1%	2.35	0.00
1fkf_42-45	0.72	3.3	84%	0.72	2.43	0.36	0.08	0%	N/A	N/A
1qopA_44-47	1.04	1.8	65%	1.05	1.88	0.55	0.06	1%	0.59	0.00
1aaj_82-85	1.17	1.5	95%	1.17	1.56	1.03	0.07	15%	2.14	0.00
1ads_99-102	1.21	1.5	95%	1.21	1.54	0.72	0.82	2%	1.89	0.01
1cbs_21-24	0.89	3.8	66%	0.89	3.88	0.54	0.01	15%	0.56	0.00
1nfp_37-40	0.22	1.4	51%	0.22	1.48	0.33	0.01	8%	1.31	0.00
Average	0.97	2.7	74%	1.03	1.96	0.72	0.12	7%	1.28	0.00
length 8										
1cruA_85-92	1.83	2.5	100%	1.83	2.56	2.19	0.03	52%	3.71	0.00
1ctqA_144-151	2.22	1.4	100%	2.22	1.46	1.70	0.01	57%	2.50	0.00
1i0hA_145-152	2.60	3.4	100%	2.60	3.44	2.61	0.04	28%	3.12	0.00
1gk8A_122-129	1.49	6.0	100%	1.49	6.00	1.16	0.03	9%	2.42	0.00
1ixh_106-113	1.54	3.7	99%	1.54	3.73	1.42	0.01	67%	1.42	0.00
1d8wA_334-341	1.63	3.8	76%	1.63	3.89	1.90	0.16	9%	3.42	0.00
1ds1A_20-27	1.72	3.0	100%	1.72	3.02	1.93	0.01	98%	1.93	0.00
1cbs_55-62	2.92	1.9	100%	2.92	1.95	2.27	0.01	61%	2.27	0.00
1ddt_127-134	2.57	2.9	100%	2.57	2.93	3.02	0.01	30%	3.03	0.00
1bt1_50-57	2.28	3.3	100%	2.28	3.39	2.34	0.01	47%	3.03	0.00
Average	2.08	3.2	97%	2.08	3.24	2.05	0.03	46%	2.69	0.00
length 12										
1q1wA 31-42	3.65	1.7	100%	3.65	1.71	4.24	0.05	37%	4.59	0.00
1ctm_9-20	3.85	1.7	100%	3.85	1.77	3.78	0.01	90%	3.83	0.01
1eguA_508-519	2.88	2.8	100%	2.88	2.87	2.78	0.01	85%	2.79	0.01
1ede_150-161	3.91	2.0	100%	3.91	2.07	3.86	0.04	34%	3.95	0.00
1d8wA_46-57	9.12	4.3	100%	9.12	4.31	8.76	0.01	35%	9.20	0.00
1ds1A_291-302	3.68	1.8	100%	3.69	1.84	3.87	0.05	14%	4.41	0.01
1f74A_11-22	4.39	2.1	100%	4.39	2.12	3.65	0.01	91%	3.65	0.01
1qopA_178-189	6.97	1.0	100%	6.97	1.07	4.74	0.03	50%	6.82	0.00
154l_153-164	3.47	2.7	100%	3.47	2.75	3.15	0.03	35%	3.30	0.00
1msc_9-20	4.79	2.2	100%	4.79	2.20	5.45	0.03	29%	5.86	0.00
Average	4.67	2.3	100%	4.67	2.28	4.43	0.03	50%	4.84	0.01

a: the RMSD of the best loop among 300 loops generated for each tested loop.
b: the RMSD of the second portion in the structure fragment with the best loop .
c: the percentage of the convergent loops; For FBCCD, at the end of the forward walk.
d: the RMSD of the best converged loop.
e: the RMSD for the second portion in the fragment with the best converged loop.

Our past experience in implementing the CCD method suggests that it takes significant more number of cycles to place the C-terminus of the loop exactly at the target than to place it at a neighborhood of the target. The proposed work demonstrates an approach to reduce the number of cycles needed while producing loops of comparable accuracy and more accurate second portion of the chain.

5.3.1 Overall Fragments

We performed two tests for FBCCD and CCD using the maximum number of iterations of 100 and 2000 respectively. For FBCCD, the distribution of the cycles among the forward and backward walk can be found in the caption of Tables 9 and 10. Since both CCD and FBCCD are meant for generating candidate loops that close the gap rather than predicting a native-like loop, a metric to judge the method is to see the quality of the best loop in the candidate pool in terms of its RMSD from the native loop.

Fig. 31 shows two examples of the structure fragments constructed for a tested loop from two proteins respectively (with PDB ID codes are 1nqr and 1d8w). Each structure fragment consists of the first portion, the best loop in the pool and the second portion of the chain. For simple viewing, only a short segment of the first portion is shown in the Fig. 31, since the first portion of the chain remains fixed during the process of loop closure. Fig. 31*A* shows the overlay of the structure fragments after 100 cycles. It appears that the best loop generated by FBCCD and CCD have comparable accuracy. In fact the RMSD of the loop is 0.811Å for CCD and 0.493Å for FBCCD (Table 9, row of 1qnr). The major difference lies in the accuracy of the second portion of the chain. We observed in Fig. 31*A* that the fragment obtained using FBCCD (in cyan) is closer to the native fragment (in red) than that using CCD (in purple). The same observation is true even after 2000 iterations of the same fragment (Fig. 31*B*). It appears that the second portion in the fragment of FBCCD (cyan in Fig. 31*A*) is closer to the native at the end of 100 iterations than that is obtained using CCD (purple in Fig. 31*B*) after 2000 iterations, even though we did not overlap them in the figure. In this case, the RMSD for second portion of the chain is 3.245Å for CCD at the end of 2000 cycles (Table 10, 1qnr row), and it is 0.742Å for FBCCD at the end of 100 cycles

(Table 10, 1qnr row). The accuracy of the loop and of the second portion appears to improve from 100 iterations to 2000 iterations for both methods (Fig. 31*A* vs. *B*). Fig. 31*A* and *B* investigated a loop of length four. Fig. 31*C* and *D* show a similar comparison for a loop of length eight (1d8w). We observe the similar message for this loop as to the shorter loop in Fig. 31*A* and *B*. It appears that the second portion of the fragment produced using FBCCD using 100 cycles is more accurate than that obtained using CCD after 2000 cycles. The loops generated by the two methods appear to have comparable accuracy (Fig. 31*C* and *D*).

5.3.2 100 Iterations

Tables 9 and 10 shows the details of the results for 30 loops after 100 (Table 9) and 2000 (Table 10) iterations were performed respectively. The best loop is the loop with the smallest RMSD from the native among the 300 possible loops that are able to connect the two portions of the chain. This work investigated the problem: how accurate the second portion of the chain will be after a loop is used to connect the two portions of the chain. Although the general belief is that if a loop closes the chain, the second portion should be fairly accurate, there has not been data to demonstrate the accuracy of the second portion, before and after the convergence of the loop.

The accuracy for the second portion of the chain is judged by the RMSD from native for a maximum of 40 amino acids in the second portion of the chain. Table 9 shows a situation that most of the 300 loops for each tested loop are not converged to the C-Anchor, since only 100 cycles were used. It is not surprising that the second portion is not quite accurate for CCD. The average RMSD of the second portion in the fragment with the best loop is 5.623Å, 10.503Å, and 7.835Å for loops of length four, eight and 12 respectively. However, the average RMSD of the second portion in the fragment with the best loop is 0.772Å, 1.433 Å, and 0.902Å for loops of length four, eight and 12 loops when FBCCD was used. The quality of the best loops is comparable for the two methods, with an average of RMSD (1.188Å, 2.342Å, 4.594Å) for CCD vs. (0.821 Å, 2.453Å, 4.660Å) for FBCCD when loops of length (4, 8, 12) are considered. The comparable loop accuracy is not surprising since both methods use the same number of iterations and the same number of random loops. In order to improve the accuracy for the best loop, more random loops need to be generated. For example, 5000 random loops were used for Table 4 in the CCD paper [219]. The percentage of the convergent loops is shown for both methods (Table 9, Column 6 and 11). The percentage for the FBCCD is the percentage of the convergent loops at the end of the forward walk. For example, 1qnr_A has nine out of 300 loops converged within RMS 0.08Å from the C-Anchor for CCD. No loops of the 300 converged for FBCCD at the end of 30 cycles of forward walk. In fact, the percentage of convergence is still close to zero at the end of the backward walk (data not shown). What is interesting is that even though less loops converge in the FBCCD method, the accuracy for the second portion is consistently smaller than that for the CCD method.

5.3.3 2000 Iterations

Table 10 shows the testing results using 2000 iterations when more loops are converged. On average, 74% of the length-4 loops converged using CCD method (Table 10, Column 4) and 7% loops converged using FBCCD method (Table 10, Column 9) after the 800 cycles of the forward walk. The percentage of converged loops is higher for longer loops. The quality of the best loops is still comparable between the two methods for lengh-4, length-8 and length-12 loops (Columns 2 and 7). However, the RMSD for the second portion of the fragment with the best loop (Columns 3 and 8) is consistently smaller for FBCCD than for CCD. When the comparison is restricted to the converged loops, the quality of the best loops is slightly better for CCD than for FBCCD. However, it could due to the number of converged loops, since less number of the loops converged in FBCCD than CCD (Columns 4 and 9). In fact, it is appropriate to use the converged loops as the candidate loops in the CCD method since the converged loops have small error for the second portion of the protein. However, the entire population of the 300 loops can be used as candidates for FBCCD, since even the non-converged loops have small RMSD for the second portion of the chain.

5.3.4 100 vs. 2000 Iterations

A goal of this method is to reduce the number of cycles needed to produce comparable or more accurate chain when it is compared to the CCD method. If we cross check Tables 9 and 10, we observe the following statistics. After 100 cycles, the second portion of the chain has on average (0.772Å, 1.433Å, 0.902Å) RMSD to native for fragment with the best loop of length (4, 8, 12) respectively (Table 9, the average rows of Column 8) when FBCCD was used. After 2000 cycles, the second portion of the chain has on average (2.746Å, 3.241Å, 2.277Å) RMSD to native for fragment with the best loop of length (4, 8, 12) respectively (Table 10, the average rows of Column 3) when CCD was used. Therefore, the second portion of the chain is more accurate when FBCCD is applied for 100 cycles. After 100 cycles, the best loop out of 300 has on average (0.821Å, 2.453Å, 4.660Å) RMSD from native for length (4, 8, 12) when FBCCD was used (Table 9, average rows of Column 7). After 2000 cycles, the best loop out of 300 has on average (0.969Å, 2.081Å, 4.673Å) RMSD from native for length (4, 8, 12) when CCD was used (Table 10, average rows of Column 2). Therefore, the best loops generated by the two methods have comparable quality. The results suggest that instead of running CCD for 2000 iterations, it is possible to run 100 iterations of FBCCD and produce loops of comparable quality and more accurate second portion of the chain. The current criterion to determine if a loop closes the gap of the chain is to check if the RMSD is within a small error, such as 0.08Å. This criterion was used in the CCD paper as well as in the current work.

One would expect that if the RMSD for loop *A* is 0.09Å and RMSD for loop *B* is 0.07Å, for instance, the second portion is more accurate when loop *A* is used to close the gap than when loop *B* is used. However, our data indicate that this is not always true. Table 11 shows five such examples. For each of the five loops, the three distances, $d1$, $d2$ and $d3$, are shown before and after the backward walk. In the example of the first loop, a smaller RMSD error (0.085Å vs. 0.139Å) produces more error in the second portion of the chain (2.441Å vs. 0.125Å) (Table 11). The data in Table 11 suggests that there might be ways to close the gap to provide more accurate second portion of the chain, without requiring that the RMSD of the error distances to be extremely small. An intuition of this is that the RMSD only measures the distance error; it does not measure the orientation error directly. The problem can be related to the alignment of two sets of three points (i.e., N, C_α, and C atoms) in 3-D space. There is only one way to align them exactly but there are infinite number of ways to align them not exactly. Our method relaxed the requirement of the RMSD error to certain extent and screen for those alignments that result in more accurate second portion of the chain.

Table 11. RMS of distance and accuracy of the second portion.

No.	Before backward walk					After backward walk				
	Distances			RMS atoms [a]	RMSD 2nd portion [b]	Distances			RMS atoms [a]	RMSD 2nd portion [b]
	d1	d2	d3			d1	d2	d3		
1	0.102	0.079	0.067	0.085	2.441	0.161	0.114	0.138	0.139	0.125
2	0.127	0.101	0.014	0.094	2.909	0.203	0.011	0.014	0.117	0.003
3	0.103	0.089	0.035	0.081	2.493	0.249	0.096	0.095	0.164	0.207
4	0.096	0.056	0.088	0.082	1.786	0.138	0.089	0.085	0.107	0.199
5	0.239	0.181	0.037	0.174	5.309	0.379	0.021	0.027	0.219	0.006

a: RMS of d1, d2 and d3.

b: RMSD to native for the second portion of the chain, measured for up to 40 Cα atoms.

