Summer 2014

Improving Structural Features Prediction in Protein Structure Modeling

Ashraf Yaseen
Old Dominion University

IMPROVING STRUCTURAL FEATURES PREDICTION

IN PROTEIN STRUCTURE MODELING

by

Ashraf Yaseen
M.Sc. July 2003, New York Institute of Technology, Old Westbury, NY
B.Sc. August 2002, Jordan University of Science and Technology, Jordan

A Thesis 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, 2014

Approved by:

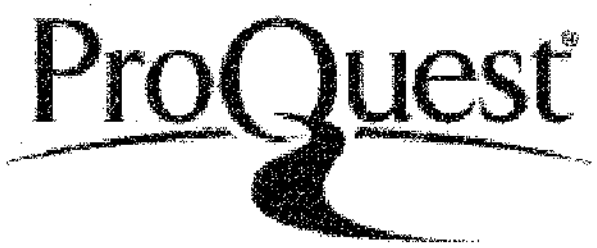
Yaohang Li (Director)

Hussein Abdel-Wahab (Member)

Kurt Maly (Member)

Desh Ranjan (Member)

Jiang Li (Member)



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

IMPROVING STRUCTURAL FEATURES PREDICTION
IN PROTEIN STRUCTURE MODELING

Ashraf Yaseen
Old Dominion University, 2014
Director: Dr. Yaohang Li

Proteins play a vital role in the biological activities of all living species. In nature, a protein folds into a specific and energetically favorable three-dimensional structure which is critical to its biological function. Hence, there has been a great effort by researchers in both experimentally determining and computationally predicting the structures of proteins.

The current experimental methods of protein structure determination are complicated, time-consuming, and expensive. On the other hand, the sequencing of proteins is fast, simple, and relatively less expensive. Thus, the gap between the number of known sequences and the determined structures is growing, and is expected to keep expanding. In contrast, computational approaches that can generate three-dimensional protein models with high resolution are attractive, due to their broad economic and scientific impacts. Accurately predicting protein structural features, such as secondary structures, disulfide bonds, and solvent accessibility is a critical intermediate step stone to obtain correct three-dimensional models ultimately.

In this dissertation, we report a set of approaches for improving the accuracy of structural features prediction in protein structure modeling. First of all, we derive a statistical model to generate context-based scores characterizing the favorability of segments of residues in adopting certain structural features. Then, together with other information such as evolutionary and sequence information, we incorporate the context-based scores in machine learning approaches to predict secondary structures, disulfide bonds, and solvent accessibility. Furthermore, we take advantage of the emerging high performance computing architectures in GPU to accelerate the calculation of pairwise and high-order interactions in context-based scores. Finally, we make these prediction methods available to the public via web services and software packages.

This thesis is dedicated to my parents (***my Dad and my Mom**; I love you both*)
and to my brothers and sisters (***I miss you all***).

ACKNOWLEDGMENTS

First of all, I would like to express my genuine appreciations to my advisor and mentor, whom I owe a great debt of gratitude for his patience, inspiration and friendship, Dr. Yaohang Li. Thank you Dr. Li for your kind help and continuous encouragement in my Ph.D. research and study. Also, I would like to give special thanks to Dr. Hussein Abdel-Wahab for his help and support, especially during my difficult times.

I would also like to give thanks to my committee members: Dr. Kurt Maly, Dr. Hussein Abdel-Wahab, Dr. Jing He, Dr. Desh Ranjan, and Dr. Jiang Li, for their helpful comments and advice.

I would like to thank my colleagues and my friends for their help and support. Special thanks to my good friends Yahya Al-Mansour for his help and Sami Al-khub for his kindness.

I also would like to acknowledge the support from NSF under Dr. Li's grant 1066471 and ODU 2013 Multidisciplinary Seed grant. Finally, I would like to thank the Department of Computer Science at Old Dominion University for providing an excellent environment for my research. Without this rich environment I doubt that many of my ideas would have come to completion.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES x

CHAPTER

1. INTRODUCTION 1
   1.1 PROBLEM DEFINITION 2
   1.2 OUR APPROACHES 3
   1.3 BACKGROUND 4
      1.3.1 PROTEIN STRUCTURE 7
      1.3.2 PROTEIN STRUCTURE DETERMINATION 9
      1.3.3 PROTEIN STRUCTURE PREDICTION 15
      1.3.4 PROTEIN STRUCTURAL FEATURES PREDICTION 18
   1.4 DISSERTATION ORGANIZATION 19

2. LITERATURE REVIEW 20
   2.1 HISTORY OF PROTEIN STRUCTURAL FEATURE PREDICTION 20
      2.1.1 1ST GENERATION: STATISTICS-BASED METHODS 20
      2.1.2 2ND GENERATION: MACHINE LEARNING-BASED METHODS 21
      2.1.3 3RD GENERATION: PREDICTION WITH EFFECTIVE FEATURES 21
   2.2 ANALYSIS OF PROTEIN STRUCTURAL FEATURE PREDICTION 24
   2.3 GPU-ACCELERATION 25
      2.3.1 GPU APPLICATIONS IN GENERAL PURPOSE COMPUTING 25
      2.3.2 GPU APPLICATIONS IN ENERGY EVALUATIONS 26

3. CONTEXT-BASED MODEL 28
   3.1 MODEL OVERVIEW 28
      3.1.1 DATASETS 29
      3.1.2 MULTIPLE SEQUENCE ALIGNMENT 30
   3.2 MODEL IMPLEMENTATION 30
      3.2.1 DERIVATION OF CONTEXT-BASED SCORES 31
      3.2.2 MACHINE LEARNING ALGORITHM 32
   3.3 METHODS OF EVALUATION 32

4. SECONDARY STRUCTURE PREDICTION 34
   4.1 C3-SCORPION 35

4.1.1 METHOD IMPLEMENTATION....35
4.1.2 RESULTS OF C3-SCORPION....40
4.1.3 DISCUSSIONS....43
4.2 C8-SCORPION....46
4.2.1 METHOD IMPLEMENTATION....47
4.2.2 RESULTS OF C8-SCORPION....47
4.2.3 DISCUSSIONS....49
4.3 TEMPLATE-BASED SCORPION....51
4.3.1 METHOD IMPLEMENTATION....51
4.3.2 RESULTS OF TEMPLATE-BASED SCORPION....54
4.3.3 DISCUSSIONS....56

5. DISULFIDE BONDING PREDICTION....58
5.1 DINOSOLVE....58
5.1.1 METHOD IMPLEMENTATION....59
5.1.2 RESULTS OF DINOSOLVE....65
5.1.3 DISCUSSIONS....68

6. SOLVENT ACCESSIBILITY....70
6.1 CASA....70
6.1.1 METHOD IMPLEMENTATION....70
6.1.2 RESULTS OF CASA....73
6.1.3 DISCUSSIONS....75

7. GPU-ACCELERATION OF MANY-BODY POTENTIALS....77
7.1 ACCELERATING 2-BODY POTENTIALS....77
7.1.1 GPU-DFIRE IMPLEMENTATION DETAILS....78
7.1.2 COMPUTATIONAL RESULTS....88
7.2 ACCELERATING 3-BODY POTENTIAL....93
7.2.1 3-BODY EFFECTS....93
7.2.2 GPU-BASED LOAD-BALANCING SCHEME FOR COMPUTING 3-BODY INTERACTIONS....94
7.2.3 COMPUTATIONAL RESULTS....106

8. SUMMARY AND POSTDISSERTATION REASEARCH....113

REFERENCES....117

VITA....127

LIST OF TABLES

Table Page

1. DSSP secondary structure states 12
2. Characteristics of CullPDB lists 29
3. 7-fold cross validation Q3 and SOV3 accuracies for 3-state prediction in SCORPION 41
4. Q3 accuracy for each amino acid type in SCORPION 41
5. Comparison of Q3 accuracy between SCORPION and other popularly used secondary structure prediction methods including Porter (ab initio), PsiPred, ProfPHD, NetSurfp, and JPred on benchmarks of CB513, CASP9, Manesh215, and Carugo338 42
6. Comparison of SOV3 accuracy between SCORPION and other popularly used secondary structure prediction servers including Porter (ab initio), PsiPred, ProfPHD, NetSurfp, and JPred on benchmarks of CB513, CASP9, Manesh215, and Carugo338 43
7. Total number of correct predictions with over 90% confidence on benchmarks of CB513, CASP9, Manesh215, and Carugo338 44
8. Segment of 3NNQA (82-115) 45
9. Misclassifications of secondary structure states on benchmark sets 46
10. 7-fold cross validation accuracy for 8-state prediction in SCORPION 48
11. Comparison of Q8 accuracy between SCORPION and RaptorXss8 on benchmarks of CB513, CASP9, Manesh215, and Carugo338 48
12. Comparison of SOV8 accuracy between SCORPION and RaptorXss8 on benchmarks of CB513, CASP9, Manesh215, and Carugo338 49
13. Detailed description of Helix and Strand misclassifications on benchmark set of CB513, CASP9, Manesh215, and Carugo338 50
14. 7-fold cross-validation accuracy in template-based 8-state prediction 54
15. Comparison between 8-state predictions with and without template on CB513, CASP9, Manesh215, and Carugo338 54
16. Comparison of 7-fold cross validation prediction accuracies in eight states when templates with different sequence similarities are used 57

17. Comparison of prediction performance of bonding states using PSSM only and PSSM with context-based scores on Cull25 and Cull50 using 10-fold cross validation 65

18. Computational results of 10-fold cross validation on Cull50 using PSSM only and PSSM + Score in neural network encoding 66

19. Prediction performance on protein chains in Manesh215, Carugo338, and CASP9...66

20. Predicted bonding probability for potential disulfide bonds in 153L(A) 67

21. Comparison of residue-level accuracies (Qc) on benchmarks of Manesh215, Carugo338, and CASP9 using Cull25 and Cull50 as training sets.....68

22. Extended state values of amino acids 71

23. Comparison of prediction performance of Solvent Accessibility using PSSM only and PSSM with context-based scores on Cull7987 using 7-fold cross validation.....74

24. Comparison of the accuracy between our method (CASA) and other popularly used solvent accessibility prediction servers including NETASA, Sable, Netsurf, SPINE, and ACCpro on benchmarks of CASP9, Manesh215, and Carugo338 74

25. Performance between sorted and unsorted atom sequences in proteins of various sizes 83

26. Percentages of energy evaluation times in Monte Carlo sampling program and MINIROT using the CPU-only implementations and the heterogeneous CPU–GPU implementations with energy function evaluation on GPU 92

27. Performance comparison of MINIROT program on 6 initial structures of 2ERL (319 atoms) using GPU–DFIRE and CPU–DFIRE. GPU–DFIRE is carried out on Tesla M2050 server 92

28. Comparison of the Number of divergent branches in GPU-AxT with half box cutoff distance and GPU-AxT without cutoff 110

LIST OF FIGURES

Figure Page

1. Ramachandran plot 2
2. The generic structure of an amino acid 5
3. Two amino acids forming a peptide bond 5
4. Amino Acids Properties (Venn diagram) 6
5. Distribution of secondary structure states in Cull5547 9
6. Protein structure hierarchy 9
7. Hydrogen bonding patterns in helical types 11
8. Hydrogen bonding patterns in Parallel & Anti parallel β-sheets 12
9. 3D structure of Protein 1BOO Chain A 13
10. Disulfide bond in protein chain 14
11. Surface area of a protein segment 14
12. The proposed methodologies in predicting protein structural features 28
13. Distribution of the structural states in the Cull16633 30
14. Context-based Model 31
15. The probability of Alanine as the middle residue of a triplet with neighboring residues at 1~5 positions away when adopting α-helix as secondary structure 36
16. Three phases of prediction (architecture and encoding) for C3-SCORPION 39
17. 3-state secondary structure prediction for protein 3NNQ chain A from CASP9 targets 45
18. Histograms of the numbers of misclassified helices (A) and strands (B) in SCORPION by their lengths 50
19. Construction of Templates 52
20. Encoding for template-based 8-state secondary structure prediction 53

21. Two phases of template-based 8-state secondary structure prediction (architecture & encoding) 53

22. Distribution of 8-state secondary structure prediction accuracy (Q8) as a function of sequence similarity 55

23. Comparison between template-less and template-based predictions on 1BTN chain A 55

24. Probability of Cysteine in disulfide bonding state with neighbors at different positions 59

25. Three possible positions of two neighbors to a Cysteine residue 60

26. Possible positions of two neighbors to a Cysteine residue pair in disulfide bond 61

27. Encoding and neural network architecture for disulfide bonding state prediction. 64

28. Encoding and neural network architecture for disulfide connectivity prediction 64

29. Disulfide connectivity prediction on protein 153L A chain 67

30. 10-fold cross validated accuracies using context-based scores generated with different window sizes 69

31. The probability of Lysine (K) as the middle residue of a triplet with neighboring residues at 1~3 positions away when adopting buried accessibility state 72

32. Encoding and neural network architecture for 2-state solvent accessibility prediction 73

33. Solvent Accessibility Prediction on protein 3NRF(A) 76

34. DFIRE subroutine for pair-wise atom interaction calculation 79

35. Pseudocode of symmetric N-body calculation in serial DFIRE 80

36. Pseudocode of workload assignment in GPU-DFIRE 80

37. Perfect balancing when N is an odd number 81

38. Nearly perfect balancing when *N* is an even number 82

39. Reordering atoms in a 6-residue protein fragment according to atom types 82

40. Effectiveness of GPU-DFIRE with fine-grained threads in small proteins (Tesla M2050) 85

41. Implementation of GPU–DFIRE on GPU memory hierarchy (Fermi architecture) ...87

42. Effect of loop unrolling in GPU–DFIRE performance (Tesla M2050) on a set of 84 proteins ranges from 318 atoms to 99 595 atoms 88

43. Overall speedup of GPU–DFIRE on Tesla M2050 and Quadro FX3800M on a set of proteins with various sizes with respect to CPU–DFIRE 89

44. Pseudocode of calculating three-body terms in a system with N particles 95

45. An example with $N = 6$ with thread 1 starting from P1 and thread 6 from P6 97

46. Pseudocode of enumerating all non-rotational symmetric position separation patterns except for the order three symmetric one and calculating three-body interaction of the corresponding triplet particles in a GPU thread 98

47. An example of enumerating triplets without sharing rotational symmetry in position separation patterns by the first thread (thread 1) in a system with 10 particles 100

48. Order three rotational symmetry in a system with 6 particles 101

49. Pseudocode of load-balancing workload distribution scheme 102

50. Workload Distribution Scheme when N = 7, 8, 9 105

51. Overall speedup of GPU–AxT on Tesla C2070 on a set of systems with various sizes with respect to CPU–AxT 108

52. Speedup of GPU-AxT Implementation using load-balancing scheme over that of direct mapping 109

53. Performance of the balanced GPU-CSSP with respect to the unbalanced GPU-CSSP on Tesla C2070 112

54. Overall speedup of GPU–CSSP on Tesla C2070, Tesla C1060 and Tesla C870 on a set of proteins with various sizes with respect to CPU–CSSP 112

55. Distribution of context-based scores accuracy and its correlation with accuracy improvement over PSSM only prediction in CB513, CASP9, Manesh215, and Carugo338 benchmarks 115

56. Secondary Structures Predicted by SCORPION on Thylakoid soluble phosphoprotein TSP9 116

CHAPTER 1

INTRODUCTION

Understanding the implication between the genetic code and life could have practically infinite applications, from ad-hoc drug synthesis to the creation of forms of life with desired characteristics. However, we are still far from reaching this goal. Research in this direction is still at its early stages, and one of the biggest open problems nowadays is the prediction of protein structure.

Proteins are the primary components of living things. In nature, proteins fold into specific three-dimensional structures which are critical to their functions. Therefore, there has been a great interest by researchers in both experimentally determining and computationally predicting the 3D structures of proteins.

The current experimental methods of protein structure determination are complicated, time-consuming, and expensive. In contrast, computational approaches that can generate 3D protein models with high resolution are attractive due to their broad economic and scientific impacts. Correctly predicting structural features such as secondary structures, disulfide bonds, and solvent accessibility, which we refer to as the intermediate prediction steps, is a critical step stone to obtain correct 3D models ultimately.

To illustrate the effectiveness of the intermediate prediction steps, we hereby use protein secondary structure as an example. The prediction of secondary structure is an important intermediate step; often viewed as a simplification of the more challenging problem of the tertiary structure perdition. Correct prediction of protein secondary structures can significantly reduce the degrees of freedom in protein tertiary structure modeling and therefore reduces the difficulty of obtaining high resolution 3D models. For example, if a segment in a protein is predicted to be an α-helix, we can take advantage of the Ramachandran plot, shown in Figure 1, to derive a much smaller range of possible torsion angles that can be assigned to the backbone of the predicted segment, or we can simply use the ideal values of the torsion angles to build a helix.

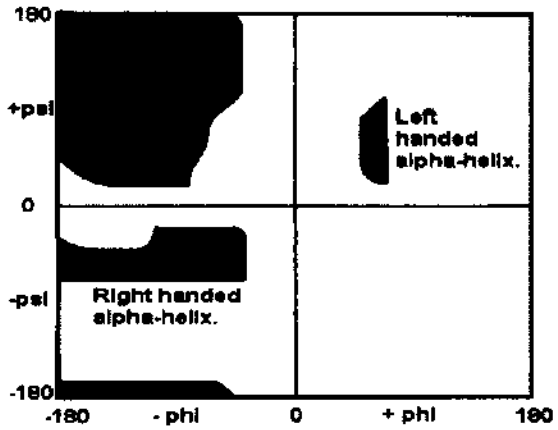


Figure 1 Ramachandran plot. Source: http://www.cryst.bbk.ac.uk/PPS95/course/3_geometry/rama.html.

Similar to secondary structure prediction, correctly predicting the formation and connectivity of disulfide bonds and the residues' solvent accessibility can reduce the conformational space to aid modeling protein structures and help predict important protein functions.

In conclusion, the knowledge of those structural features of protein residues, when predicted with high accuracy, can provide extremely valuable information for predicting protein 3D structure and function [1].

1.1 Problem Definition

Historically, the improvement of structural features prediction methods benefits from the incorporation of effective features/information that helps separate among the different structural states. Nowadays, almost all modern prediction methods make use of evolutionary information revealed by multiple sequence alignment (MSA) of a family of homologues proteins. This information forms the input encodings to a machine learning algorithm, trained to recognize and discriminate the different structural states.

The accuracy of the current prediction methods is stagnated for the past few years and obtaining improvements of even fractions of a percent is becoming very difficult. For example, the accuracy of secondary structure prediction is stagnated between 76-80% in 3-state and ~68% in 8-state. Furthermore, most of the reported accuracies from different

structural features prediction methods are not cross-validated and/or obtained from several small datasets.

This dissertation work targets the research problem of how to continuously improve the accuracy of predicting protein structural features toward their theoretical upper bounds. The objective of this dissertation is to improve predictions of protein structural features to a new level of accuracy.

Reducing the inaccuracy of protein structural features prediction, will be very useful in improving the efficiency of protein tertiary structure prediction, because the search space for finding a tertiary structure goes up super-linearly with the fraction of inaccuracy in structural feature prediction.

1.2 Our Approaches

In machine learning, it is well-known that extracting and selecting "good" features can significantly enhance the prediction performance of a predictor. Probably the most effective features, when predicting the structural states of residues, are the structural states of the neighboring residues (residue's context). For example, a residue is likely exposed to solvent if the neighboring residues are highly exposed to solvent. Moreover if both adjacent neighbors are helices, the middle residue is most likely to be helix, and vice versa; if the adjacent positions of a residue are not helices, it is impossible for this middle residue to adopt helix as its secondary structure. In fact, our computational results show that if the true secondary structures of neighboring residues are encoded, machine learning using a simple feed-forward neural network can easily lead to above 90% prediction accuracy, which exceeds the theoretical upper bound.

Unfortunately, using the true structural states as features is not feasible since they cannot be known beforehand. However, this inspires us that the favorability of a residue adopting a certain structural state can be also an effective feature. The statistical scores measuring the favorability of a residue adopting a certain structural state within its amino acid environment can be evaluated from the experimentally determined protein structures in the Protein Data Banks (PDB). Encoding these scores as features provides a way to address a long standing difficulty in machine learning methods, such as neural networks, of taking interdependency among structural states of neighboring residues into account.

Previously, deriving statistically meaningful context-based scores to characterize high-order inter-residue interactions was not feasible, until recently, where sufficient numbers of experimentally determined protein structures are available in PDB. Moreover, recent developments in GPUs with massively parallel computing mechanism offer attractive opportunities to take advantage of the parallelism at the chip level to achieve high performance computing. Implementation of an efficient, load balanced algorithm for *N*-body interaction calculation on GPU will speed up the calculation of context-based scores from high order inter-residue statistics.

In this work we derive a statistical model to generate context-based scores. The context-based scores indicate the favorability of a residue adopting a structural state in presence of its neighboring residues in sequence. These scores are incorporated as sequence-structure features together with other sequence and evolutionary information in machine learning approaches to predict secondary structures in 3-state and 8-state, disulfide bonding states and bonding connectivity, and solvent accessibility. We validate our methods on several commonly used benchmarks and compare our results with a set of popular structural features prediction methods. Furthermore, we propose an approach for accelerating high-order inter-residue interaction calculations by taking advantage of the GPU architecture. Finally, we develop web services hosting our prediction methods for the protein structure research community.

1.3 Background

The word "Protein" comes from the Greek word "Proteios" which means "primary", or "of prime importance." Proteins are the primary components of living things and the vital organism parts on the planet, making more than half of dry weight in every cell. They are the molecules in charge of the essential functions of cells. For example, they provide the infrastructure that holds a creature together (structural support); they are enzymes that make the chemical reactions necessary for life possible; they are the switches that control gene expression; they are the sensors that see, taste and smell, and the effectors that make muscles move.

Proteins are complex molecules consisting of linear sequences of smaller molecules called amino acids. A protein chain may contain from a few dozens to

thousands of amino acids. The term 'residue' is used to refer to an amino acid molecule integrated into the protein chain.

There are twenty types of amino acids in nature. Each is referred to by three letters code or one letter for short. An amino acid is made of a central carbon atom (Cα), a hydrogen atom attached to Cα, an amine group (H2N), a carboxyl group (COOH) and a side chain (R) that differs from one amino acid to another. Figure 2 shows the generic structure of an amino acid. Amino acids in a protein chain (residues) are connected by peptide bonds. Each bond is formed between the Carbon (C) atom from carboxyl group of one amino acid and the Nitrogen (N) atom from amine group of the following amino acid [2]. Figure 3 depicts the formation of a peptide bond between two residues.

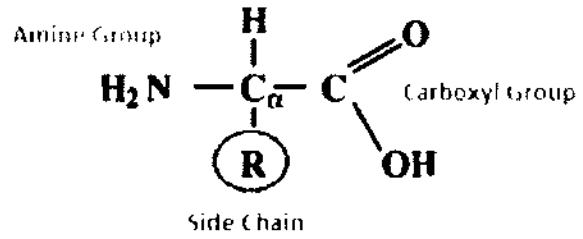


Figure 2 The generic structure of an amino acid.

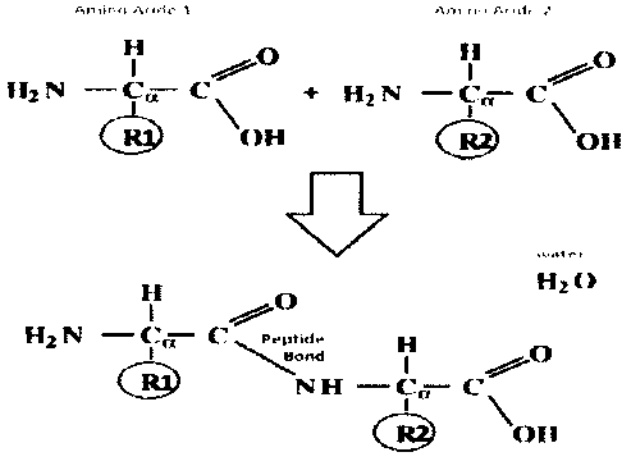


Figure 3 Two amino acids forming a peptide bond.