CHAPTER 6

BUILDING THE FULL MODEL

In Chapter 5, a reverse kinematic solution for the loop closure problem in loop modeling is used. In loop closure problem, the N- and C-anchors of a missing fragment in the initial model are known. In this Chapter, we will use the traces extracted from Protein skeleton and the initial model built for secondary structures to build the entire model of the protein using FBCCD.

Given the skeleton traces on the density map and the initial partial structure built for SSEs-D sticks, the full model of the protein can be built in a two-stage approach. The first stage is to fill the gap between secondary structures in the partial model using the traces from volumetric skeleton obtained using Gorgon. In the second stage, the full models built for top topologies will be re-ranked using a multi-well energy function.

6.1 INTRODUCTION

The problem of loop closure is similar to the inverse kinematic problem in robotics. In robotics, if the number of degrees of freedom (DOF) is less than six, the number of solutions for the manipulator in 3-D space is finite. It has been proven that the loop closure problem with six degrees of freedom has at most sixteen possible solutions, whereas the number of possible solutions is infinite for the problem with more than six degrees of freedom [211,215,216,217]. Some analytical solutions were proposed [179,210,211,214]. When the number of DOF exceeds six, no analytical solution can be found. Therefore, one of the optimization methods should be used [215,216,217]. More details can be found in Chapter 5.

Building a loop fragment that fills the loop portion and best fit the trace found in the skeleton is similar to a robotic problem called sample-based motion planning. In robotics, the motion planning problem is to avoid collision with known obstacles while producing a continuous motion that connects a start and goal configurations. One solution for motion planning problem is to use sampling-based algorithms. In sampling-based algorithms the configuration space is represented with a roadmap of sampled configurations. The algorithm samples number of configurations in configuration space, and retains those in collision-free configuration space to use as milestones. A roadmap is then constructed that connects two milestones if the connection is completely in collision-free configuration space. Numerous works can be found in the literature that solves the problem of motion-planning using sampling-based algorithms. Most of methods are based on the probabilistic framework proposed in [225]. One method is to break the loop into active and passive parts [226]. In the active part, the forward kinematics sampling techniques are used and close samples then connected using a local planer. Exact inverse kinematics solution is found for the passive part and the samples follow the motion. Cortes *et al.* [227] have extended the active-passive method where one DOF is sampled at a time. A recent application of this extension to protein modeling was introduced in [228]. Another method [229] is to ignore the constraints at the beginning and then apply them through gradient descent. If no obstacles exist, a polynomial-time planner introduced with spherical joints [230].

The problem of filling the gap in the incomplete structure model is addressed in [231]. In their work, Lotan *et al.* [231] aimed to fill the gap using the density map of X-ray crystallography. The algorithm proposed consists of two stages. In the first stage, a total of 1000 random initial conformations are generated. The CCD method is used then to close the loop. Additional constraints are used to consider the density map and to avoid the collision. In the second stage, the initial conformations are ranked according to their density fit. The top-ranked conformations are then refined by minimizing a standard real-space target function. The problem addressed by [231] differs from our problem. In [231] the density map used has a high resolution and the partial structure given is well defined. The run time needed to build loops varies from 30 minutes for short loops (four amino acids) to 178 minutes for longer loops (15 amino acids). In the problem we are proposing, the traces and partial conformation are approximate. Lindert *et al.* [232] proposed a *de novo* folding approach, EM-Fold, guided by medium resolution volumetric maps, EM-Fold. EM-Fold uses a Monte Carlo sampling method to build the initial model of SSEs. A Monte Carlo refinement process then used to improve the placement of SSEs. On a later step, the loop and side chains are added by Rosetta's iterative side-chain repacking and backbone reconstruction protocols in which to generate a model at atomic resolution.

6.2 METHODS

The method of building the full model of a protein using its volumetric density map consists of three steps. The first step is to extract the features from the volumetric map. These features include the position of the secondary structure elements and the connections (loop traces) between these secondary structure elements. The first feature (i.e., the position of Secondary structure elements) is used to build the topology graph (see Section 4.4) where the second feature (i.e., loop traces) is used to update the weight of the links in this graph (see Section 4.7). Next, a number of full model all-atom candidates for each topology are built using the traces extracted from the volumetric map in the first step. The FBCCD method introduced in Chapter 5 is used to build the missing loop portions using the traces. Finally, the models of all candidates are ranked using a multi-well energy potential.

6.2.1 Feature Extraction from the Volumetric Density Map

We use a newly developed method, called SSETracer [124], to extract the position of secondary structure elements from volumetric density map. In SSETracer, image processing concepts are translated to features in a multi-task learning problem and are then solved by using Support Vector Machine (SVM). Each voxel in the volumetric map is classified into one of the three types of voxels: helix voxels, sheet voxels and background voxels. The feature extraction step in SSETracer characterizes each voxel based on the local geometrical features. Local gradient is often used to characterize the geometrical features and the local tensor used to define the local shape. Fig. 32A shows the secondary structure elements extracted from volumetric density map using SSETracer.

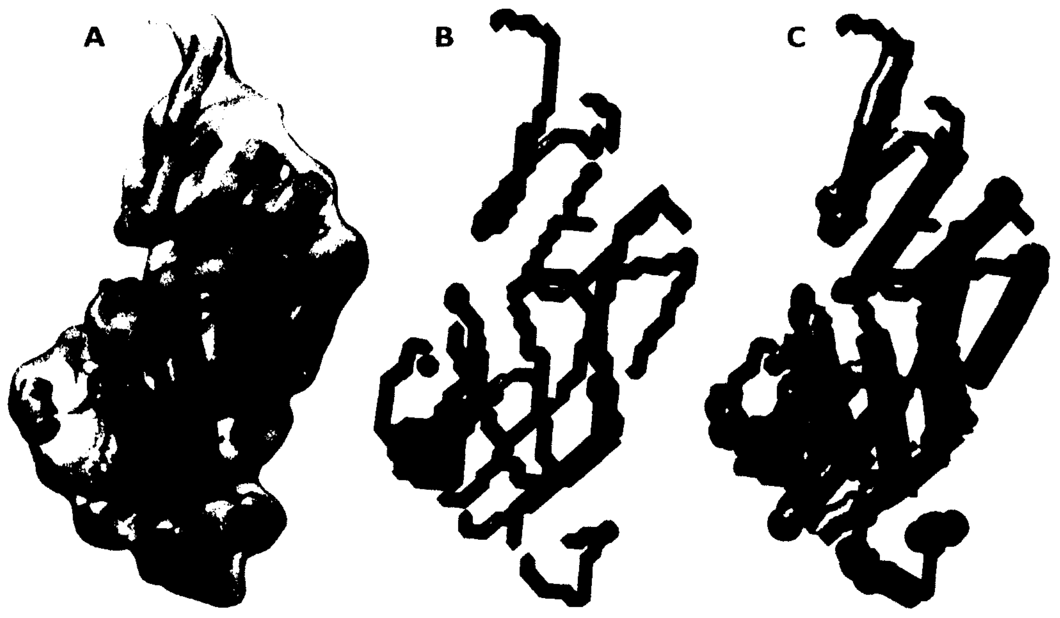


Figure 32. The SSEs-D helical sticks and Gorgons' skeleton. (A) The density map (grey) was simulated to 10Å resolution using protein 1HZ4 from the PDB. The helical SSEs-D (rods) were detected using SSETracer and viewed by Chimera. (B) The skeleton obtained for the same protein using Gorgon. The skeleton has some gaps. (C) The helical SSEs-D sticks (rods) superimpose the skeleton obtained by Gorgon.

In order to obtain other features from the CryoEM volumetric map, we used Gorgon to produce the skeleton of the protein (Fig. 32*B*). Based on the skeleton, we extracted the traces between secondary structure elements. One drawback of using Gorgon is the existence of gaps in the produced skeleton. The user may be able to fill the gaps through a graphical interface. However, we are able to overcome the problem of gaps in the skeleton when use the method introduced in Section 4.7. At the end of this step, a list of possible traces between each pair of SSEs-D sticks' ends is generated. These lists are used to update the weight of links in the topology graph.

6.2.2 Enumerate Top-K Topologies

The second step to build the full model is to reduce the large topological space quickly to a small subset of possible topologies without the use of energy evaluation. The goal is to include the true topology in such a subset, so that the conformations can be built for the likely topologies. We use the topology graph introduced in Section 4.4 to enumerate the top-K topologies. The design of the graph aims to utilize the constraint that arises from the length of the loop connecting two SSEs, since the loop length is a strong constraint to distinguish the correct topology. Note that any two nodes connected by an edge provide the information of the assignments of two consecutive SSEs on the protein sequence. The level of satisfactory of this constraint can be naturally expressed as an edge weight. The current weight mainly represents the difference between the estimated length of the loop connecting the two helical SSEs-D sticks and the estimated length of the trace extracted using Gorgon's skeleton between them in 3-D space. For more details see Section 4.7. Fig. 33 shows the traces extracted from CryoEM volumetric map for one topology. In order for the true topology to be ranked near the top of the list, the predicted helices based on the protein sequence have to be accurate enough. Two situations will affect the ranking the most. One is when a long helix is wrongly predicted as two short helices that are connected by a short loop. The other is when two short helices are predicted as one long helix. In order to identify the true topology, the search needs to consider the possible errors from both the secondary structure prediction and those from the detection in 3-D space. Biswas *et al.* [233] have studied the question, that is, how to reduce the computation of the mapping when the inaccuracy of the secondary structure predictions is considered. They have presented a method that combines the concept of dynamic graph with our constrained shortest path to identify the topology of the secondary structures.

The current implementation is limited in the ranking of the helices, although our dynamic programming approach is general for either helices or β-strands. In order to extend the current approach to β-strands, the location of the β-strands needs to be estimated. However, the β-sheets are generally not as accurately detected as the helices in the intermediate resolution density maps. It is expected that the β-strands will be estimated with many possible alternatives. It is still a challenging problem to rank the topology with both α-helices and β-sheets without the knowledge of a template.

6.2.3 Building the Atomic Model

The process of building the atomic model is divided into two major steps. In the first step, the atomic model of the helices was built. In the second step, the loops were reconstructed to connect the helices. The loop reconstruction was guided by the traces obtained from the skeleton of the volumetric map.

Figure 33. The skeleton obtained for the true topology. The traces extracted for protein 1HZ4 from the skeleton. SSEs-D sticks (rods) are shown. The full model has a minimum multi-well energy is shown.

6.2.3.1 Helices Reconstruction

SSETracer [124], an extension version of HelixTracer [115], was used to detect the SSEs from the volumetric map. The detected sticks are not always straight. In SSETracer [124], the helix is defined by a set of points, called spline, represents the central axial of the helix. For each topology ranked in the previous process, the helices are constructed to agree with the spline obtained from the volumetric map. The forward walk of FBCCD [185] was used to reconstruct the helices in this work. FBCCD aims at closing a gap that satisfies the constrained locations predefined by two ends of the chain by adjusting a single torsion angle at a time. It applies the idea of inverse kinematics in robotics. The idea of the construction of a bent helix can be potentially applied for the construction of β-strands.

The process starts by building a perfect helix using the torsion angles of α-helices. The values of phi and psi torsion angles used are -57 and -47, respectively. The number of amino acids in the perfect helix is determined by dividing the length of density spline by the rise of the helix (i.e., 1.5Å). The initial perfect helix constructed is straight. A set of points represent the perfect spline is calculated. The number of points in perfect spline and the spline detected from density map were selected to be equal. The points are selected in which any two consecutive points are 6Å apart (i.e., four amino acids apart). The only exception is for the last two points represent the last segment. The process starts by aligning the first point of the two splines. Fig. 34*A* shows the perfect helix and its spline after this step. The two splines virtually form *n*-1 segments, where *n* is the number of points in each spline. For the perfect helix, the torsion angles of the four amino acids correspond to each segment is determined. The alignment of segments from the two splines then starts one at a time. The eight torsion angles (i.e., two torsion angles for each amino acid) correspond to the segment being aligned are updated using FBCCD. To preserve the structure of the helix the update is only accepted if the new value of the torsion angle is within the predefined range (i.e., $\phi \in [-80, -40]$ and $\psi \in [-60, 10]$). The process terminates either when the maximum number of cycles is reached or the distance between the two points represent the ends of the segments is less than the cutoff distance. In our current implementation, the maximum number of cycles is set to 100 and the cutoff distance is set to 0.1Å. Fig. *34B* shows the two splines after all segments are aligned. Fig. *34C* shows the final bent helix aligned with the native helix.

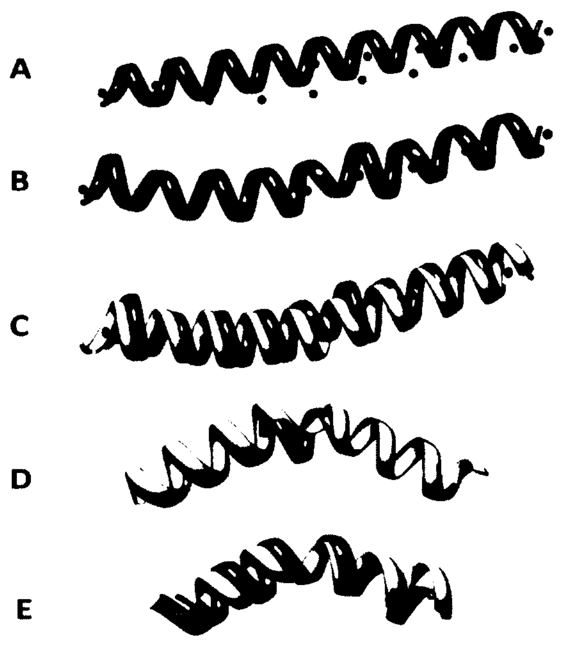


Figure 34. Helices reconstruction process. (A) The points (in black) represent the spline of the helix stick detected from volume map and the points (in red) represent the perfect helix constructed (in red). (B) The two set of points represent the two splines of helices are aligned. The bent helix is shown in red. (C) The bent helix superimposed with the native helix. (D) An example of a bent helix constructed (red) for one helix (gray) from Protein 1OXJ (PDB ID). (E) An example of a bent helix (red) constructed for one helix (gray) from protein 2IU1 (PDB ID).

The process of building bent helices improves the final RMSD of the model. In a previous study [183] we have used straight helices in the modeling process. The accuracy of reconstructed helices negatively impacts the final RMSD. Table 12 shows the RMSD of straight and bent helices reconstructed with the native structure. RMSD is calculated for backbone atoms (i.e., N, C_α, C, and O). For example, the RMSD of bent helix reconstructed for protein 3ODS is 0.88Å whereas the RMSD of the perfect straight helix is 1.72Å. This displacement in one helix will negatively affect the accuracy of the final model and enlarge the difference with the native structure. Moreover, this is expected to impact the potential energy of the final model because the contact map of residues is different than the native structure. Therefore, the true model may not be ranked among highest structures. Another advantage of the process is its speed. The process is very fast. It takes less than 40 milliseconds to build one bent helix of 51 amino acids (Table 12, row 5) using a regular laptop computer. The average time of reconstructing one helix of five bent helices is 37.4 milliseconds on a Lenovo X300 laptop (at 1.2 GHz).

6.2.3.2 Building Loops using the Skeleton

Loop closure problem arises in nearly all loop prediction problems. Loop closure is a problem of generating a loop whose N and C terminal residues satisfy the constrained locations. It has been proven that the loop closure problem with six degrees of freedom has at most sixteen possible solutions, whereas the number of possible solutions is infinite for the problem with more than six degrees of freedom [215,217]. Although analytical solutions have been proposed [179,210], optimization methods are often adopted [214,215].

The skeleton trace obtained from the volumetric map provides a rough estimate of the loop shape in this problem. The goal of our task is to build a loop that fits the trace. This is similar to the sample-based motion planning in robotics. The motion planning problem is to produce a continuous motion and to avoid collision with known obstacles. One solution for motion planning problem is to use sampling-based algorithms in which a roadmap of the collision-free configurations are retained. Most of methods are based on the probabilistic framework proposed in [225]. Dawen *et al.* break the loop into active and passive parts [226]. In the active part, the forward kinematics sampling techniques are used and close samples then connected using a local planer. The exact inverse kinematics solution is found for the passive part and the samples follow the motion. Cortes *et al.* [227] have extended the active-passive method where one DOF is sampled at a time. A recent application of this extension to protein modeling was introduced in [228]. Yakey *et al.* [229] ignore the constraints at the beginning and then apply them through gradient descent. The problem of filling the gap in incomplete structure model is addressed in [231]. In their work, Lotan *et al.* [231] aimed to fill the gap using the density map of X-ray crystallography. The algorithm proposed consists of two stages. In the first stage, a number of initial conformations were randomly generated. The CCD [219] method was then used to close the loop. Additional constraints from the density map were used to avoid the collision. In the second stage, the initial conformations were ranked by their fit to the density map.

Table 12. The performance of reconstructing bent helices.

No	ID[a]	Helix[b]	AA[c]	RMSD straight[d]	RMSD bent[e]	time[f]
1	1OXJ	644-669	26	2.61	1.26	55
2	1BZ4	90-123	34	1.32	0.81	30
3	2IU1	365-384	20	1.73	1.36	31
4	3ODS	124-146	23	1.72	0.88	31
5	2OEV	646-696	51	2.98	1.97	39
average			30.8	2.07	1.26	37.4

a: the PDB ID of the protein.

b: sequence number of the helix reconstructed.

c: number of amino acids in the helix.

d: the RMSD of the backbone atoms between the perfect straight helix reconstructed and the native structure.

e: the RMSD of the backbone atoms between the bent helix reconstructed and the native structure.

f: the time (in milliseconds) needed to reconstruct the bent helix.

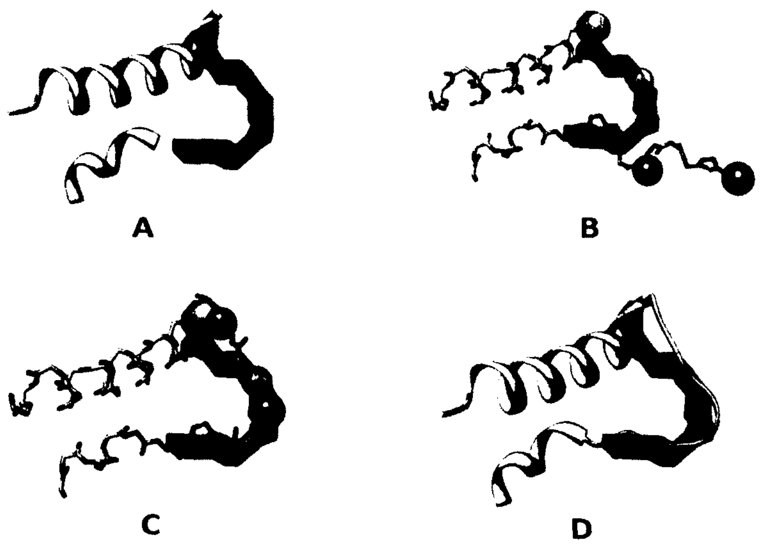


Figure 35. The approach of reconstructing the loop. In this example the loop and skeleton trace were divided into two segments each. The pink spheres represent the target points at skeleton trace. The two red spheres represent the moving points of the loop to be superimposed with pink spheres. (A) The structure of the two SSEs with skeleton trace. (B) The initial random loop is built and divided into two segments. (C) The loop was built to agree with the trace of the skeleton. (D) The loop after reconstruction shown in ribbon style.

Although the problem addressed by [231] shares the similar nature with our loop problem, it differs by the precision in the density map. The density map obtained using X-ray crystallography often has much higher resolution than the medium resolution map in our problem. Therefore, the method in [231] aims at finding the specific loop that satisfies the density constrains in fine precision. Since the skeleton of the medium resolution map only provides rough trace of the loop, we aim to find one of the many loops that align with the skeleton, but efficiently. The run time reported in [231] needs 30 minutes to build short loops (four amino acids) to 178 minutes for longer loops (15 amino acids). We developed a segment-wise loop-building method that can produce approximate loops efficiently. A typical collision-free loop of 10 amino acids takes approximately three minutes to build.

The step of building the loop is similar to the step is used to build the bent helices with two minor differences. In helix, the entire initial helix is built at the beginning and FBCCD is used to update the structure to superimpose the density spline. Moreover, the length of density spline and the length of the spline of the helix are equal (the density spline is used to estimate the length of the helix). In loop modeling, one segment of the loop is modeled at a time. Moreover, the length of the density trace and sequence trace are not necessarily same. The skeleton represents a rough trace of the loop and the length of the loop sequence is predetermined (i.e., the sequence between two SSEs). The process starts by dividing the sequence into segments of four amino acids. Thus, the virtual length of each segment is 15Å. In order to divide the skeleton trace into same number of segments, the points represent the segments were chosen in which they are (LengthTrace/nLoopSegments) Å apart. Where LengthTrace is the length of the skeleton trace and nLoopSegments is the number of segments in the loop sequence. Note that the length of the segment in the skeleton trace may not be the same to that on the loop sequence. We used the length of four amino acids for the segment of loop sequence for the ease of building. If the segment length is too short, the length of the skeleton trace will be very short and the agreement of the two segments will be hard. On the other hand, a long segment can produce loops quite different from the skeleton. Each segment is then reconstructed using the FBCCD in a process similar to helices reconstruction. We use the forward walk of the FBCCD to build one segment at a time. An initial random loop of four amino acids is constructed. It is simply connected with the first SSE, if it is the first portion of the loop to be modeled; otherwise it is connected with the previous modeled portion of the loop. One torsion angle is updated to move the end point of the loop segment to the end point of corresponding trace segment. The process continues until the two end points are 0.5Å apart. Also, the process is terminated if reaches the maximum number of cycles. The maximum number of cycles for all forward walks is set to 300. When terminated for the current segment, FBCCD moves to build the next segment. The same process is used to build all segments of the loop except for the last one. The backward walk of FBCCD is used to model the last segment of the loop. Backward walk is used due to its accuracy to model the segment and keep the second SSE in the original position. The target points we use in the superimposition step are the last two backbone atoms (C_α and C) and the first C_α atom of the second SSE. The process stops either if the RMSD of moving atoms is less or equal 0.05Å with the corresponding target atoms or the maximum number of cycles reaches 300. For more details about the FBCCD method we refer the reader to Chapter 5 and [185]. Fig. 35 shows one example of the approach used to reconstruct the structure of the loop.

6.2.3.3 Ranking Final Models

The work in this dissertation aims to establish an effective computational framework to derive the atomic detail of protein that is not visible at the medium resolution of volumetric maps. After the correspondence between SSEs on the map and SSEs on the sequence determined, the full model of the protein is computationally reconstructed. The structure of the SSEs and loops between SSEs are modeled using the two steps illustrated in the previous Subsection. Current implementation of this work calculates only the best 100 topologies. However, our experiments suggest that the true topology can be found among the best 35 topologies (see Subsection 4.5.2).

For each calculated topology, 100 random structures are modeled using the two steps described in the previous Subsection. Two kind of information are needed to construct the full model. For each SSE, we randomly assign the translation and shift for each stick and sequence elements, respectively. The translation parameter, T, is used to deal with the errors of SSEs detected from CryoEM volumetric map. The SSEs extracted from volumetric map using SSETracer might contain errors (i.e., the actual length) and can be off from the actual position. The shift parameter, S, is used to simulate the error of the secondary structure prediction of sequence segments. We allow translation for each stick in 3-D to be between [-3, 3]Å and we allow a shift on the sequence up to [-2, 2] amino acid positions.

Thus, for each topology, we constructed a pool of backbones, each of which can be represented by a set of parameters $(S_1,T_1),(S_2,T_2),\dots,(S_{M'},T_N)$, where $N \le M' \le M$, M is the number of sequence segments and N is the number of density sticks. The side chains of the modeled backbone structure are then packed using our own implementation of R3 algorithm [190]. The 10,000 modeled structures were sorted by the effective multi-well contact energy [52] and the RMSD100 of backbone atoms (i.e., N, C_α, C and O) is calculated for each modeled structure. RMSD100 is a normalized RMSD to a 100 residue protein [234]. RMSD is the averaged distance between two superimposed proteins.