The composition of the side chain characterizes an amino acid -- it determines the shape, the mass, the volume and the chemical properties of an amino acid. According to the properties of the side chains, amino acids are classified into small/large, polar/non-polar, hydrophobic/hydrophilic, aromatic or aliphatic. These properties play a significant role in protein structure folding and protein-protein interactions. For example, hydrophobic amino acids are usually buried in the middle of the protein 3D structure, while hydrophilic amino acids are likely to be exposed to solvent. Figure 4 is a classical Venn diagram grouping amino acids according to their properties. Original representation and information can be found in [2, 3].

The different groups shown in Figure 4 are:

- Polar/Non-Polar: a non-polar molecule has a relatively even distribution of charge. Some polar amino acids are positively or negatively charged in solution.
- Hydrophobic/Hydrophilic: hydrophobic residues tend to come together to form compact core that exclude water; they tend to be on the inside of a protein, rather than on its surface. Hydrophilic residues are the opposite.

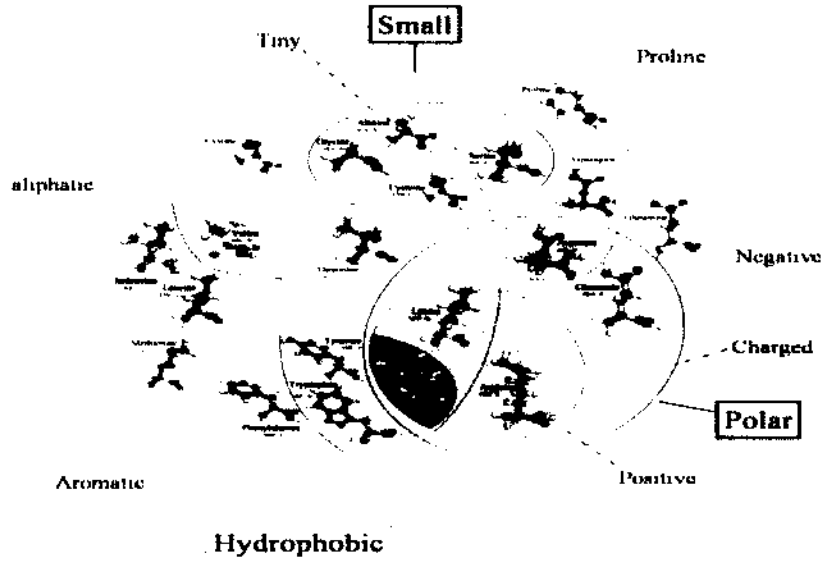


Figure 4 Amino Acids Properties (Venn diagram).
Source: http://www.dsimb.inserm.fr/~debrevern/VENN_DIAGRAM/.

- Aromatic: aromatic amino acid forms closed rings of carbon atoms with alternating double bonds
- Aliphatic: aliphatic amino acids side chain contains only carbon or hydrogen atoms

The differences in the physico-chemical properties of amino acids result in different 3D proteins' structures. Other aspects like electrostatic interactions, van der Waals forces, and hydrogen bonding among the different amino acids are important forces driving the protein folding process. Each different combination of amino acids will result in different protein structural conformations, whose energy values are determined by the interactions among the different amino acids within the structure as well as the interactions between the amino acids and the surrounding solvent.

1.3.1 Protein Structure

According to Anfinsen's thermodynamics hypothesis, under certain chemical and physical conditions, a protein (at least globular protein) folds into a very specific structure, the same structure every time, such that this structure is in the most stable state it can adopt. This unique structure is referred to as the native structure. The sequence of amino acids, building up the protein, ultimately determines its native structure [4].

Proteins start functioning when they are interacting with the cell molecules, or with each other. In order for such interaction to take place and be effective, proteins need to recognize other molecules and bind to them, like a lock and a key. Therefore, protein structure determines its biological function.

Protein Structure can be expressed in different hierarchal levels:

1. Primary Structure: refers to the linear amino acid sequence in the protein. It shows the order in which the amino acids are connected by peptide bonds forming the protein chain. The two ends of the protein chain 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.
2. Secondary structure: refers to the general 3D form of the protein local segments (with no description of the atomic coordinates in 3D space), formally defined by the patterns of hydrogen bonds between backbone amide (H-N) and carboxyl groups (C=O), where the classification of secondary structure elements is determined by recognizing these hydrogen bonding patterns.

The two most common stable secondary structure elements are α-helices and β-sheets; whose stability comes from the regular patters of hydrogen bonds.

a. α-Helix: stabilized by H-bonding between N-H group and C=O group of peptide bonds four residues apart. Its orientation produces a helical coiling of the peptide backbone such that the side chain groups stem out of the helix and perpendicular to its axis. Some amino acids prefer forming alpha helices (helix formers) and others do not (helix breakers), due to some constraints of their side chains. For example, Alanine, Asparatic Acid, Glutamic Acid, Isoleucine, Leucine and Methionine favor the formation of α-helices, whereas, Glycine and Proline favor disruption of the helix. Among the different types of proteins local structures, α-helix is the most regular type.
b. β-sheet: is the second common conformation. It is composed of two or more different segments (strands) of stretches along the primary structure of the protein. Beta strands of 3 to 10 amino acids long are connected by at least 2 to 3 backbone hydrogen bonds. Beta 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.

There are also other less common forms of helices and strands, such as the 3_{10}-helix, π-helix, and β-bridge. Other forms of secondary structure elements are turns and bends, linking the more regular secondary structure elements. Turns refer to the close approach of two consecutive Cα atoms (less than 7 Å) in which no hydrogen bonds are formed, and bends are high twists in the protein chain. Finally, random coils refer to any secondary structure type that lacks a regular form.

Figure 5 presents the composition percentage of the secondary structure states in protein dataset Cull5547 [5], which contains 5,547 protein chains, 25% pair-wise sequence identity, and 2.0 Å resolution. In general, π-helices and β-bridges have relatively infrequent appearances in protein data banks (PDB).

3. Tertiary Structure: refers to the complete 3D structure of a single protein molecule. Protein tertiary structure is stabilized by hydrophobic interactions, hydrogen bonding, disulfide bonding (in some proteins), and non-bonded interactions including electrostatic interactions, and Van Der Waals forces.

Figure 6 shows the first 3 levels of protein structure hierarchy.

4. Quaternary Structure: refers to multiple polypeptide chains that may form the protein molecule. The quaternary structure is stabilized by the same interactions as the tertiary structure.

1.3.2 Protein Structure Determination

The efforts of understanding protein structures and functions started in the 1950s. Biochemists were trying to define the relationship between protein sequences and their different chemical properties.

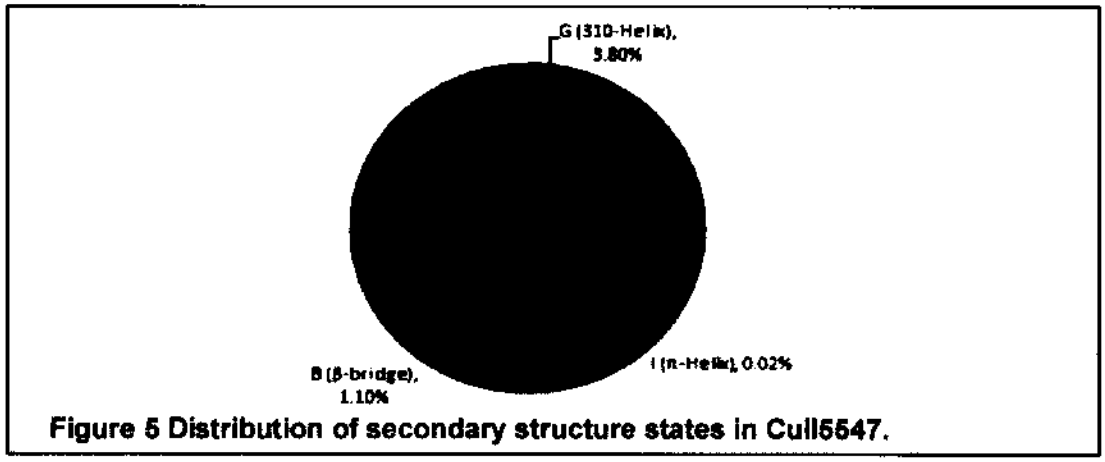


Figure 5 Distribution of secondary structure states in Cull5547.

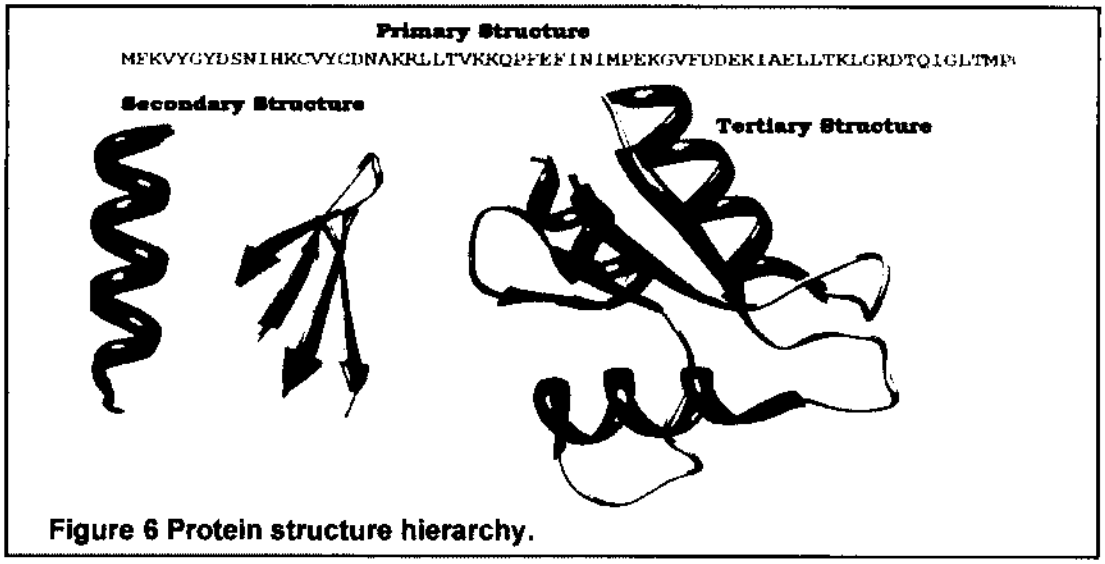


Figure 6 Protein structure hierarchy.

As an early step in characterizing protein chemistry, in 1955 British biochemist Sanger designed an experiment to identify the sequence of insulin [6]. The early effort by researchers on protein determination began in late 1950s. Two British scientists, John Kendrew and Max Perutz, published the very first high resolution protein structure [7].

Nowadays, the most common technique to determine proteins' 3D structures is X-ray Crystallography. In this method, a protein molecule is crystallized first, then a beam of X-rays hits the crystal where it will diffracts into many specific directions. A 3D picture of the density of atoms within the crystal can be produced from diffraction patterns, angles, and intensities of these diffracted beams [8]. This method determines the actual positions of the proteins' atoms and their chemical bonds. In fact, most of the proteins in PDB were determined by this technique [9, 10]. Nuclear magnetic resonance (NMR) is another popular technique used to determine the structure of proteins. With this method, a protein molecule is placed inside a strong magnetic field and irradiated with radio-frequency pulses. The energy radiated back at specific resonance frequency will then be used to calculate the positions of the atoms [11].

Applying either method is not a straightforward process. Both methods require very expensive equipment, field experts, and weeks of work, which require extra dollars for expert's labor. Moreover, X-ray Crystallography is limited by the difficulty of some proteins to form crystals and NMR can only be used to determine small proteins.