6.3 EVALUATION AND RESULTS

A data set of 15 α-proteins was used to test the performance of our approach. The data set consists of 13 simulated volumetric maps and two experimentally derived density maps. The simulated maps were generated from the PDB structures using chimera [2] to 10Å resolution. The two experimentally derived density maps (EMDB ID 5100 with 6.8Å resolution, and 5030 with 6.4Å resolution) were downloaded from the Electron Microscopy Data Bank. The atomic structure (3IXV and 3FIN PDB ID, respectively) is available and aligned for both of the two CryoEM maps. The proteins range from 100 to 415 amino acids in length. We selected medium to large proteins in the data set due to the fact that many proteins in the CryoEM maps are medium to large in size. All the 15 proteins selected are α-proteins that do not contain β-sheets. Helices are often detected more accurately than the β-sheets in the medium resolution density maps. It is still a challenging problem to derive the atomic structures from the medium resolution data when β-sheets are involved. We used the actual position of secondary structure segments on the sequence obtained from the model in PDB. The helices on CryoEM volumetric maps were detected using SSETracer [124]. The Binary skeleton was obtained using Gorgon [187]. We built the topology graph and assigned the edge weight by tracing the skeleton. The K-shortest paths approach was used to calculate the best 100 topologies for each protein. We built 100 models for each of the 100 possible topologies. The models were ranked by our multi-well energy function. The models were evaluated by the RMSD100 of the backbone atoms when they are compared to the native structures.

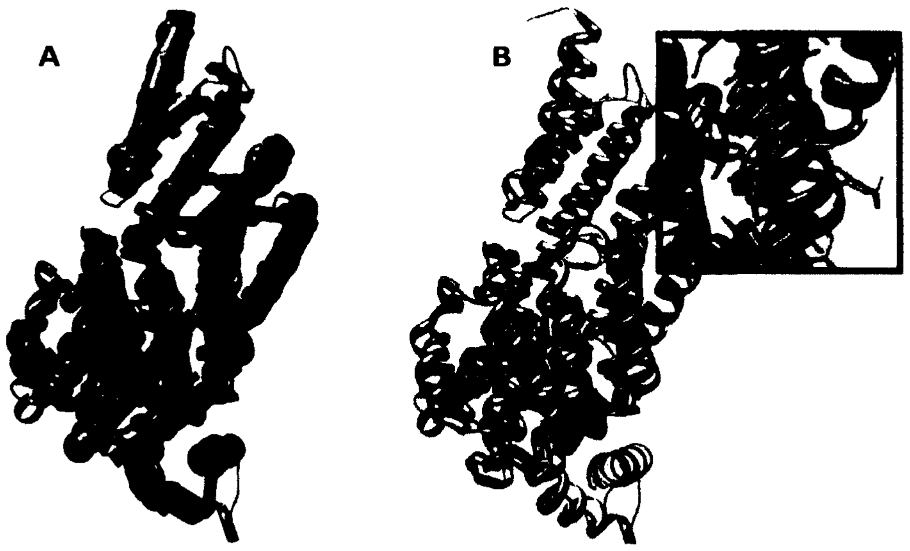


Figure 36. The full model built for the true topology. (A) The traces extracted for protein 1HZ4 from the skeleton. SSEs-D sticks (rods) are shown. The full model has a minimum multi-well energy is shown. (B) The superimposition of the native protein structure (light-blue) and the built model (dark-magenta).

Fig. 36A shows an example of helix sticks (red) detected from the density map that was simulated to 10Å resolution. In this case, SSETracer detected 19 of the 21 helices (1HZ4, Table 13, the 12th protein). In theory, there are $\binom{M}{N}(N!\,2^N)$ different topologies that is about 134×10^{23}. In reality though, a stick is often possible to be connected to only a few nearby sticks, instead of all the rest of the sticks, the resulting topology graph is often less dense as expected in theory. The true topology was ranked the 2nd in the huge topology space. We built 10,000 models for each protein, and used the multi-well energy function to rank the models. The highest ranked model with the correct topology (dark purple in Fig. 36B) is the 2nd out of 10,000 structures (Table 13, row 12). It has RMSD100 of 3.87Å for the backbone atoms when it is compared to the native protein. Note that our method models the full chain except the first and last loop, since they are often not resolved in the density map. More examples of the structures built are shown in Fig. 37.

Our approach to rank the topologies appears to be effective. For all the 15 proteins tested, our topology detection method consistently ranks the true topology among the top 35, with top-1 for six of the 15 proteins (Table 13, column 6). The 10,000 structures were ranked by our multi-well contact energy that uses the center of the side chain as a reduced representation of the side chain [52].The highest rank of the structure that has the correct topology is listed in column 7 (Table 13). Our previous study has shown that if the backbone coordinates of the native structure are fixed, the correct topology can generally be located at the top 25% of the list that is ranked by the effective contact energy [52]. In this study we relaxed the requirement of fixing the backbone coordinates of the native structure and built the possible backbones from the central helical axis. In addition, we used the connection information between the helical sticks to estimate the likelihood of the correspondence between the sequence segments and the sticks. Note that although the skeleton provides critical information regarding the connection relationship, it can be ambiguous in many places of the skeleton. Our work suggests that a true structure can be found near the top of the list (within the top 5% in the test). For example, in the case of 3LTJ (Table 13, row 8), the highest ranked conformation with the correct topology is at the 396th (the top 4% of all the 10,000 structures generated) of the list that was sorted by the energy

Table 13. The accuracy of protein modeling using CryoEM.

No.	ID[a]	#AA[b]	#hlces[c]	#sticks[d]	#Possible Topologies[e]	TopoRank[f]	eRank[g]	RMSDbb[h]
1	2PSR	100	5	4	1,920	31	21	5.45
2	1BZ4	144	5	5	3,840	1	7	3.34
3	1A7D	118	6	4	5,760	1	6	3.87
4	1JMW	146	6	4	5,760	11	25	4.65
5	1FLP	142	7	7	645,120	1	35	3.45
6	1NG6	148	9	7	23×10^{6}	2	33	3.63
7	2XB5	207	13	9	13×10^{10}	11	10	3.49
8	3LTJ	201	16	12	35×10^{14}	2	396	4.07
9	3ACW	293	17	14	97×10^{16}	32	43	3.29
10	1Z1L	345	23	14	116×10^{19}	11	114	3.51
11	3ODS	415	21	16	279×10^{20}	12	31	3.27
12	1HZ4	373	21	19	134×10^{23}	2	2	3.87
13	3HJL	329	20	20	255×10^{22}	1	37	2.99
14	3IXV_A	222	14	9	372×10^{9}	1	13	5.90
15	3fin_R	117	4	4	384	1	189	5.98

a: the PDB ID of the protein.
b: number of amino acids in the protein.
c: the number of actual helices in the protein.
d: the number of detected helices from CryoEM map.
e: the approximate total number of all possible topologies for the protein.
f: the rank of the true topology using topology graph.
g: the highest rank of the model of the protein with the correct topology among 1000 using the multi-well energy function.
h: the backbone RMSD100 of the best true model reconstructed with the native structure based on potential energy.

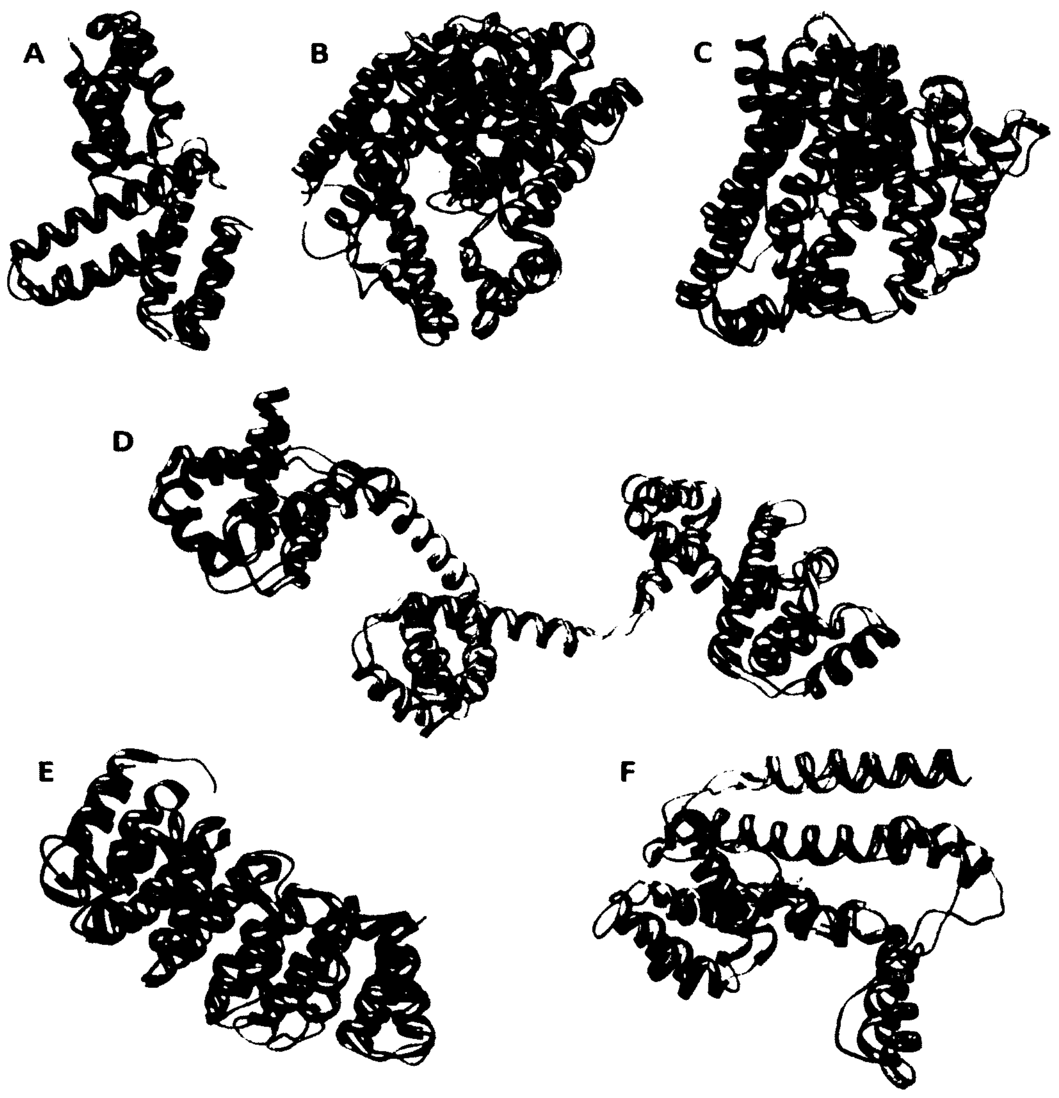


Figure 37. The full model built for some proteins. The native structure (light blue) is shown and superimposed with the built structure (dark-magenta) in (A) 1NG6 (B) 1Z1L (C) 3ACW (D) 3HJL (E) 3LTJ (F) 3IXV.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This dissertation has addressed the problem of *de novo* structure modeling of a protein using its CryoEM volumetric density map. Proteins are the molecules carry out the vital function and make more than the half of dry weight in every cell. The function of proteins varies from acting as enzymes, cellular signaling (i.e., hemoglobin) and Molecular transport. Protein in nature folds into a unique and energetically favorable 3-D structure which is critical and unique to its biological function [4,5]. The unique conformation in which the protein folds into is called a native structure [9]. The sequence of amino acids build up the protein ultimately determine its native structure, in which corresponds to the favorable energy of the molecule [10].

The current methods of protein structure determination are complicated, time-consuming, and expensive. Moreover, they are not suitable for all kind of proteins such as membrane proteins. On the other hand, the sequencing of proteins is fast, simple and relatively less expensive. Thus, the gap between number of known sequences and determined structures is growing, which is expected to keep growing, and the need for computational methods to increase the number of structures available is become critically important.

In contrast to traditional experimental techniques used to determine protein structures, CryoEM is a promising advanced image processing method for structure determination. Unlike X-ray crystallography, CryoEM is able to produce volumetric maps of proteins that are poorly soluble, large, and/or hard to crystallize. Unfortunately, the volumetric maps generated by CryoEM are unable to determine the structure of protein at atomic-resolution. However, some features of the protein can be visually and computationally identified such as the location of secondary structures.

The two protein structure prediction techniques (i.e., *ab initio* and comparative) have been proven to be capable of producing relatively good structural models for isolated proteins or domains [106]. Therefore, recent work has shown the ability of density maps to help in discriminating between models built by *ab intio* and/or comparative modeling to build final models [54,100,110,111,112,113] and other works show the grateful help of CryoEM maps in *de novo* modeling of protein structures.

7.1 CONCLUSION

In our work, we simply divide the *de novo* modeling of a protein using CryoEM volumetric map into two steps. The first step is to model the secondary structures elements. Then we model the missing loops between the elements. We use the CryoEM volumetric map to detect secondary structure elements. Several tools are available to detect secondary structure elements from CryoEM volumetric maps. In order to model secondary structure elements, we first solve the problem of correspondence between the SSEs-D detected from CryoEM volumetric map and the SSEs-S predicted from the primary sequence.

The topology determination for the secondary structure elements detected from the density map is a critical question in deriving the backbone from the CryoEM map at the medium, resolution. The major challenge in this problem is the large solution space due to the combinatorial nature of the problem. The number of possible topologies for a protein of M number of helices in the primary structure and N sticks in the CryoEM volumetric map is $\binom{M}{N}N!\,2^N$. The topology determination is proved to be NP-Hard (Subsection 4.6.1). We have proposed three approaches to solve such a problem.

The first approach is the naïve approach (Section 4.3). In the naïve approach, we have enumerated all possible topologies and then used a geometrical screening step to keep the only valid topologies. This approach does not involve the construction of the loops, yet it is still able to distinguish most of the bad structures. Such approach includes the newly developed parallel simulated annealing process, the distance and length screening, and the incorporation of more efficient algorithms for adding side chains. A test with 35 low resolution density maps shows that the highest ranked structure that has the correct topology can be found within the top 1% of the list ranked by the effective energy that is formed by the helices.

We presented a graph representation to the problem of the enumeration of the valid topologies of the skeletons that can be detected from the CryoEM volumetric density map. Our implementation shows that the graph is effective in enumerating the valid topologies in general. For some proteins, it is more effective than others depending on the geometrical nature of the skeletons.

In the second approach, we used the depth first search algorithm to generate valid topologies of a protein directly from its topology graph (Subsection 4.4). This approach, as expected, beats the first approach in terms of speed. This approach, similar to the first approach, does not involve the construction of the loops. The geometry screening step was implicitly used in the building of the layered graph. A test of 25 low-resolution density maps shows that the highest ranked structure that has the correct topology can be found within the top 1% of the list ranked by the same effective multi-well energy that was used in the same approach.

In the third approach we have introduced a dynamic programming approach, TopoDP. In TopoDP, the complexity of the problem was minimized. The factorial term in $\binom{M}{N}N!\,2^N$ was reduced to $(D+1)^2N^2$, where D is the difference between number of SSEs-S and SSEs-D (M-N). However, the problem is still NP-Hard. We gave a $O((D+1)^2N^2 2^N)$ dynamic programming algorithm to find the valid topology with the minimum cost (Section 4.5). An initial test of 15 proteins suggests that our dynamic programming method is feasible for the proteins of much larger size than we could handle before. It also suggests that it is possible to derive a small subset of the entire topological space and contain the true topology. The test shows, as expected, that the third approach beats the first two approaches in terms of speed and memory usage.

In contrast to the second approach, we used more features from CryoEM volumetric map to update the weights of the topology graph. The skeleton of the protein was extracted using Gorgon [162] and a list of traces between each two secondary structure elements was built. The problem of gaps may be found in the skeleton was solved in Section 4.7. We also showed the concept of finding the top-K ranked valid topologies (Subsection 4.6.3). To find the top-K topologies, we converted the problem into a problem of finding K-shortest paths for the constraint topology graph. An algorithm based on Yen's algorithm was introduced. A comparison between our dynamic programming algorithm and Gorgon [162] on a set of 14 proteins has shown the superiority of our method over Gorgon in terms of accuracy, speed, and memory usage (Subsection 4.7.1). The comparison was mainly accomplished by comparing the quality of the set of candidate topologies each method produces.

In an advanced step, we used CryoEM volumetric map features to build the full model of the protein. The traces built using the skeleton which obtained by Gorgon were used to model the loops between secondary structure elements on the CryoEM volumetric map (Section 6.2). We have used the concepts and techniques from robotics and computational geometry to model the long chain kinematics of missing loops. In order to model the missing loops, the loop closure problem should be solved. CCD is a well-known method for the loop closure problem. It is mathematically clear and easy to implement. We have investigated the accuracy of the second portion of the chain when CCD is used to close the gap. The RMSD for superimposing the C-terminus of the loop with the C-Anchor has been used to judge if a loop converges to the destination in CCD. We have given counter examples for the relationship between the RMSD error and the accuracy of the second portion of the chain, measured by the RMSD to native. In Chapter 5, we have developed an effective method to connect two portions of a chain using FBCCD. This method is not dependent on the convergence of the loop. It relaxed the requirement of the RMSD error to some extent. Our results indicate that it takes less number of iterations to produce comparable quality of loops and it consistently produces more accurate second portion of the chain than the CCD method. This is due to the addition of the backward walk that directly adjusts the accuracy of the second portion of the chain.

A test on a set of 15 proteins and their density maps at 10Å resolution was done. We built a topology graph for each protein and generated the shortest 100 paths which represented the top 100 topologies. For each topology, we constructed 100 conformations each consisting of the coordinate of the heavy atoms in the backbone and side chains of the protein except for the first and last loops. The conformations were sorted based on their effective energy. The test shows that a true structure can be found within the top 5% of the structures generated.

7.2 FUTURE WORK

For larger proteins, *de novo* modeling needs efficient algorithmic methods and advanced computational resources, such as supercomputers. Additionally, computations carried out by High Performance Computing (HPC) resources will exhibit better performance and speed-up. This will certainly aid in predicting more proteins than conventional computational resources. Furthermore, this will shorten the gap between protein sequences available and structures of proteins modeled. Unfortunately, the current HPC resources might not be available for many researchers. Therefore, we are planning to use Graphical Processing Unit (GPU), as an alternative, because it is more readily available. Additionally, GPU exhibited impressive processing capabilities, as well as helping in the reduction of power consumption and cost. Those two factors are considered the main driving force behind the rapid increase in the use of GPUs for scientific computing over the last few years.

We plan to extend previous work to include sheet secondary structures in the framework in the near future. The recently proposed tool, SSETracer, shows some positive results for extracting the sheets secondary structure from the CryoEM volumetric density maps. Fig. 38 shows an example of a region detected by SSETracer for the sheet secondary structure.

Moreover, we plan to develop the current framework as a tool which can help researchers in predicting proteins using CryoEM volumetric maps. Such a tool will be available online as a better alternative for Gorgon.

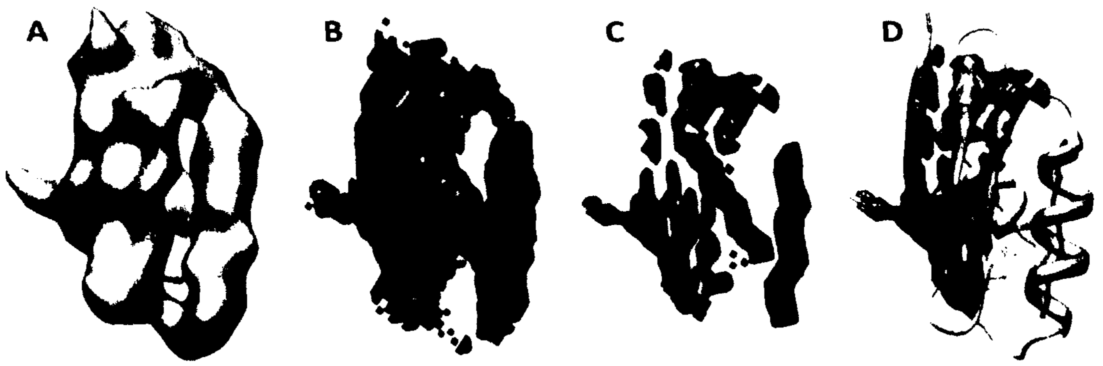


Figure 38. SSEs-D detection using SSETracer for protein 2AW0. (A) Density map of this protein at 8Å resolution; (B) Initial predicted helix (red) & sheet (blue) voxels after machine learning step; (C) helix (red) & sheet (blue) voxels after post-processing step; (D) Comparison of the X-ray structure and identified secondary structure locations.

REFERENCES

1. Ludtke SJ, Baldwin PR, Chiu W (1999) EMAN: Semi-automated software for high resolution single particle reconstructions. Journal of Structural Biology 128: 82-97.
2. Pettersen EF, Goddard TD, Huang CC, Couch GS, Greenblatt DM, et al. (2004) UCSF Chimera—A visualization system for exploratory research and analysis. Journal of Computational Chemistry 25: 1605-1612.
3. Ridley M (2000) Genome. New York: Harper Perennial. 352 p.
4. Blundell TL, Bedarker SN, Rinderknecht E, Humbel RE (1978) Insulin-like growth factor: a model for tertiary structure accounting for immunoreactivity and receptor binding. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 75: 180-184.
5. Weber IT (1990) Evaluation of homology modeling of HIV protease. Proteins: Structure, Function, and Bioinformatics 7: 172-184.
6. Rappe AK, Casewit CJR (1997) Molecular Mechanics Across Chemistry: University Science Books.
7. Siegel GJ, Albers RW, Brady S, Price D (2006) Basic Neurochemistry, Seventh Edition: Molecular, Cellular and Medical Aspects: Elsevier Academic Press.
8. Ashok K Chauhan, Varma A (2009) A Textbook of Molecular Biotechnology: I K International Publishing House. 1352 p.
9. Murray RK, Granner DK, Mayes PA, Rodwell VW (2006) Harper's Illustrated Biochemistry: McGraw-Hill Medical.
10. Anfinsen CB (1973) Principles that govern the folding of protein chains. Science 181: 223-230.
11. Guzzo AV (1965) The Influence of Amino Acid Sequence on Protein Structure. Biophysical Journal 5: 809-822.
12. Lewis PN, Gō N, Go M, Kotelchuck D, Scheraga HA (1970) Helix Probability Profiles of Denatured Proteins and Their Correlation with Native Structures. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 65: 810-815.
13. Schiffer M, Edmundson AB (1967) Use of helical wheels to represent the structures of proteins and to identify segments with helical potential. Biophysics Journal 7: 121-135.
14. Prothero JW (1966) Correlation between the distribution of amino acids and alpha helices. Biophysical Journal 6: 367-370.
15. Donald Voet, Judith G. Voet, Pratt CW (2008) Fundamentals of Biochemistry: Life at the Molecular Level: Wiley. 1240 p.
16. CM V (1968) Stereochemical criteria for polypeptides and proteins. V. Conformation of a system of three linked peptide units. Biopolymers 6: 1425-1436.
17. Némethy G, Printz MP (1972) The γ Turn, a Possible Folded Conformation of the Polypeptide Chain. Comparison with the β Turn. Macromolecules 5: 755-758.

18. Bernasconi A, Segre AM (2000) Ab Initio Methods for Protein Structure Prediction: A New Technique based on Ramachandran Plots. the European Research Consortium for Informatics and Mathematics (ERCIM).
19. Moult J, Fidelis K, Kryshtafovych A, Rost B, Hubbard T, et al. (2007) Critical assessment of methods of protein structure prediction-Round VII. Proteins: Structure, Function, and Bioinformatics 69: 3-9.
20. Fersht AR (2002) Max Ferdinand Perutz OM FRS. Nature Structural Biology 9: 245-246.
21. Berman H, Westbrook J, Feng Z, Gilliland G, Bhat T, et al. (2000) The Protein Data Bank. Nucleic Acids Research 28: 235-242.
22. Sussman JL, Lin D, Jiang J, Manning NO, Prilusky J, et al. (1998) Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules. Acta Crystallographica Section D, Biological Crystallography 54: 1078-1084.
23. Xiang Z (2006) Advances in Homology Protein Structure Modeling. Current Protein and Peptide Science 7: 217-227.
24. Johnson MS, Srinivasan N, Sowdhamini R, Blundell TL (1994) Knowledge-based protein modeling. Critical Reviews in Biochemistry and Molecular Biology 29: 1-68.
25. Levinthal C (1968) Are there pathways for protein folding? Extrait du Journal de Chimie Physique 65: 44.
26. Fischer D, Barret C, Bryson K, Elofsson A, Godzik A, et al. (1999) CAFASP-1: critical assessment of fully automated structure prediction methods. Proteins: Structure, Function, and Bioinformatics Suppl 3: 209-217.
27. Defay T, Cohen FE (1995) Evaluation of current techniques for ab initio protein structure prediction. Proteins: Structure, Function, and Bioinformatics 23: 431-445.
28. Jones DT (1997) Progress in protein structure prediction. Current Opinion in Structural Biology 7: 377-387.
29. Benner SA, Cohen MA, Gerloff D (1992) Correct structure prediction? Nature 359.
30. Zhang Y (2008) Progress and challenges in protein structure prediction. Current Opinion in Structural Biology 18: 342-348.
31. Bonneau R, Baker D (2001) Ab initio protein structure prediction: progress and prospects. Annual Review of Biophysics and Biomolecular Structure 30: 173-189.
32. Hinds DA, Levitt M (1994) Exploring conformational space with a simple lattice model for protein structure. Journal of Molecular Biology 243: 668-682.
33. Ishikawa K, Yue K, Dill KA (1999) Predicting the structures of 18 peptides using Geocore. Protein Science 8: 716-721.
34. Skolnick J, Kolinski A (1991) Dynamic Monte Carlo simulations of a new lattice model of globular protein folding, structure and dynamics. Journal of Molecular Biology 221: 499-531.
35. Dill KA, Bromberg S, Yue K, Fiebig KM, Yee DP, et al. (1995) Principles of protein folding--a perspective from simple exact models. Protein science 4: 561-602.