1.3.2.1 Secondary Structure

Once a protein 3D structure is determined, secondary structure elements can be identified, either by recognizing patterns of hydrogen bonds [12] or by mapping the backbone torsion angles into specific regions in the Ramachandran plot [13]. Several automated methods have been developed to determine protein secondary structures. The most commonly used method is DSSP (Dictionary of Protein Secondary Structure) [12]. Given the atomic coordinates of a protein, the DSSP program will assign a secondary structure type for each residue. An α-helix assignment (state 'H') starts when two consecutive amino acids have (i, i+4) hydrogen bonds, and ends likewise with two consecutive (i-4, i) hydrogen bonds. This definition is also used for 3_{10}-helix (state 'G') with (i, i+3) hydrogen bonds, and for π-helix (state 'I') with (i, i+5) hydrogen bonds as well. Figure 7 shows the 3 different types of protein helical shapes.

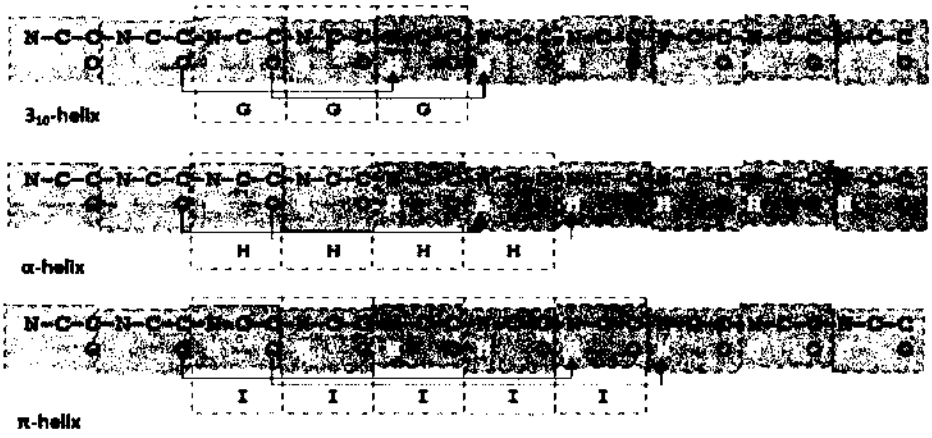


Figure 7 Hydrogen bonding patterns in helical types.

Residues in α-helices typically adopt backbone (φ, ψ) dihedral angles around (-60°, -45°), such that the ψ dihedral angle of one residue and the φ dihedral angle of the next residue sum to roughly -105°. On the other hand, residues in 3_{10}-helices typically adopt (φ, ψ) dihedral angles near (−49°, −26°), such that the ψ dihedral angle of one residue and the φ dihedral angle of the next residue sum to roughly −75°. The majority of π-helices are only 7 residues in length and do not adopt regularly repeating (φ, ψ) dihedral angles throughout the entire structure. When the first and last residue pairs are excluded, dihedral angles exist such that the ψ dihedral angle of one residue and the φ dihedral angle of the next residue sum to roughly -125°. The first and last residue pairs sum to -95° and -105°, respectively.

A minimal size helix is set to have two consecutive hydrogen bonds in the helix, leaving out single helix hydrogen bonds, which are assigned as turns (state 'T').

Beta-sheet residues (state 'E') are defined as either having two hydrogen bonds in the sheet, or being surrounded by two hydrogen bonds in the sheet. In the first case, sheets can be anti-parallel or parallel, as illustrated in Figure 8. In the second case, when sheets are being surrounded by hydrogen bonds forming isolated residues, are considered as bridges (state 'B').

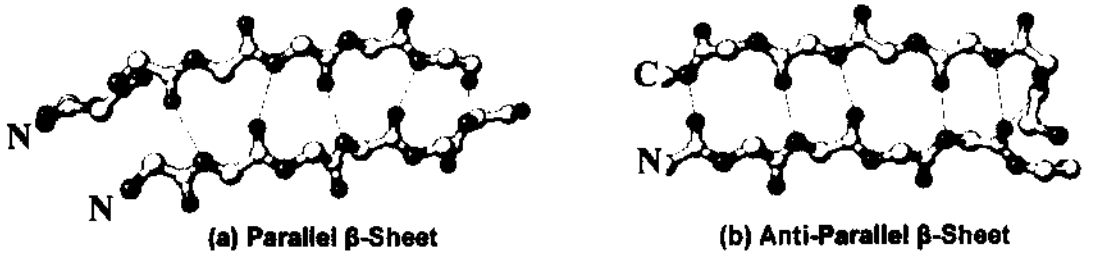


Figure 8 Hydrogen bonding patterns in Parallel & Anti parallel β-sheets.

State 'S' assignment by the DSSP program indicates a bend, which is an irregular structure, corresponding to a high twist of at least 70° in the chain. The remaining DSSP state is (space) which indicates an unassigned/other state. Table 1 list all 8-state of protein secondary structure as defined by DSSP. Figure 9 illustrates various types of secondary structures in protein 1BOO(A).

Other structural features including disulfide bonds, contacts, and solvent accessible areas can also be identified given protein 3D structure. The DSSP program can also be used to determine these structural features.

Table 1 DSSP secondary structure states.

Output	State	Description
H	α-helix	Two consecutive amino acids with (i, i+4) hydrogen bonds, and ends likewise with two consecutive (i-4, i) hydrogen bonds
G	3_{10}-helix	Same as α-helix but with (i, i+3) hydrogen bonds
I	π-helix	Same as α-helix but with (i, i+5) hydrogen bonds
T	Turn	A hydrogen bonded turn
E	β-sheet	(or β-strand) An extended strand (anti-parallel or parallel sheet)
B	β-bridge	Isolated residue - a single residue β-strand
S	Bend	A bend in the chain
	Other	Unassigned - any residue that does not belong to any of the previous 7 states.

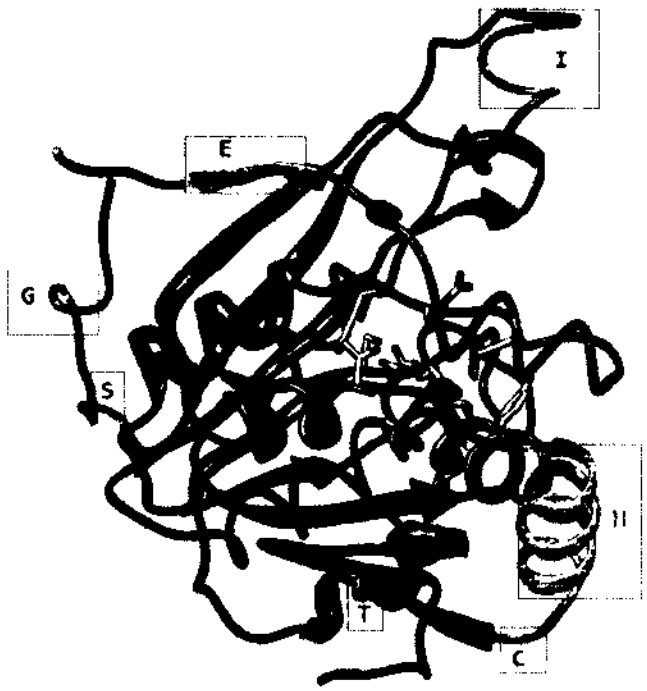


Figure 9 3D structure of Protein 1BOO Chain A.

1.3.2.2 Disulfide Bonds

Disulfide bonds (alternatively called disulfide bridges or SS-bonds) are covalent bonds formed between two sulfur atoms from nonadjacent Cysteine pairs of a protein structure. Figure 10 shows an example of a disulfide bond between two Cysteine residues in protein 153L(A). Disulfide bonds are often found in extracellular proteins, which play an important role in folding and enhancing thermodynamic and mechanical stability. Disulfide bonding patterns can also be used to discriminate structure similarity, even when low sequence similarities are present [14]. Furthermore, certain disulfide configurations provide mechanisms for sensing and responding to tensile forces, diversifying and functionalizing protein folds, minimizing aggregation, confining and coupling conformational changes, and controlling packaging and releasing for intercellular transport [15].

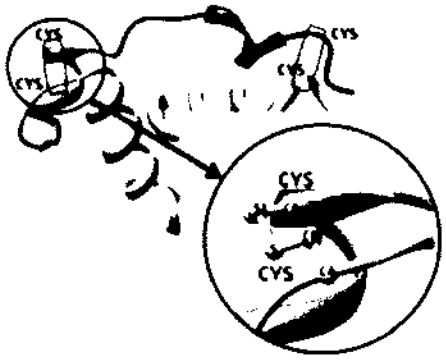


Figure 10 Disulfide bond in protein chain.

1.3.2.3 Solvent Accessibility

The solvent-accessible surface area, or accessibility, of a residue is the surface area of the residue that is exposed to solvent. The residue accessibility is a useful indicator to the residue's location, on the surface or in the core, in the protein molecule. Figure 11 shows the surface area surrounding a protein segment.

Residue solvent accessibility is usually measured by rolling a spherical water molecule over a protein surface and summing the area that can be accessed by this molecule on each residue. To allow comparisons between the accessibility of the different amino acids in proteins, typically relative values are calculated as the ratio between the absolute solvent accessibility value and that in an extended tripeptide (Ala-X-Ala) conformation [16]; referred to as the percentage of maximally accessible area. The DSSP program [12], can be used to calculate the absolute solvent accessibility values of proteins.

Figure 11 Surface area of a protein segment.

Residue solvent accessibility plays an important role in folding and enhancing proteins' thermodynamic and mechanical stability. The burial of residues at core (hydrophobic residues) is a major driving force for folding [17]. Moreover, the hydrophobic free energies are directly related to residues' solvent accessibilities, of both polar and nonpolar groups [18]. Furthermore, active sites of proteins are located on its surface. Hence, prediction of the surface residues is considered an important step in determining proteins functions [19].

1.3.3 Protein Structure Prediction

Protein in nature folds into a unique and energetically favorable 3D structure which is critical and unique to its biological function. Hence, solving the protein folding problem is the key to many applications in the fields of protein engineering and drug design.

Genome projects came out with millions of protein sequences of which only a small fraction have their 3D structure experimentally determined. (As of Tuesday April 29, 2014 there are 99,775 structures in pdb.org [9]). Researchers in the field of protein structure determination are putting large effort on bridging the huge gap between sequence and structure. The current experimental methods of protein structure determination are complicated, time-consuming, expensive, and, in some cases, unsuitable. Therefore, computational methods for structure prediction are becoming an irreplaceable option.

In the early 1950s, an American biochemist, Anfinsen, observed that the active polypeptide of a model protein could fold spontaneously into a unique 3D structure. He also observed that an enzyme unfolded under extreme environment could refold spontaneously into the same 3D structure that it folds into under natural conditions (its original or native conformation) [20]. Based on his observations, Anfinsen had developed his theory of protein folding: "The native conformation is determined by the totality of interatomic interactions and hence, by the amino acid sequence, in a given environment". This theory, also known as Anfinsen's thermodynamics hypothesis, established the foundation of *ab initio* protein structure prediction problem, i.e. predicting the native conformation of a protein from its primary sequence.

Following Anfinsen's thermodynamics hypothesis early approaches for solving the protein structure prediction were based on the thermodynamics of protein folding. Computer searching methods were applied to investigate the free energy of many local minimum energy confrontations in an attempt to find the global minimum conformation i.e., the thermodynamically most stable conformation of the protein. The main challenges in these methods were in the huge conformational space, due to the flexibility of the proteins, and the complexity of the energy functions.

The search space of the protein structure prediction problem is astronomically large. In 1969, Cyrus Levinthal stated that the protein molecule has an astronomical number of possible conformations, due to the large number of degrees of freedom in the protein primary structure. Despite this huge search space, proteins fold reliably and quickly to their native conformation. This is known as "Levinthal's Paradox" [21]. For example, a protein molecule consisting of 100 residues will have 99 peptide bonds, and therefore 198 different (φ, ψ) bond angles. If we assume that each of these angles can be in one of three stable conformations, then the protein may misfold into a maximum of 3^{198} different conformations. Therefore, if we try to sequentially enumerate all the possible conformations and evaluate each one, in order to get the one conformation with the lowest energy, assuming that each conformation is sampled at 1 nanosecond timescale, then this requires $9.4*10^{77}$ years! The "paradox" is that most proteins fold spontaneously on a millisecond or even microsecond time scale.

In order to reduce the search space of the protein structure prediction problem, one must consider developing an accurate energy function and a rapid searching algorithm. Hence, researchers dedicated large efforts in this area, which was then referred to as *ab initio* structure prediction.