36. Skolnick J, Kolinski A (1994) Monte Carlo simulations of protein folding. I. Lattice model and interaction scheme. Proteins: Structure, Function, and Bioinformatics 18: 338-352.
37. Skolnick J, Kolinski A (1994) Monte Carlo simulations of protein folding. II. Application to protein A, ROP, and crambin. Proteins: Structure, Function, and Bioinformatics 18: 353-366.
38. Reva BA, Finkelstein AV, Sanner MF, Olson AJ (1996) Adjusting potential energy functions for lattice models of chain molecules. Proteins: Structure, Function, and Bioinformatics 25: 379-388.
39. Oldziej S, Czaplewski C, Liwo A, Chinchio M, Nanias M, et al. (2005) Physics-based protein-structure prediction using a hierarchical protocol based on the UNRES force field: Assessment in two blind tests. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 102: 7547-7552.
40. Sun W, He J. Effect of sidechain anisotropy on residue contact determination; 2009. pp. 181-188.
41. Krivov GG, Shapovalov MV, Dunbrack RL (2009) Improved prediction of protein side-chain conformations with SCWRL4. Proteins: Structure, Function, and Bioinformatics 77: 778-795.
42. Wang CX, Wan SZ, Xiang ZX, Shi YY (1997) Incorporating hydration force determined by boundary element method into stochastic dynamics. Journal of Physical Chemistry B 101: 2320-2235.
43. Xiang ZX, Shi YY, Xu YW (1995) Solving the finite-difference, nonlinear, Poisson-Boltzmann equation under a linear-approach. Journal of Computational Chemistry 16: 200-206.
44. Bowie JU, Eisenberg D (1994) An evolutionary approach to folding small alpha-helical proteins that uses sequence information and an empirical guiding fitness function. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 91: 4436-4440.
45. Alm E, Baker D (1999) Matching theory and experiment in protein folding. Current Opinion in Structural Biology 9: 189-196.
46. Huang ES, Subbiah S, Levitt M (1995) Recognizing native folds by the arrangement of hydrophobic and polar residues. Journal of Molecular Biology 252: 709-720.
47. Samudrala R, Xia Y, Huang E, Levitt M (1999) Ab Initio Protein Structure Prediction Using a Combined Hierarchical Approach. Proteins: Structure, Function, and Bioinformatics: 194-198.
48. Miyazawa S, Jernigan RL (1999) An empirical energy potential with a reference state for protein fold and sequence recognition. Proteins: Structure, Function, and Bioinformatics 36: 357-369.
49. Sippl MJ (1995) Knowledge-based potentials for proteins. Current Opinion in Structural Biology 5: 229-235.
50. Simons KT, Ruczinski I, Kooperberg C, Fox BA, Bystroff C, et al. (1999) Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins. Proteins: Structure, Function, and Bioinformatics 34: 82-95.

51. Sun W, He J (2009) Native secondary structure topology has near minimum contact energy among all possible geometrically constrained topologies. Proteins: Structure, Function, and Bioinformatics 77: 159-173.
52. Sun W, He J (2009) Reduction of the secondary structure topological space through direct estimation of the contact energy formed by the secondary structures. BMC Bioinformatics 10: S40.
53. Rohl CA, Strauss CE, Misura KM, Baker D (2004) Protein structure prediction using Rosetta. Methods Enzymol 383: 66-93.
54. Lu Y, Strauss CEM, He J (2007) Incorporation of Constraints from Low Resolution Density Map in Ab Initio Structure Prediction Using Rosetta. Proceeding of 2007 IEEE International Conference on Bioinformatics and Biomedicine Workshops: 67-73.
55. Simons KT, Kooperberg C, Huang E, Baker D (1997) Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. Journal of Molecular Biology 268: 209-225.
56. Bradley P, Misura KMS, Baker D (2005) Toward high-resolution de novo structure prediction for small proteins. Science 309: 1868-1871.
57. Das R, Qian B, Raman S, Vernon R, Thompson J, et al. (2007) Structure prediction for CASP7 targets using extensive all-atom refinement with Rosetta@home. Proteins: Structure, Function, and Bioinformatics 69: 118-128.
58. Kaufmann KW, Lemmon GH, DeLuca SL, Sheehan JH, Meiler J (2010) Practically useful: what the Rosetta protein modeling suite can do for you. Biochemistry 49: 2987-2998.
59. Zhang Y, Skolnick J (2004) Automated structure prediction of weakly homologous proteins on a genomic scale. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 101: 7594-7599.
60. Wu S, Skolnick J, Zhang Y (2007) Ab initio modeling of small proteins by iterative TASSER simulations. BMC Bioinformatics 5: 17.
61. Klepeis JL, Wei Y, Hecht MH, Floudas CA (2005) Ab initio prediction of the three-dimensional structure of a de novo designed protein: a double-blind case study. Proteins: Structure, Function, and Bioinformatics 58: 560-570.
62. Klepeis JL, Floudas CA (2003) ASTRO-FOLD: A Combinatorial and Global Optimization Framework for Ab Initio Prediction of Three-Dimensional Structures of Proteins from the Amino Acid Sequence. Biophysical Journal 85: 2119-2146.
63. Kihara D, Lu H, Kolinski A, Skolnick J (2001) TOUCHSTONE: An ab initio protein structure prediction method that uses threading-based tertiary restraints. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 98: 10125-10130.
64. Skolnick J, Zhang Y, Arakaki AK, Kolinski A, Boniecki M, et al. (2003) TOUCHSTONE: a unified approach to protein structure prediction. Proteins: Structure, Function, and Bioinformatics 53: 469-479.
65. Baker D, Sali A (2001) Protein structure prediction and structural genomics. Science 294: 93-96.

66. Ginalski K (2006) Comparative modeling for protein structure prediction. Current Opinion in Structural Biology 16: 172-177.
67. John B, Sali A, Journals O (2003) Comparative protein structure modeling by iterative alignment, model building and model assessment. Nucleic Acids Research 31: 3982-3992.
68. Tress M, Ezkurdia I, Grana O, Lopez G, Valencia A (2005) Assessment of Predictions Submitted for the CASP6 Comparative Modeling Category. Proteins: Structure, Function, and Bioinformatics 7: 27-45.
69. Fiser A, Sali A (2003) Comparative Protein Structure Modeling. In: Chasman D, editor. Protein Structure: Determination, Analysis, and Application for Drug discovery. New York: Marcel Dekker, Inc. pp. 167-206.
70. Sánchez R, Sali A (1997) Comparative protein structure modeling as an optimization problem. Journal of Molecular Structure (Theochem) 398: 489-496.
71. Bowie JU, Luthy R, Eisenberg D (1991) A method to identify protein sequences that fold into a known three-dimensional structure. Science 253: 164-170.
72. Jones DT, Taylort WR, Thornton JM (1992) A new approach to protein fold recognition. Nature 358: 86-89.
73. Moult J, Fidelis K, Kryshtafovych A, Rost B, Tramontano A (2009) Critical assessment of methods of protein structure prediction—Round VIII. Proteins: Structure, Function, and Bioinformatics 77: 1-4.
74. Zhang Y, Skolnick J (2005) The protein structure prediction problem could be solved using the current PDB library. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 102: 1029-1034.
75. Skolnick J, Kihara D, Zhang Y (2004) Development and large scale benchmark testing of the PROSPECTOR_3 threading algorithm. Proteins: Structure, Function, and Bioinformatics 56: 502-518.
76. Jaroszewski L, Rychlewski L, Li Z, Li W, Godzik A (2005) FFAS03: a server for profile-profile sequence alignments. Nucleic Acids Research 33: W284-288.
77. Zhou H, Zhou Y (2005) Fold Recognition by Combining Sequence Profiles Derived From Evolution and From Depth-Dependent Structural Alignment of Fragments. Proteins: Structure, Function, and Bioinformatics 58: 321-328.
78. Ginalski K, Pas J, Wyrwicz LS, Grotthuss MV, Bujnicki JM, et al. (2003) ORFeus: Detection of distant homology using sequence profiles and predicted secondary structure. Nucleic Acids Research 31: 3804-3807.
79. Jones DT (1999) GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. Journal of Molecular Biology 287: 797-815.
80. Cheng J, Baldi P (2006) A machine learning information retrieval approach to protein fold recognition. Bioinformatics 22: 1456-1463.
81. Karplus K, Barrett C, Hughey R (1998) Hidden Markov models for detecting remote protein homologies. Bioinformatics 14: 846-856.
82. Söding J (2005) Protein homology detection by HMM-HMM comparison. Bioinformatics 21: 951-960.

83. Shi J, Blundella TL, Mizuguchi K (2001) FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. Journal of Molecular Biology 310: 243-257.
84. Lee MR, Tsai J, Baker D, Kollman PA (2001) Molecular dynamics in the endgame of protein structure prediction. Journal of Molecular Biology 313: 417-430.
85. Ponder JW, Case DA (2003) Force fields for protein simulations. Advances in Protein Chemistry 66: 27-85.
86. Tsui V, Case DA (2001) Theory and applications of the generalized Born solvation model in macromolecular simulations. Biopolymers 56: 275-291.
87. Wroblewska L, Skolnick J (2007) Can a physics-based, all-atom potential find a protein's native structure among misfolded structures? I. Large scale AMBER benchmarking. Journal of Computational Chemistry 28: 2059-2066.
88. Summa CM, Levitt M (2007) Near-native structure refinement using in vacuo energy minimization. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 104: 3177-3182.
89. Sali A, Blundell TL (1993) Comparative protein modelling by satisfaction of spatial restraints. Journal of Molecular Biology 234: 779-815.
90. Fiser A, Šali A (2003) Modeller: generation and refinement of homology-based protein structure models. Methods in Enzymology 374: 461-491.
91. Topham CM, Thomas P, Overington JP, Johnson MS, Eisenmenger F, et al. (1990) An assessment of COMPOSER: a rule-based approach to modelling protein structure. Biochemical Society Symposium 57: 1-9.
92. Guex N, Peitsch MC (1997) SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modeling. Electrophoresis 18: 2714-2723.
93. Chiu W, Schmid MF (1997) Pushing back the limits of electron cryomicroscopy. Nature Structural Biology 4: 331-333.
94. Zhou ZH, Dougherty M, Jakana J, He J, Rixon FJ, et al. (2000) Seeing the herpesvirus capsid at 8.5 A. Science 288: 877-880.
95. Ludtke SJ CD, Song JL, Chuang DT, Chiu W. (2004) Seeing GroEL at 6 A resolution by single particle electron cryomicroscopy. Structure 12: 1129-1136.
96. Chiu W, Baker ML, Jiang W, Zhou ZH (2002) Deriving folds of macromolecular complexes through electron cryomicroscopy and bioinformatics approaches. Current opinion in structural biology 12: 263-269.
97. Wang L, Sigworth FJ (2006) Cryo-EM and Single Particles. Physiology 21: 13-18.
98. Jiang Q-X, Wang D-N, MacKinnon R (2004) Electron microscopic analysis of KvAP voltage-dependent K+ channels in an open conformation. Nature 430: 806-810.
99. Chang W-H, Chiu MTK, Chen C-Y, Yen C-F, Lin Y-C, et al. (2010) Zernike phase plate cryoelectron microscopy facilitates single particle analysis of unstained asymmetric protein complexes. Structure 18: 17-27.
100. Topf M, Lasker K, Webb B, Wolfson H, Chiu W, et al. (2008) Protein structure fitting and refinement guided by cryo-EM density. Structure 16: 295-307.
101. Saibil HR (2000) Conformational changes studied by cryo-electron microscopy. Nature Structural Biology 7: 711-714.

102. Mitra K, Frank J (2006) Ribosome dynamics: insights from atomic structure modeling into cryo-electron microscopy maps. Annual Review of Biophysics and Biomolecular Structure 35: 299-317.
103. Zhang X, Jin L, Fang Q, Hui WH, Zhou ZH (2010) 3.3 Å Cryo-EM Structure of a Nonenveloped Virus Reveals a Priming Mechanism for Cell Entry. Cell 141: 472-482.
104. Böttcher B, Wynne SA, Crowther RA (1997) Determination of the fold of the core protein of hepatitis B virus by electron cryomicroscopy. Nature 386: 88-91.
105. Conway JF, Cheng N, Zlotnick A, Wingfield PT, Stahl SJ, et al. (1997) Visualization of a 4-helix bundle in the hepatitis B virus capsid by cryo-electron microscopy. Nature 386: 91-94.
106. Baker ML, Jiang W, Wedemeyer WJ, Rixon FJ, Baker D, et al. (2006) Ab initio modeling of the herpesvirus VP26 core domain assessed by CryoEM density. PLoS Computational Biology 2: e146.
107. Martin AG, Depoix F, Stohr M, Meissner U, Hagner-Holler S, et al. (2007) Limulus polyphemus hemocyanin: 10 A cryo-EM structure, sequence analysis, molecular modelling and rigid-body fitting reveal the interfaces between the eight hexamers. Journal of Molecular Biology 366: 1332-1350.
108. Villa E, Sengupta J, Trabuco LG, LeBarron J, Baxter WT, et al. (2009) Ribosome-induced changes in elongation factor Tu conformation control GTP hydrolysis. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 106: 1063-1068.
109. Lawson CL, Baker ML, Best C, Bi C, Dougherty M, et al. (2011) EMDataBank.org: unified data resource for CryoEM. Nucleic Acids Research 39: D456-D464.
110. Topf M, Baker ML, John B, Chiu W, Sali A (2005) Structural characterization of components of protein assemblies by comparative modeling and electron cryo-microscopy. Journal of Structural Biology 149: 191-203.
111. Topf M, Baker ML, Marti-Renom MA, Chiu W, Sali A (2006) Refinement of protein structures by iterative comparative modeling and CryoEM density fitting. Journal of Molecular Biology 357: 1655-1668.
112. Lu Y, He J, Strauss CE (2008) Deriving topology and sequence alignment for the helix skeleton in low-resolution protein density maps. Journal of Bioinformatics and Computational Biology 6: 183-201.
113. DiMaio F, Tyka MD, Baker ML, Chiu W, Baker D (2009) Refinement of protein structures into low-resolution density maps using rosetta. Journal of Molecular Biology 392: 181-190.
114. Jiang W, Baker ML, Ludtke SJ, Chiu W (2001) Bridging the information gap: computational tools for intermediate resolution structure interpretation. J Mol Biol 308: 1033-1044.
115. Del Palu A, He J, Pontelli E, Lu Y (2006) Identification of Alpha-Helices from Low Resolution Protein Density Maps. Proceeding of Computational Systems Bioinformatics Conference(CSB): 89-98.
116. Baker ML, Ju T, Chiu W (2007) Identification of secondary structure elements in intermediate-resolution density maps. Structure 15: 7-19.

117. Kong Y, Ma J (2003) A structural-informatics approach for mining beta-sheets: locating sheets in intermediate-resolution density maps. Journal of Molecular Biology 332: 399-413.
118. Kong Y, Zhang X, Baker TS, Ma J (2004) A Structural-informatics approach for tracing beta-sheets: building pseudo-C(alpha) traces for beta-strands in intermediate-resolution density maps. Journal of Molecular Biology 339: 117-130.
119. Lindert S, Staritzbichler R, Wötzel N, Karakaş M, Stewart PL, et al. (2009) EM-Fold: De Novo Folding of α-Helical Proteins Guided by Intermediate-Resolution Electron Microscopy Density Maps. Structure 17: 990-1003.
120. Wu Y, Chen M, Lu M, Wang Q, Ma J (2005) Determining protein topology from skeletons of secondary structures. Journal of Molecular Biology 350: 571-586.
121. Abeysinghe S (2010) A Geometric Approach for Deciphering Protein Structure from Cryo-EM Volumes. St. Louis: Washington University in St. Louis 108 p.
122. Jiang W, Baker ML, Ludtke SJ, Chiu W (2001) Bridging the information gap: computational tools for intermediate resolution structure interpretation. Journal of Molecular Biology 308: 1033-1044.
123. Lasker K, Dror O, Shatsky M, Nussinov R, Wolfson HJ (2007) EMatch: discovery of high resolution structural homologues of protein domains in intermediate resolution cryo-EM maps. IEEE/ACM Transactions on Computational Biology and Bioinformatics 4: 28-39.
124. Si D, Ji S, Al-Nasr K, He J (2012) A machine learning approach for the identification of protein secondary structure elements from cryoEM density maps. Biopolymers 97: 698-708.
125. Kabsch W, Sander C (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22: 2577-2637.
126. Mount DW (2004) Bioinformatics: Sequence and Genome Analysis: Cold Spring Harbor Laboratory Press.
127. Dor O, Zhou Y (2007) Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training. Proteins: Structure, Function, and Bioinformatics 66: 838-845.
128. Lin K, Simossis VA, Taylor WR, Heringa J (2005) A simple and fast secondary structure prediction method using hidden neural networks. Bioinformatics 21: 152-159.
129. Rost B, Sander C (1993) Prediction of protein secondary structure at better than 70% accuracy. Journal of Molecular Biology 232: 584-599.
130. Cole C, Barber JD, Barton GJ (2008) The Jpred 3 secondary structure prediction server. Nucleic Acids Research 36: W197-W201.
131. Jones DT (1999) Protein secondary structure prediction based on position-specific scoring matrices. Journal of Molecular Biology 292: 195-202.
132. Pollastri G, McLysaght A (2005) Porter: a new, accurate server for protein secondary structure prediction. Bioinformatics 21: 1719-1720.

133. Pintiliea GD, Zhangb J, Goddardc TD, Chiub W, Gossard DC (2010) Quantitative analysis of cryo-EM density map segmentation by watershed and scale-space filtering, and fitting of structures by alignment to regions. Journal of Structural Biology 170: 427-438.
134. Chacón P, Wriggers W (2002) Multi-resolution contour-based fitting of macromolecular structures. Journal of Molecular Biology 317: 375-384.
135. Volkmannb N, Hanein D (1999) Quantitative fitting of atomic models into observed densities derived by electron microscopy. Journal of Structural Biology 125: 176-184.
136. Rossmann MG (2000) Fitting atomic models into electron-microscopy maps. Acta Crystallographica Section D, Biological Crystallography 56: 1341-1349.
137. Kovacs JA, Chacón P, Cong Y, Metwally E, Wriggers W (2003) Fast rotational matching of rigid bodies by fast Fourier transform acceleration of five degrees of freedom. Acta Crystallographica Section D, Biological Crystallography 59: 1371-1376.
138. Wriggers W, Chacón P (2001) Modeling tricks and fitting techniques for multiresolution structures. Structure 9: 779-788.
139. Alber F, Eswar N, Sali A (2008) Structure Determination of Macromolecular Complexes by Experiment and Computation. Nucleic Acids and Molecular Biology, 15: 73-96.
140. Volkmann N, Hanein D, Ouyang G, Trybus KM, DeRosier DJ, et al. (2000) Evidence for cleft closure in actomyosin upon ADP release. Nature structural biology 7: 1147-1155.
141. Wriggers W, Milligan RA, McCammon JA (1999) Situs: A package for docking crystal structures into low-resolution maps from electron microscopy. Journal of Structural Biology 125: 185-195.
142. Wriggers W, Agrawal RK, Drew DL, McCammon A, Frank J (2000) Domain motions of EF-G bound to the 70S ribosome: insights from a hand-shaking between multi-resolution structures. Biophysical Journal 79: 1670-1678.
143. Wriggers W, Birmanns S (2001) Using situs for flexible and rigid-body fitting of multiresolution single-molecule data. Journal of Structural Biology 133: 193-202.
144. Tama F, Miyashita O, Brooks CL (2004) Normal mode based flexible fitting of high-resolution structure into low-resolution experimental data from cryo-EM. Journal of Structural Biology 147: 315-326.
145. Suhre K, Navazab J, Sanejouand Y-H (2006) NORMA: a tool for flexible fitting of high-resolution protein structures into low-resolution electron-microscopy-derived density maps. Acta Crystallographica Section D, Biological Crystallography 62: 1098-1100.
146. Ming D, Kong Y, Wakil SJ, Brink J, Ma J (2002) Domain movements in human fatty acid synthase by quantized elastic deformational model. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 99: 7835-7899.
147. Ming D, Kong Y, Lambert MA, Huang Z, Ma J (2002) How to describe protein motion without amino acid sequence and atomic coordinates. Proceedings of the

National Academy of Sciences of the United States of America (PNAS) 99: 8620-8625

148. Fabiola F, Chapman MS (2005) Fitting of High-Resolution Structures into Electron Microscopy Reconstruction Images. Structure 13: 389-400.

149. Velazquez-Muriel J-Á, Valle M, Santamaría-Pang A, Kakadiaris IA, Carazo J-M (2006) Flexible Fitting in 3D-EM Guided by the Structural Variability of Protein Superfamilies. Structure 14: 1115-1126.

150. Velazquez-Muriela JA, Carazo J-Ma (2007) Flexible fitting in 3D-EM with incomplete data on superfamily variability. Journal of Structural Biology 158: 165-181.

151. Schröder GF, Brunger AT, Levitt M (2007) Combining efficient conformational sampling with a deformable elastic network model facilitates structure refinement at low resolution. Structure 15: 1630-1641.

152. Jolley CC, Wells SA, Fromme P, Thorpe MF (2008) Fitting low-resolution cryo-EM maps of proteins using constrained geometric simulations. Biophysical Journal 94: 1613-1621.

153. Wells S, Menor S, Hespenheide B, Thorpe MF (2005) Constrained geometric simulation of diffusive motion in proteins. Physical Biology 2: S127-S136.

154. Li W, Frank J (2007) Transfer RNA in the hybrid P/E state: Correlating molecular dynamics simulations with cryo-EM data. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 104: 16540-16545.

155. Trabuco LG, Villa E, Mitra K, Frank J, Schulten K (2008) Flexible fitting of atomic structures into electron microscopy maps using molecular dynamics. Structure 16: 673-683.

156. Kovacs JA, Yeager M, Abagyan R (2007) Computational Prediction of Atomic Structures of Helical Membrane Proteins Aided by EM Maps. Biophysical Journal 93: 1950-1959.

157. He J, Lu Y, Pontelli E (2004) A Parallel Algorithm for Helix Mapping between 3-D and 1-D Protein Structure using the Length Constraints. Lecture Notes in Computer Science 3358: 746-756.

158. Dal Palu A, Pontelli E, He J, Lu Y (2006) A constraint logic programming approach to 3D structure determination of large protein complexes. Proceedings of the 2006 ACM Symposium on Applied Computing. Dijon, France: ACM. pp. 131-136.

159. Bahar I, Jernigan RL (1997) Inter-residue potentials in globular proteins and the dominance of highly specific hydrophilic interactions at close separation. Journal of Molecular Biology 266: 195-214.

160. Haliloglu T, Bahar I (1998) Coarse-Grained Simulations of Conformational Dynamics of Proteins: Application to Apomyoglobin. Proteins: Structure, Function, and Bioinformatics 31: 271-281.

161. Bahar I, Kaplan M, Jernigan RL (1997) Short-range Conformational Energies, Secondary Structure Propensities, and Recognition of Correct Sequence-Structure Matches. Proteins: Structure, Function, and Bioinformatics 29: 292-308.

162. Abeysinghe S, Ju T, Baker ML, Chiu W (2008) Shape modeling and matching in identifying 3D protein structures. Computer-Aided Design 40: 708-720.