During these times when *ab initio* researchers were working on their energy minimization approaches, other researchers were investigating different methodologies of protein structure prediction. Back then, an important observation was marked; "proteins that share similar sequences often share similar structures". This observation opened the gate towards a new method in solving the protein structure prediction problem, which came to be known as template-based modeling. Below is a brief description of both, template-based modeling and template-less modeling (*ab initio*).

1.3.3.1 Template-based Modeling

The template-based modeling methods are based on knowledge learned from the known protein structures deposited in protein databases [22]. These methods use sequence similarity to model the unknown target structure based on known structures that are understood to be homologous to it. When a query protein shares at least 30% sequence similarity with a protein of known structure, template-based modeling can be used to predict the structure with reasonably good accuracy [23, 24].

Template-based modeling consists of four steps:

a) Find a good template from already determined structures in the protein data bank, by means of sequence alignments.
b) Align query sequence with the template structure.
c) Build the structural framework based on alignment, by copying aligned regions.
d) Fill up the gaps on the framework [25].

Template-based modeling has been successful in many cases when homologs with high sequence similarity are available. However, when such homologs with high sequence similarity are not present in PDB or sequence alignments are incorrect, building high-quality templates becomes a difficult challenge. Moreover, the assumption that proteins with high sequence similarity share similar structures is not always true.

1.3.3.2 Template-less Modeling

Different from template-based prediction, *ab initio* (or *de novo*) predictions are methods that do not rely on templates of known structures with high sequence similarity.

An *ab initio* prediction system is generally composed of two main elements: a search algorithm and a scoring function (or energy function). The search algorithm is designed to broadly explore the protein conformation space guided by the scoring function. The scoring function is derived from physics laws or statistics, or is a combination of both, characterizing the favorability of a protein sequence adopting a certain structural conformation. The scoring function should be able to distinguish good conformations from bad confirmations. It should also properly describe the forces behind the folding process. Many scoring functions have been proposed in the literature of *ab initio* prediction. Examples include, DFIRE [26], a distance-based all atom knowledge-based function, CHARMM [27], a physics-based energy function, ICOSA, a knowledge-

based contact potential correlating residue-residue interaction distance and orientation [28], and ROSETTA[29], that includes terms combing physics-based and knowledge-based approaches.

One of the main challenges in *ab initio* protein structure prediction is the tremendously large conformation space. Reducing the conformation search space is the key to successful *ab initio* prediction. For this regard, the structural features, such as secondary structures, disulfide bonds, solvent accessibility, residue contacts, etc., become extremely useful. For example, if a protein segment is predicted to be in a certain secondary structure state with high confidence, then a limited range of the corresponding torsion angles can be used. Such restrictions on the degrees of freedom in protein segments will significantly reduce the conformation search space.

1.3.4 Protein Structural Features Prediction

The problem of structural feature prediction is formulated as a classification problem, such that each residue is predicted to be in one of several states. A coarse-grain classification of secondary structure uses 3 states (helix, sheet, and coil), whereas a fine-grain classification uses all 8 states (α-helix, π-helix, 3_{10}-helix, β-strand, β-bridge, turn, bend and others). The disulfide bonding prediction involves two stages. The first stage is the bonding state prediction, whose goal is to determine whether each Cysteine residue in a protein chain is involved in forming a disulfide bond or not. Afterward, the second stage carries out the connectivity prediction, where Cysteine pairs likely to form disulfide bonds are identified. Residue solvent accessibility is usually classified into two states, such that each residue is classified into either buried or exposed. Finer-grain classification is also possible, where levels of solvent-exposure are defined and used in the classification.

Almost all current methods for protein structural features prediction use evolutionary information revealed by multiple sequence alignment (MSA) of a family of homologues proteins. This information forms the input encodings into a machine learning algorithm, trained to recognize and discriminate the different structural features' states.

1.4 Dissertation Organization

The remainder of this dissertation is organized as following. Chapter 2 presents the related work in predicting protein structural features and also the related work in GPU-acceleration in general purpose computing and in computational biology. Chapter 3 describes our approaches for enhancing the accuracy of predicting protein structural features. We present the application of the proposed approach in predicting secondary structures, disulfide bonds, and solvent accessibility in Chapters 4, 5 and 6, respectively. In each application area, we present implementation details of the approach, computational results, and discussions. In Chapter 7, we present our method of accelerating many-body potentials on the GPU. Finally, Chapter 8 summarizes our conclusions and provides our future (post-dissertation) research directions.

CHAPTER 2

LITERATURE REVIEW

In this literature review we present a review of the history of protein structural features' predictions. Three generations of predictors are presented in section 2.1. After that, in section 2.2, we provide an analysis of the current prediction methods. In Section 2.3 we present a review of the GPU-acceleration in general purpose computing and in computational biology.

2.1 History of Protein Structural Feature Prediction

Assuming that "there should be a strong correlation between amino acid sequence and structural state," historically, protein structural feature predictions evolve through three generations.

2.1.1 1st Generation: Statistics-based methods

The early methods of prediction are based on the fact that amino acids vary in their favorability in adopting specific structural states. Using secondary structure as an example, some amino acids prefer to adopt helical conformations (Methionine and Alanine), some favor ß-strand conformations (Tryptophan, Isoleucine and Valine), and some are commonly found in turns (Proline and Glycine). Thereby, statistical analysis can be performed for each amino acid in the various types of structural states which will result in single-residue frequency. These statistical analysis approaches become the foundation of the first-generation of structural features predictors.

Representative examples of the first generation methods include Chou and Fasman's method [30] and the GOR method for secondary structure prediction and a method by Fiser et al. [31] for disulfide bonding state prediction.

Although these early statistics-based methods are very simple to implement, databases of known protein structures are of limited size and, more importantly, simple amino acid preferences are not sufficient enough to predict the protein structural information with high accuracy. As a result, the accuracy of the early prediction methods

did not exceed ~60% in 3-state secondary structure prediction and ~70% in disulfide bonding prediction.

2.1.2 2nd Generation: Machine learning-based methods

The second generation of structural features predictors extends the early methods by using the segment-based statistics (11-21 residues per segment). The basic idea is based on the likelihood of the central residue in a segment (or window) of size *N* residues to adopt a particular structural state.

In order to detect higher order correlations among amino acids, machine learning methods were used in this era of structural features prediction history. The typical methodology, based on the earliest published work by Qian & Sejnowski in 1988 [32], is to train a learning machine (for example, a feed-forward neural network) to recognize amino acid patterns adopting certain structural feature. In addition to neural networks, support vector machine, nearest neighbors, random fields, and Hidden Markov Chain are popular machine learning tools in predicting protein structural features.

Due to advancements in machine learning methods, the accuracy of structural feature prediction is significantly improved. Secondary structure prediction approaches 70% and disulfide bonding state predictions reaches 81%.

2.1.3 3rd Generation: Prediction with Effective Features

In addition to sequence information, computational biologists found that certain information, when used as features in machine learning, can effectively enhance the prediction accuracy. Therefore, modern predictors focus on generating and selecting effective features in machine learning. The most effective feature is the evolutionary information obtained by multiple sequence alignment (MSA). The fundamental idea of using evolutionary information is based on the fact that structure is more conserved than sequence. When MSA is used, one can extract more information, including conserved residues and residue substitutions during evolution. Such information is particularly useful to achieve further accuracy improvement.

The following subsections provide literature review of the current methods of predicting protein structural features.

2.1.3.1 3-state secondary structure

The first approach to use MSA profiles in secondary structure prediction was developed by Rost and Sander [33, 34], in a method called PHD, presenting an accuracy surpassing 70% for the first time when introduced. Additional input information derived from the results of MSA, including conservation weights and number of insertions and deletions, were used in later versions of PHD method [35] and resulted in ~1.6% increase of the accuracy over the original PHD method.

The PSI-PRED program [36] uses PSI-BLAST to replace BLAST to generate sequence position profiles and reaches an accuracy of ~76.5-78.3%. The more recent update of PHD method, called PROFPHD [37], also uses PSI-BLAST-derived profiles as well as ensembles of bidirectional recurrent neural network architectures and a large non-redundant training set to achieve an overall prediction accuracy of ~78%.

More recent methods with enhanced strategies and additional features lead to continuing improvements of secondary structure prediction accuracy. Porter [38], with the use of bidirectional recurrent neural networks (BRNN) and PSI-BLAST profiles, reached an accuracy of ~79%. YASPIN [39], with the use of HMM to filter the prediction of a NN, leads to similar accuracies of ~79%. SPINE [40], with the inclusion of physico-chemical properties of each amino acid combined with PSI-BLAST profiles, obtains an accuracy of ~80%. Moreover, large scale training is been conducted in order to improve the prediction performance. The recent version of PSI-PRED reported an accuracy of ~80% due to large scale training. Furthermore, new approaches have been proposed to apply a number of the recent prediction methods then combine their results and take a consensus prediction on top of them. Jpred [41] is an example of consensus predictors with an accuracy of ~81%.

All secondary-structure prediction methods are evaluated by the EVA experiment, a Web based assessment tool that evaluates prediction servers since 2006 [42]. It is also important to notice that there is certain grade of uncertainty in the assignment of secondary structures. The theoretical maximum prediction accuracy for the 3-state secondary structure is in the range between 88% and 90% [43], due to the errors resulted from secondary structure assignments based on crystal structure, and due to inconsistency of secondary structure assignments by different methods of different parameters, e.g.,

DSSP [12] and STRIDE [24]. These errors will cause ambiguity in mapping 3D atom coordinates into secondary structure classes.

2.1.3.2 8-state secondary structure

Unlike 3-state secondary structure prediction, very few methods were developed for the 8-state secondary structure prediction, to the best of our knowledge. Pollastri et al, extend their 3-state prediction method, SSpro, by developing another version for 8-state secondary structure prediction, SSpro8 [37]. The 8-state prediction accuracy reported in their work was 62-63%. A more recent prediction method developed by Xu et al [44] reports 67.9% accuracy through the use of conditional neural field (CNF) model, the method is called RaptorXss8.

2.1.3.3 Disulfide bonding state and bonding connectivity

Regarding the disulfide bonding state prediction methods, the use of evolutionary information contained in MSA leads to substantial improvements. Fariselli et al. [45] designed a jury of NNs trained by sequence profiles using MSA and resulted in 81% accuracy. Fiser and Simon [46] derived conservation scores from MSA to predict the oxidation state of Cysteine residues and obtained an accuracy of 82%. More recent methods with enhanced strategies and additional features lead to continuing improvements of bonding state prediction accuracy. Mucchielli-Giorgi et al. [47] investigated the contribution of the overall amino acid composition of the protein and managed to increase the accuracy to 84%. Ceroni et al. [48] proposed a method using spectrum kernel in SVMs, which yielded 85% prediction accuracy. Martelli et al. [49] combined a hybrid HMM and a NN in their prediction system and reached 84% and 88% accuracy measured on protein basis and Cysteine basis, respectively. Song et al. [50] incorporated dipeptide composition as features in prediction and gained similar accuracy.

The connectivity prediction started with the early method proposed by Fariselli and Casadio [51] based on graph matching, such that edges are weighted by residue contact potentials. Although this method was 17 times higher than a random predictor, still it is not comparable with the current connectivity predictors that incorporate evolutionary information contained in MSA, in advanced machine learning technologies. Ceroni et al. [52] encoded MSA data into Recursive Neural Networks in their

DISULFIND server with 54.5% pattern precision and 60.2% bonded pair accuracy. Ferre and Clote [53] took advantage of secondary structure encoding in their DiANNA server and reached 86% accuracy. Cheng et al. [54] performed large-scale prediction of disulfide connectivity using kernel methods, two-dimensional recursive neural networks, and weighted graph matching and obtained accuracy of 51% pattern precision. Vincent et al. [55] took advantage of decomposition kernels for classifying chains instead of individual residues and achieved prediction accuracy comparable to the other prediction methods.

2.1.3.4 Solvent accessibility

A number of methods have been developed using different protein datasets and different computational methods, including neural networks [56-61], support vector machines [62, 63], nearest neighbor [64, 65], information theory [66], and Bayesian statistics [67]. In most of these methods, the prediction is performed in a discrete fashion, where predictors discriminate among a number of predefined levels or states of residues' exposure with predefined thresholds.