163. Ju T, Baker ML, Chiu W (2007) Computing a family of skeletons of volumetric models for shape description. Computer-Aided Design 39: 352-360.
164. Lindert S, Stewart PL, Meiler J (2009) Hybrid approaches: applying computational methods in cryo-electron microscopy. Current Opinion in Structural Biology 19: 218-225.
165. Soto CS, Fasnacht M, Zhu J, Forrest L, Honig B (2008) Loop modeling: Sampling, filtering, and scoring. Proteins: Structure, Function, and Bioinformatics 70: 834-843.
166. Xiang Z, Soto CS, Honig B (2002) Evaluating conformational free energies: the colony energy and its application to the problem of loop prediction. Proceedings of the National Academy of Sciences of the United States of America (PNAS) 99(11): 7432-7437.
167. Jacobson M, Pincus D, Rapp C, Day T, Honig B, et al. (2004) A hierarchical approach to all-atom protein loop prediction. Proteins: Structure, Function, and Bioinformatics 55: 351-367.
168. Deane C, Blundell T (2001) CODA: A combined algorithm for predicting the structurally variable regions of protein models. Protein Science 10: 599-612.
169. Fiser A, Do RK, Sali A (2000) Modeling of loops in protein structures. Protein Science 9: 1753-1773.
170. Ko J, Lee D, Park H, Coutsias EA, Lee J, et al. (2011) The FALC-Loop web server for protein loop modeling. Nucleic Acids Research 39: W210-W214.
171. Michalsky E, Goede A, Preissner R (2003) Loops in proteins (LIP)—a comprehensive loop database for homology modelling. Protein Engineering 16: 979-985.
172. Jamroz M, Kolinski A (2010) Modeling of loops in proteins: a multi-method approach. BMC Structural Biology 10: 5.
173. Fine RM, Wang H, Shenkin PS, Yarmush DL, Levinthal C (1986) Predicting antibody hypervariable loop conformations. II: Minimization and molecular dynamics studies of MCPC603 from many randomly generated loop conformations. Proteins: Structure, Function, and Bioinformatics 1: 342-362.
174. Shenkin PS, Yarmush DL, Fine RM, Wang HJ, Levinthal C (1987) Predicting antibody hypervariable loop conformation. 1. ensembles of random conformations for ring-like structure. Biopolymers 26: 2053-2085.
175. Liu P, Zhu F, Rassokhin DN, Agrafiotis DK (2009) A Self-Organizing Algorithm for Modeling Protein Loops. PLoS Computational Biology 5: e1000478.
176. DePristo MA, Bakker PIWd, Lovell SC, Blundell TL (2003) Ab initio construction of polypeptide fragments: efficient generation of accurate, representative ensembles. Proteins: Structure, Function, and Bioinformatics 51: 41-55.
177. Rohl CA, Strauss CEM, Chivian D, Baker D (2004) Modeling structurally variable regions in homologous proteins with rosetta. Proteins: Structure, Function, and Bioinformatics 55: 656-677.
178. Zhu K, Pincus D, Zhao S, Friesner R (2006) Long loop prediction using the protein local optimization program. Proteins: Structure, Function, and Bioinformatics 65: 438-452.

179. Mandell DJ, Coutsias EA, Kortemme T (2009) Sub-angstrom accuracy in protein loop reconstruction by robotics-inspired conformational sampling. Nature Methods 6: 551-552.
180. Li Y, Rata I, Jakobsson E (2011) Sampling Multiple Scoring Functions Can Improve Protein Loop Structure Prediction Accuracy. Journal of Chemical Information and Modeling 51: 1656-1666.
181. Li Y, Rata I, Chiu S-w, Jakobsson E (2010) Improving predicted protein loop structure ranking using a Pareto-optimality consensus method. BMC Structural Biology 10: 22.
182. Al-Nasr K, Ranjan D, Zubair M, He J (2011) Ranking Valid Topologies of the Secondary Structure elements Using a constraint Graph. Journal of Bioinformatics and Computational Biology 9: 415-430.
183. Al-Nasr K, Sun W, He J (2010) Structure prediction for the helical skeletons detected from the low resolution protein density map. BMC Bioinformatics 11: S44.
184. He J, Al-Nasr K (2007) An Approximate Robotics Algorithm to Assemble a Loop between Two Helices. The Proceeding of IEEE International Conference on Bioinformatics and Biomedicine Workshops: 74-79.
185. Al-Nasr K, He J (2009) An effective convergence independent loop closure method using Forward-Backward Cyclic Coordinate Descent. International Journal of Data Mining and Bioinformatics 3: 346-361.
186. Lindert S, Staritzbichler R, Wotzel N, Karakas M, Stewart PL, et al. (2009) EM-fold: De novo folding of alpha-helical proteins guided by intermediate-resolution electron microscopy density maps. Structure 17: 990-1003.
187. Baker ML, Abeysinghe SS, Schuh S, Coleman RA, Abrams A, et al. (2011) Modeling protein structure at near atomic resolutions with Gorgon. Journal of Structural Biology 174: 360-373.
188. Dunbrack RL (2002) Rotamer libraries in the 21st century. Current opinion in structural biology 12: 431-440.
189. Dunbrack RL, Karplus M (1993) Backbone-dependent Rotamer Library for Proteins: Application to Side-chain prediction. Journal of Structural Biology 230: 543-574.
190. Xie W, Sahinidis NV (2006) Residue-rotamer-reduction algorithm for the protein side-chain conformation problem. Bioinformatics 22: 188-194.
191. Eppstein D (1998) Finding the k Shortest Paths. SIAM Journal on Computing 28: 652-673.
192. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numerische Mathematik 1: 269-271.
193. Bellman R (1958) On a routing problem. Quarterly of Applied Mathematics 16: 87-90.
194. Ford LR, Fulkerson DR (1956) Maximal flow through a network. Canadian Journal of Mathematics 8: 399-404.
195. Floyd RW (1962) Algorithm 97: Shortest path. Communications of the ACM 5: 345.
196. Warshall S (1962) A Theorem on Boolean Matrices. Journal of the ACM 9: 11-12.

197. Johnson DB (1977) Efficient Algorithms for Shortest Paths in Sparse Networks. Journal of the ACM 24: 1-13.
198. Hoffman W, Pavley R (1959) A Method for the Solution of the Nth Best Path Problem. Journal of the ACM 6: 506-514.
199. Pollack M (1961) The kth Best Route through a Network. Operations Research 9: 578-580.
200. Dreyfus SE (1969) An Appraisal of Some Shortest-Path Algorithms. Operations Research 17: 395-412.
201. Brander AW, Sinclair M. A comparative study of k-shortest path algorithms; 1995.
202. Yen JY (1971) Finding the K Shortest Loopless Paths in a Network. Management Science 17: 712-716.
203. Hershberger J, Maxel M, Suri S (2007) Finding the *k* shortest simple paths: A new algorithm and its implementation. ACM Transactions on Algorithms 3: 45.
204. Martins EdQV, Pascoal MMB, Santos JLEd (1999) Deviation Algorithms for Ranking Shortest Paths. International Journal of Foundation of Computer Science 10: 247-263.
205. Martins EQV, Pascoal MMB (2003) A new implementation of Yen's ranking loopless paths algorithm. 4OR: A Quarterly Journal of Operations Research 1: 121-133.
206. Lawler EL (1972) A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem. Management Science 18: 401-405.
207. Cong Y, Zhang Q, Woolford D, Schweikardt T, Khant H, et al. (2009) Structural Mechanism of SDS-Induced Enzyme Activity of Scorpion Hemocyanin Revealed by Electron Cryomicroscopy. Structure 17: 749-758.
208. Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. Communications of the ACM 16: 575-577.
209. Abeysinghe SS, Baker M, Wah C, Tao J. Segmentation-free skeletonization of grayscale volumes for shape understanding; 2008. pp. 63-71.
210. Wedemeyer WJ, Scheraga HA (1999) Exact analytical loop closure in proteins using polynomial equations. Journal of Computational Chemistry 20: 819-844.
211. Raghavan M, Roth B (1989) Kinematic analysis of the 6r manipulator of general geometry. International Symposium on Robotics Research. pp. 314-320.
212. Palmer KA, Scheraga HJ (1991) Standard-geometry chains fitted to x-ray derived structures : Validation of the rigid-geometry approximation. I. chain closure through a limited search of loop conformations. Journal Computational Chemistry 2: 505-526.
213. Shehu A, Clementi C, Kavraki LE (2006) Modeling protein conformational ensembles: From missing loops to equilibrium Fluctuations. Proteins: Structures, Functions, and Bioinformatics 65: 164-179.
214. Coutsias EA, Seok C, Jacobson MP, Dill KA (2004) A kinematic view of loop closure. Journal of Computational Chemistry 25: 510-528.
215. Go N, Scheraga HA (1970) Ring closure and local conformational deformations of chain molecules. Macromolecules 3: 178-186.

216. Kolodny R, Guibas L, Levitt M, Koehl P (2005) Inverse kinematics in biology:The protein loop closure problem. International Journal of Robotics Research 24: 151.
217. Manocha D, Canny J (1994) Efficient inverse kinematics for general 6R manipulator. IEEE Transactions on Robotics and Automation 10: 648-657.
218. Zhang M, White RA, Wang L, Goldman R, Kavraki L, et al. (2005) Improving conformational searches by geometric screening. Bioinformatics 21: 624-630.
219. Canutescu AA, Dunbrack RLJ (2003) Cyclic coordinate descent: A robotics algorithm for protein loop closure. Protein Science 12: 963-972.
220. Craig JJ (1989) Introduction to robotics: manipulation and control.: Addison-Wesley.
221. Boomsma W, Hamelryck T (2005) Full cyclic coordinate descent: Solving the protein loop closure problem in Cα space. BMC Bioinformatics 6: 159.
222. Wang C, Bradley P, Baker D (2007) Protein-protein docking with backbone flexibility. Journal of Molecular Biology 373: 503-519.
223. Shehu A, Kavraki L (2012) Modeling Structures and Motions of Loops in Protein Molecules. Entropy 14: 252-290.
224. Jiang H, Blouin C (2006) Ab initio construction of all-atom loop conformations. Journal of Molecular Biology 12: 221-228.
225. Kavraki LE, Svestka P, Latombe JC, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Tansactions on Robotics and Automation 12: 566-580.
226. Dawen X, Amato NM. A kinematics-based probabilistic roadmap method for high DOF closed chain systems; 2004. pp. 473-478.
227. Cortes J, Simeon T, Laumond JP. A random loop generator for planning the motions of closed kinematic chains using PRM methods; 2002. pp. 2141-2146.
228. Cortés J, Siméon T, Remaud-Siméon M, Tran V (2004) Geometric algorithms for the conformational analysis of long protein loops. Journal of Computational Chemistry 25: 956-967.
229. Yakey JH, LaValle SM, Kavraki LE (2001) Randomized path planning for linkages with closed kinematic chains. IEEE Transactions on Robotics and Automation 17: 951-958.
230. Trinkle JC, Milgram RJ (2002) Complete Path Planning for Closed Kinematic Chains with Spherical Joints. The International Journal of Robotics Research 21: 773-789.
231. Lotan I, Van Den Bedem H, Deacon AM, Latombe J-C (2005) Computing Protein Structures from Electron Density Maps: The Missing Fragment Problem Algorithmic Foundations of Robotics VI. In: Erdmann M, Overmars M, Hsu D, van der Stappen F, editors: Springer Berlin / Heidelberg. pp. 345-360.
232. Lindert S, Alexander N, Wötzel N, Karaka M, Stewart Phoebe L, et al. (2012) EM-Fold: De Novo Atomic-Detail Protein Structure Determination from Medium-Resolution Density Maps. Structure 20: 464-478.
233. Biswas A, Si D, Al-Nasr K, Ranjan D, Zubair M, et al. (2011) A Constraint Dynamic Graph Approach to Identify the Secondary Structure Topology from cryoEM

Density Data in Presence of Errors. Proceedings of the 2011 IEEE International Conference on Bioinformatics and Biomedicine: IEEE Computer Society. pp. 160-163.

234. Carugo O, Pongor S (2001) A normalized root-mean-spuare distance for comparing protein three-dimensional structures. Protein Science 10: 1470-1473.

VITA

Kamal H. Al Nasr
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

Kamal H. Al Nasr was born in Irbid city, Jordan on February 1980. After completing his high school at Irbid Secondary School, Irbid in 1998, he attended Yarmouk University, Irbid in 1999, receiving his bachelor degree of science in Computer Science on February 2003. In his Bachelor study, he was listed on the honor list of the college of science. He entered the graduate school in the Department of Computer Information Systems at Yarmouk University, Irbid in 2003, receiving a Master's of Computer Information Systems in August, 2005. He ranked the 2nd out of the students who graduated in the same academic year. While attending the graduate school at Yarmouk University, he served as a Math/Computer teacher in United Nations schools. He moved to United stated in 2007, entering the graduate school in the Department of Computer Science at New Mexico State University, New Mexico. He Obtained a Master's of Science in Computer Science in 2011. During his stay at NMSU, he worked in the Center for Research Excellence in Bioinformatics and Computational Biology (CREST). He was awarded a graduate assistantship excellence award for his research contribution and outstanding. He joined the graduate school in the Department of Computer Science at Old Dominion University, Virginia, in the Ph.D. program in September, 2009. He awarded College of Science's university fellowship on July 2010. While in Ph.D. program, he served as an instructor of one of undergraduate courses. Mr. Al Nasr has multiple publications in many national and international refereed journals and conferences such as BMC Bioinformatics, Journal of Bioinformatics and Computational Biology, International Journal of Bioinformatics and Data Mining, and IEEE BIBM conference.