Predicting solvent accessibility using evolutionary information, revealed by multiple sequence alignments, led to a significant accuracy increase. Rost et al [68], Cuff et al [69], and Thompson et al [67] reported a two-state prediction accuracy of ~75% with 0.25 threshold. More recent prediction methods benefit from PSI-BLAST derived profiles to reach higher accuracies of ~78% in two-state prediction with 0.25 threshold, and an accuracy of ~64% in three-state prediction with 0.9 and 0.36 thresholds [59, 63-65].

Most of the current methods nowadays provide real value prediction, in addition to discrete-fashion prediction (in 2-state, 3-state, or more). The Pearson correlation coefficient (between the predicted and true values) reported in real value predictors is ~0.65 [64, 70].

2.2 Analysis of Protein Structural Feature Prediction

Large improvements have been obtained since the first generation structural features predictors. However, little significant progress has been reported in the past few years. Obtaining improvements of even a fraction of a percent has become very difficult. One of the reasons is the closer to the theoretical upper bound, the harder to achieve a

higher accuracy. More importantly, lack of further effective features limits the performance of the learning machines.

Probably the most effective features, when predicting the structural state of a residue, are the structural states of the neighboring residues. For example, if the neighboring residues are exposed to solvent, the middle residue is likely exposed to solvent as well. If the neighboring residues are helices (sheets), the middle residue also has a high probability to adopt helix (sheet). Unfortunately, the structural features of the neighboring residues cannot be directly applied to the learning machines, since they are unknown beforehand.

The main contribution of this work is the generation of context-based scores to describe the favorability of the neighboring residues in adopting certain structural states and then encode these scores to train machine learning methods, with the expectation of improving prediction accuracy. In Chapter 3, we will explain our approach of extracting, selecting, and then encoding these features in a neural network algorithm to improve the prediction accuracy of protein structural features.

2.3 GPU-acceleration

Today's Graphical Processing Units (GPUs) greatly outpace CPUs in arithmetic throughput and memory bandwidth, forming a dramatic shift in the field of high performance computing [71]. With the massively parallel computing mechanisms, GPUs are able to deliver performance speedups tens of times more than the CPU (and sometimes hundreds of times), to solve problems in few minutes instead of hours or days. Hence, GPUs became the ideal processor to accelerate a wide variety of data parallel applications.

2.3.1 GPU applications in general purpose computing

Efforts to utilize the GPU for non-graphical applications have been underway since 2003. With the introduction of the CUDA environment from NVIDIA in 2007, a wide variety of applications have emerged [71-73] taking advantage of the GPU capabilities in accelerating the operations of these applications. Owens et al. [72] described the techniques used in mapping general-purpose computation to GPUs, and

they surveyed and categorized the latest developments in general-purpose application development on GPUs. Another survey was conducted by Stone et al. [74] on the development of molecular modeling algorithms that uses GPU computing. Zhu has designed a number of GPU-accelerated algorithms with CUDA for global optimization [75, 76]. Zhou and Tan [77] developed a parallel approach to run standard particle swarm optimization on GPU with speedup of around 11. You [78] designed a parallel Ant Colony Optimization (ACO) system for Traveling Salesman Problem on GPUs. Bakhtiari et al. applied GPU to Monte Carlo optimization in dose calculation in radiation therapy [79].

2.3.2 GPU applications in energy evaluations

It is common that the computation time of a protein structure modeling program ranges from several hours to several days or even longer. Many protein structure modeling applications can greatly benefit from a significant reduction in computation time by enabling one to sample broader conformation space to discover the appropriate structures, execute more simulation steps to obtain models with better accuracy, or carry out simulation on much larger proteins.

The most costly operations in many protein modeling applications are energy evaluations of protein molecules. Protein energy evaluation involves calculating all the interactions among protein atoms, which is typically an *N*-body problem. Reducing the energy evaluation time is the key to accelerate many protein structure modeling applications.

Being able to simultaneously calculate interaction forces among *N* particles, GPU has been employed to accelerate the *N*-body simulation in a variety of applications. For example, Nyland, Harris, and Prins [80] developed a fast *N*-body astrophysical simulation on NVIDIA GeForce 8800 GTX GPU. They reported on the performance of the all-pairs *N*-body kernel for the simulation, demonstrating several optimizations to improve the performance.

Stock and Gharakhani [81] introduced an efficient multi-pole-accelerated tree code method for turbulent flows computations. Anderson, Lorenz, and Travesset [82] developed a general purpose molecular dynamics program that runs entirely on the GPU, showing that GPU provides an inexpensive alternative to a fast 30 processors distributed

memory cluster. Mark et el. [83] described a complete implementation of all-atom protein molecular dynamics running entirely on GPU, including all standard force field terms, integration, constraints, and implicit solvent. They provided two implementations on ATI and NVIDIA GPUs. Belleman, Bedorf and Zwart [84] took the advantage of the parallelism in the GPUs in speeding up the force evaluations in a gravitational direct N-body simulation. They concluded that modern GPUs offer an attractive alternative to special purpose hardware designed for simulations.

CHAPTER 3

CONTEXT-BASED MODEL

In this chapter we describe our proposed approach. An overview of the materials and the methods are presented in section 3.1, implementation details are described in section 3.2, and methods of evaluations are presented in section 3.3.

3.1 Model Overview

Figure 12 is a flow chart representing our developed approaches in predicting protein structural features. The dashed lines in the figure represent optional steps. First of all, in this work, we derive a statistical model to obtain context-based scores. Then, together with other features such as evolutionary information, we incorporate the context-based scores in machine learning approaches to predict secondary structures, disulfide bonding state and bonding connectivity, and solvent accessibility.

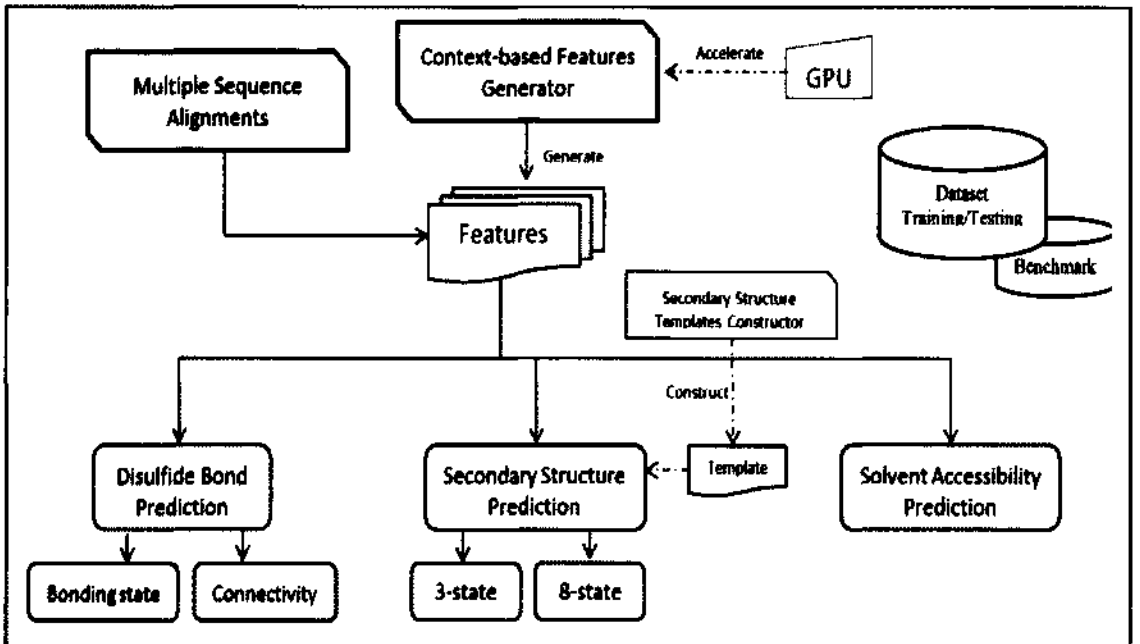


Figure 12 The proposed methodologies in predicting protein structural features.

Moreover, we take advantage of the emerging parallel computing architectures in GPU to accelerate the derivation of context-based features. Finally, we make the predictors available to the public via web services and software packages.

It is well known that there exist general short range regularities in the primary structure of proteins [85]. Early studies have shown that the types and conformations of neighboring residues play a significant role in the structural conformation a residue adopts. In fact, taking advantage of the local information embedded in the context of a residue is the foundation for most protein structural feature prediction methods.

3.1.1 Datasets

- *Training Data Sets*

A number of protein datasets are taken from the protein sequence culling server, PISCES [5]: Cull16633, Cull7987 and Cull5547, referred to as CullPDB lists. Table 2 lists the characteristics of each dataset. These lists are used in our experiments for the proposed approach in the different structural features prediction methods. Cull16633, the largest among the datasets, is used for generating context-based statistics. The other two datasets are used for NN training in structural prediction.

The structural features of each residue in the CullPDB lists are determined by the DSSP program [12]. These features include 8 states of secondary structure, solvent accessible surface areas, and disulfide bonds among Cysteine residues only. Figure 13 shows the distribution of various structural features in Cull16633. Similar to most existing structural feature prediction methods, we apply a general elimination strategy to CullPDB list. We eliminate very short chains with less than 40 residues, since the PSI-BLAST program is usually unable to generate profiles for very short sequences. We also remove very large chains whose lengths are greater than 1000 residues.

Table 2 Characteristics of CullPDB lists.

	Cull16633	Cull7987	Cull5547
Number of chains	16,633	7,987	5,547
Pair-wise sequence identity	50%	25%	25%
Resolution (Å)	3.0	3.0	2.0

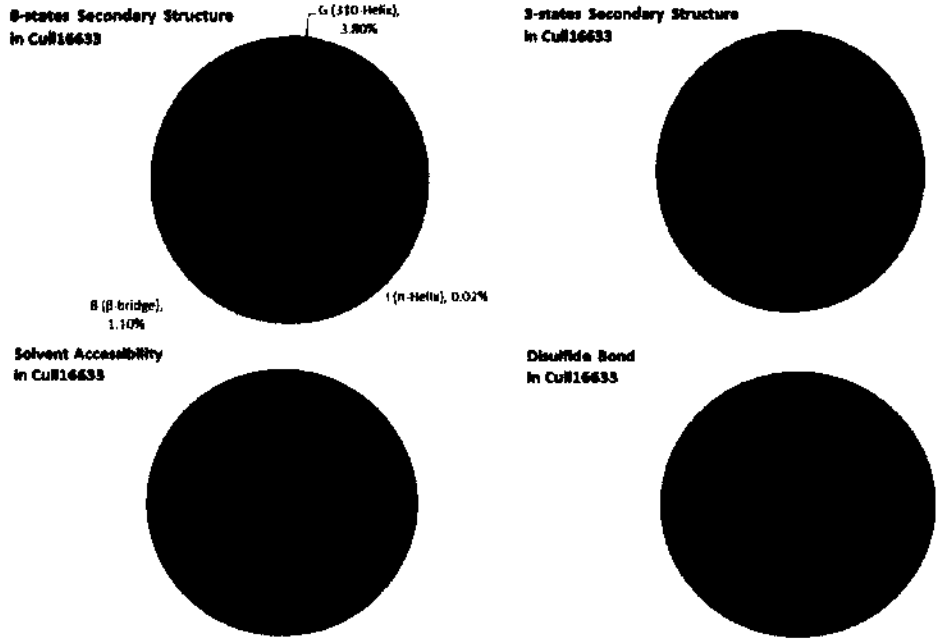


Figure 13 Distribution of the structural states in the Cull16633.

- *Benchmarks*

For the purpose of comparing our methods with the current methods of structural feature predictions, several protein benchmark sets are used. These sets include CB513 [69], Manesh512 [2], Carugo338 [86] and the CASP9 targets [87].

3.1.2 Multiple Sequence Alignment

Our proposed approaches critically rely on the evolutionary information in multiple sequence alignment (MSA). Similar to many modern protein structural feature predictors, this evolutionary information, representing the divergence of a protein chain in its structural family of proteins and contained in matrix format referred to as PSSM (Position Specific Scoring Matrix), is obtained by running PSI-BLAST with 3 iterations of searching against non-redundant database of protein sequences (NR).

3.2 Model Implementation

Figure 14 depicts the flowchart of the proposed model. PSSM data of the CullPDB sets are generated by running PSI-BLAST against the NR (non-redundant)

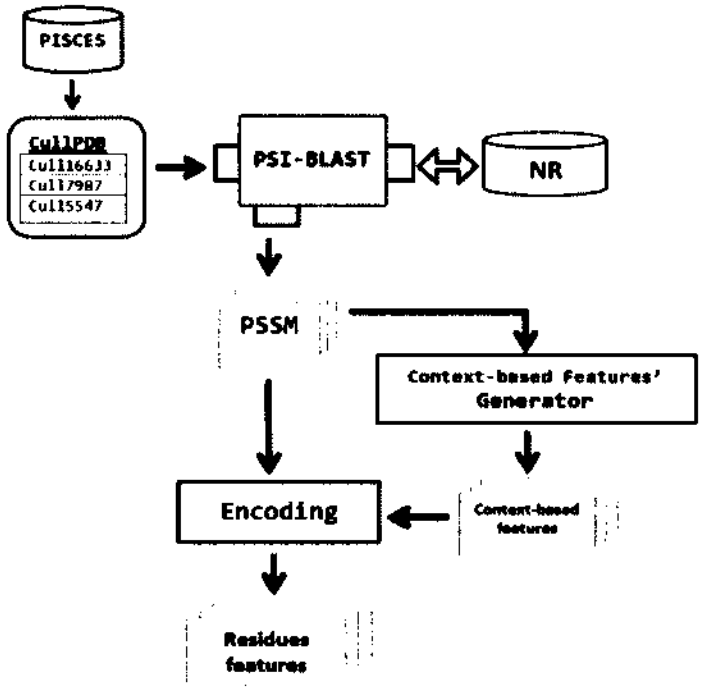


Figure 14 Context-based Model.

database. After that, context-based features are generated based on the PSSM data of Cull16633 and then combined with the PSSM data of Cull7987 or Cull5547 to describe each residue. The resulting residues' features are then encoded in neural networks for training and testing. The context-based features are represented as pseudo-potential scores to estimate the favorability of residues in adopting specific structural states within their amino acid environment. These potentials are calculated based on the context-based statistics, which are derived from the protein datasets.

3.2.1 Derivation of Context-based Scores

We extract statistics of residues at different relative positions in Cull16663 protein sequences. These statistics represent estimations of the probabilities of residues adopting specific structural states when none, one, or more of their neighbors in context are taken into consideration. Fortunately, the recent increasing number of determined structures in protein data banks will make derivation of high-order-inter-residue correlation statistics feasible.

Furthermore, to obtain more precise neighboring correlation statistics of residues' structural states, we consider the divergence of a protein sequence in its structural family by using the PSSM data specifying the frequency of each amino acid type in a protein MSA. The derived statistics of correlations between each residue and its nearby neighbors are then used to calculate context-dependent pseudo-potential scores using Sippl's potentials of mean force method based on the inverse-Boltzmann theorem [88]. These scores are encoded in NN training for the different structural features predictors.

3.2.2 Machine Learning Algorithm

We consider neural network as our machine learning model. Implementation details will be presented with each predictor in chapters 4, 5 and 6.

3.3 Methods of Evaluation

We use Q*i* and SOV*i* scores, where *i* is the number of states, to measure the prediction quality of secondary structure and solvent accessibility. The definition of Qi is, $Q_i = 100 * {}^{P_i}/_{N_i}$, where P_i is the total number of correctly predicted residues in state i and N_i is the total number of residues in this state. For example, in 3-state secondary structure prediction, Q3 is the percentage of residues predicted correctly in one of the three states: helix, strand, and coil.

For state i, SOVi (Segment overlap [89]) is defined as,

$$SOV_i = 100 * \frac{1}{N_i} * \sum_{S_i} \left[\frac{\text{minOV}(s1, s2) + \delta(s1, s2)}{\text{maxOV}(s1, s2)} * \text{len}(s1) \right]$$

,where (s1, s2) is a pair of overlapping segments, S_i is the set of all overlapping pairs of segments in state i and N_i is the summation of the s1 lengths in Si (where there is an overlap with s2) plus the summation of the lengths of s1 in S_i' (where there is no overlap between segment s2 and s1). $minOV(s1, s2)$ is the length of the actual overlap of (s1, s2) in state i, $maxOV(s1, s2)$ is the total extent for which either segment (s1 or s2) has residue in state i and $\delta(s1, s2)$ is defined as: $\delta(s1, s2) = \min\{maxOV(s1, s2) - minOV(s1, s2), minOV(s1, s2), int(len(s1) / 2), int(len(s2) / 2)\}$, where $len(s1)$ and $len(s2)$ are the lengths of s1 and s2 respectively.

In order to measure the quality of disulfide bond prediction, we use sensitivity (Sn), specificity (Sp), and Matthew's correlation coefficient (Mcc) [90]. The definitions of each is given by

$$S_n = {}^{TP}/_{(TP + FN)}$$

$$S_p = {}^{TN}/_{(TN + FP)}$$

$$Mcc = \frac{(TP * TN - FN * FP)}{\sqrt{(TP + FN) * (TN + FP) * (TP + FP) * (TN + FN)}},$$

where TP, TN, FP, and FN are the number of true positives, the number of true negatives, the number of false positives, and the number of false negatives, respectively. We also use residue-level accuracy (Q_c) and protein-level accuracy (Q_p) to measure the prediction accuracy. The residue-level accuracy is defined as, $Q_c = {}^{P_c}/_{N_c}$, where P_c is the total number of correctly predicted residues (in a specific structural state), and N_c is the total number of residues. The protein-level accuracy is defined as, $Q_p = {}^{P_p}/_{N_p}$, where P_p is the total number of proteins where the structural states of all of its residues are correctly predicted and N_p is the total number of proteins in the data set.

Furthermore, in order to have a reliable estimate of the prediction accuracy, *N*-fold cross validation is conducted. We randomly divide the protein chains in the training set into *N* subsets with approximately the same number of chains. At each fold, *N*-2 subsets are used for training, one for testing, and one for validation. To ensure complete separation of the training set and testing set for each fold, we generate a set of scores only based on the sequences in the *N*-2 training subsets and then encode it in training. Hence, totally *N* sets of context-based scores are generated for *N*-fold cross validation. The overall prediction accuracy is calculated as the average accuracy of the N fold predictions.

Regarding the GPU acceleration, a simple performance metric is used to measure the speedup, defined by dividing the CPU time (tCPU) by the GPU time (tGPU). $speedup$ = tCPU / tGPU.

CHAPTER 4

SECONDARY STRUCTURE PREDICTION

Using the context-based model, we extract context-based statistical scores to measure favorability of a residue adopting secondary structure in its amino acid environment. The fundamental idea is based on the fact that the formation of secondary structure exhibit strong local dependency, particularly, residues in a protein sequence are strongly correlated in different sequence positions in coils, β-sheets, 3-10 helices, α-helices, and π-helices. The context-based statistics indicate the favorability of a residue adopting a secondary structure conformation in presence of its neighbors in sequence. We derive statistics for singlets, doublets, and triplets in a sequence window from the CullPDB dataset. Then scores measuring the pseudo-potentials of a residue adopting a certain secondary structure are calculated using the potentials of mean force approach. These scores are incorporated as sequence-structure features together with the PSSM data to train the secondary structure prediction neural networks. We apply our approach to predict secondary structures in both 3-state and 8-state. Our server implementing this method is named SCORPION (**Se**c**CO**nda**R**y structure **P**redict**ION**) [91]. C3-SCORPION for 3-state prediction is available at: http://hpcr.cs.odu.edu/c3scorpion and C8-SCORPION for 8-state prediction is available at: http://hpcr.cs.odu.edu/c8scorpion. To take advantage of available homolog information, we also develop a template-based approach to construct templates for secondary structure predictors [92].

We test our methods on benchmarks of CB513 [69], Manesh215 [16], and Carugo338 [86] as well as the CASP9 targets [87]. We compare our results with a set of popular secondary structure prediction methods including Porter (*ab initio*) [38], Psipred [36], PROFphd [35], Netsurfp [60], and Jpred [41] for 3-state predictions, and with RaptorXss8 [44] for 8-state predictions. Prediction accuracy of our methods is further analyzed in this work.

4.1 C3-SCORPION

The early studies in protein secondary structure show that the types of nearby neighboring residues play a predominating role to secondary structure conformation a residue adopts. In particular, the formation of interactions within coils beyond nearest neighbors appears not to contribute statistically significantly in determining coil structure [93]. The hydrogen bonds between residues at positions i and i+3, i and i+4, and i and i+5 lead to the formation of 3-10 helices, α-helices, and π-helices, respectively. Moreover, residues in contacting parallel or anti-parallel β-sheets are connected by hydrogen bonds in alternative positions.

Figure 15 shows the probability of Alanine as the middle residue of a triplet with neighboring residues at 1~5 positions away when adopting α-helix as secondary structure. As expected, the nearest neighbors have the strongest influence to the middle Alanine and the further the neighbors are away, the weaker the influence. However, even residues five positions away have non-negligible influence on the secondary structure of the middle Alanine.

Hence, capturing these correlations and then incorporate them as features into the learning process of a secondary structure predictor can enhance the prediction accuracy.

4.1.1 Method Implementation

- *Context-based Statistics*

We extract statistics of singlets (R_i), doublets (R_iR_{i+k}), and triplets ($R_iR_{i+k_1}R_{i+k_2}$) residues at different relative positions from protein sequences in Cull Database. These statistics represent the estimated probabilities of certain residues adopting a specific structural state when none, one, or two of their neighbors in context are taken into consideration, respectively [94].

The observed probabilities of the i^{th} residue R_i in a singlet (R_i), doublet (R_iR_{i+k}), and triplet ($R_iR_{i+k_1}R_{i+k_2}$) adopting a specific structural state C_i will be respectively estimated by

$$P_{obs}(C_i|R_i) = \frac{N_{obs}(C_i, R_i)}{N_{obs}(R_i)},$$

$$P_{obs}(C_i|R_iR_{i+k}) = \frac{N_{obs}(C_i, R_iR_{i+k})}{N_{obs}(R_iR_{i+k})}, \text{and}$$

$$P_{obs}(C_i|R_iR_{i+k_1}R_{i+k_2}) = \frac{N_{obs}(C_i, R_iR_{i+k_1}R_{i+k_2})}{N_{obs}(R_iR_{i+k_1}R_{i+k_2})}.$$

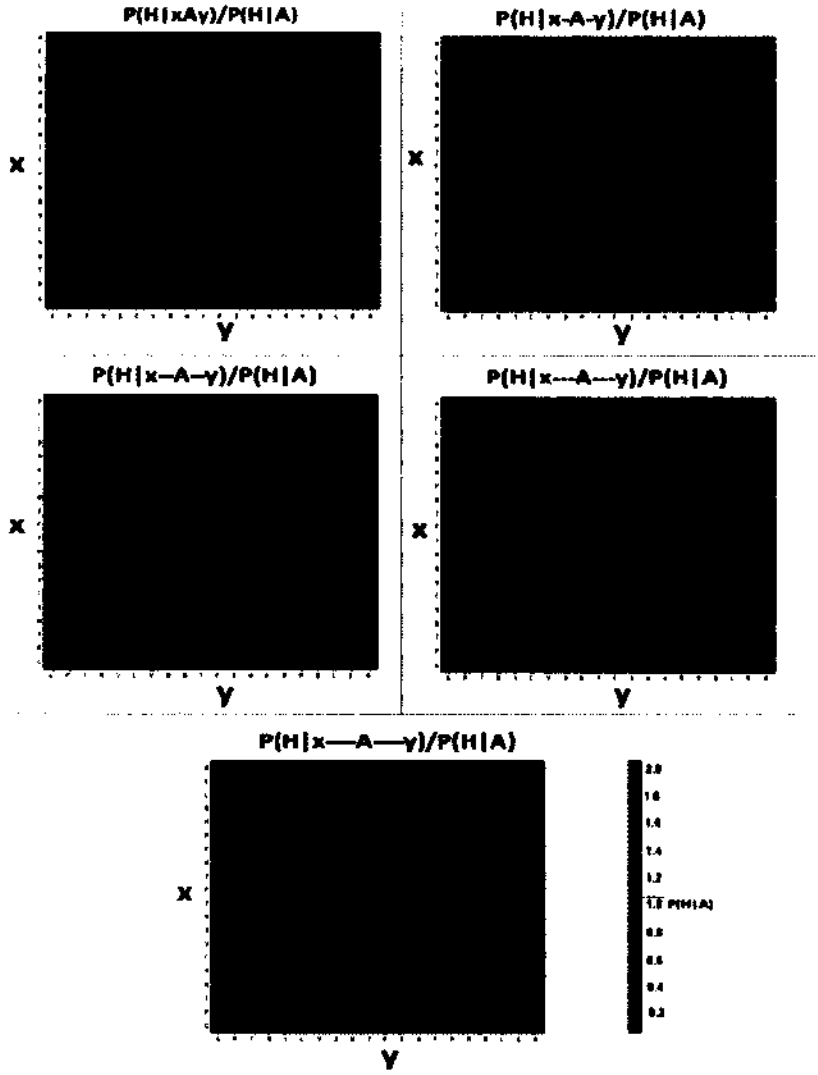


Figure 15 The probability of Alanine as the middle residue of a triplet with neighboring residues at 1~5 positions away when adopting α-helix as secondary structure. x, y, and - represent the left neighbor, right neighbor, and gap, respectively. The neighboring residues are ordered by their favorability of forming α-helix. The nearest neighbors have the strongest influences; however, neighbors 5 positions away still have certain non-negligible influences on the middle Alanine.

Here $N_{obs}(C_i, R_i)$, $N_{obs}(C_i, R_i R_{i+k})$, and $N_{obs}(C_i, R_i R_{i+k_1} R_{i+k_2})$ are the weighted observed number of singlet (R_i), doublet ($R_i R_{i+k}$), and triplet ($R_i R_{i+k_1} R_{i+k_2}$) with R_i adopting conformation C_i in the protein structure database. $N_{obs}(R_i)$, $N_{obs}(R_i R_{i+k})$, and $N_{obs}(R_i R_{i+k_1} R_{i+k_2})$ are the weighted observed number of singlets, doublets, and triplets. The observed numbers will be calculated as

$$N_{obs}(R_i) = \sum_{Protein} \sum_{j} PSSM_j(R_i),$$

$$N_{obs}(R_i R_{i+k}) = \sum_{Protein} \sum_{j} PSSM_j(R_i) * PSSM_j(R_{i+k}),$$

$$N_{obs}(R_i R_{i+k_1} R_{i+k_2}) = \sum_{Protein} \sum_{j} PSSM_j(R_i) * PSSM_j(R_{i+k_1}) * PSSM_j(R_{i+k_2}),$$

$$N_{obs}(C_i, R_i) = \sum_{Protein} \sum_{j}^{C_j=C_i} PSSM_j(R_i),$$

$$N_{obs}(C_i, R_i R_{i+k}) = \sum_{Protein} \sum_{j}^{C_j=C_i} PSSM_j(R_i) * PSSM_j(R_{i+k}), \text{and}$$

$$N_{obs}(C_i, R_i R_{i+k_1} R_{i+k_2}) = \sum_{Protein} \sum_{j}^{C_j=C_i} PSSM_j(R_i) * PSSM_j(R_{i+k_1}) * PSSM_j(R_{i+k_2}),$$

where $PSSM_j(R_i)$ is the PSSM frequency for residue type R_i at the j^{th} position of a protein sequence.

- *Context-based potentials*

The context-dependent pseudo-potentials are generated using the derived statistics of correlations between each residue and its nearby neighbors based on Sippl's potentials of mean force method. According to the inverse-Boltzmann theorem [88], we calculate the mean-force potential $U_{singlet}(R_i, C_i)$ for a singlet residue R_i adopting structural state C_i,

$$U_{singlet}(C_i, R_i) = -RTln\frac{P_{obs}(C_i|R_i)}{P_{ref}(C_i|R_i)}.$$

Here R is the gas constant, T is the temperature, and $P_{ref}(C_i|R_i)$ is the referenced probability. In our method, we employ the conditional probability approach described in [95] to estimate the referenced probability by

$$P_{ref}(C_i|R_i) = \sum_{j}^{C_j=C_i} N_{obs}(C_j, R_j) \Big/ \sum_{j} N_{obs}(R_j).$$

Similarly, the mean-force potentials $U_{doublet}(C_i, R_iR_{i+k})$ and $U_{triplet}(C_i, R_iR_{i+k_1}R_{i+k_2})$ for residue adopting structural state are calculated as

$$U_{doublet}(C_i, R_iR_{i+k}) = -RTln\frac{P_{obs}(C_i|R_iR_{i+k})P_{ref}(C_i|R_i)}{P_{ref}(C_i|R_iR_{i+k})P_{obs}(C_i|R_i)}$$

and

$$U_{triplet}(C_i, R_iR_{i+k_1}R_{i+k_2}) = -RTln\frac{P_{obs}(C_i|R_iR_{i+k_1}R_{i+k_2})P_{ref}(C_i|R_iR_{i+k_2})P_{ref}(C_i|R_iR_{i+k_1})P_{obs}(C_i|R_i)}{P_{ref}(C_i|R_iR_{i+k_1}R_{i+k_2})P_{obs}(C_i|R_iR_{i+k_2})P_{obs}(C_i|R_iR_{i+k_1})P_{ref}(C_i|R_i)},$$

with the corresponding referenced probability,

$$P_{ref}(C_i|R_iR_{i+k}) = \sum_{j}^{\substack{C_j=C_i \\ R_{j+k}=R_{i+k}}} N_{obs}(C_j, R_jR_{j+k}) \Big/ \sum_{j} N_{obs}(R_jR_{j+k}),$$

and

$$P_{ref}(C_i|R_iR_{i+k_1}R_{i+k_2}) = \sum_{j}^{\substack{C_j=C_i \\ R_{j+k_1}=R_{i+k_1} \\ R_{j+k_2}=R_{i+k_2}}} N_{obs}(C_j, R_jR_{j+k_1}R_{j+k_2}) \Big/ \sum_{j} N_{obs}(R_jR_{j+k_1}R_{j+k_2}).$$

Then, the context-dependent pseudo-potential for R_i is the summation of singlet, doublet, and triplet potentials within window size

$$U(C_i, R_i) = U_{singlet}(C_i, R_i) + \sum_{k} U_{doublet}(C_i, R_iR_{i+k}) + \sum_{k_1,k_2} U_{triplet}(C_i, R_iR_{i+k_1}R_{i+k_2}).$$

These pseudo-potential scores are then incorporated as sequence-structure features together with the PSSM data to train the secondary structure prediction process.

- *Neural Network Model*

We incorporate three phases of feed-forward neural network training in C3-SCORPION. The first and second phases are sequence-to-structure and structure-to-structure training, respectively, while the third phase is used to refine the prediction results. Figure 16 illustrates the neural network encoding and architecture for three

phases of training. In the sequence-to-structure training, a sliding window of 15 residues is selected, where each neural network is trained to predict the class of that residue in the middle of the window. Each residue is represented by 20 PSSM values and 1 extra value to indicate C- or N-terminals overlap. When the context-based scores are incorporated, 3 additional encoding values for each residue are needed.

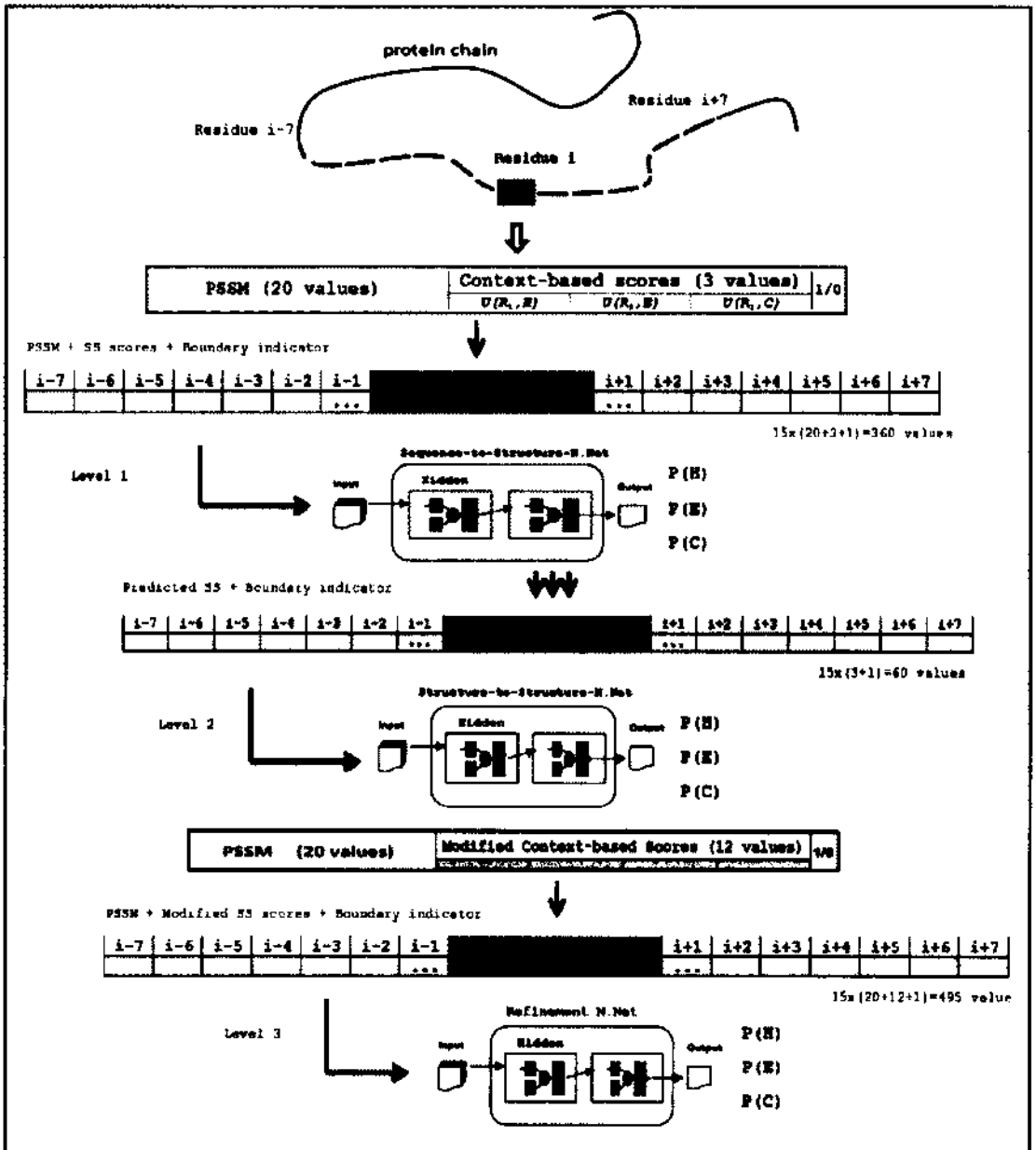


Figure 16 Three phases of prediction (architecture and encoding) for C3-SCORPION.

Overall, 360 input values are used to encode each residue in 3-state prediction. After sequence-to-structure training, the next phase is to carry out a structure-to-structure training to eliminate unrealistic secondary structure predictions such as ...CCCHCCC.... The last phase employs a similar manner as the first one, but setting some context-based scores to "absolute favorable" if the results from structure-to-structure prediction indicates that the probability of a residue adopting a certain secondary structure is higher than 90%.

4.1.2 Results of C3-SCORPION

Table 3 compares the 7-fold cross validation Q3 and SOV3 accuracies of neural networks for 3-state prediction with context-based score encoding (PSSM + Context-based Score) and without context-based score encoding (PSSM Only). Both neural network trainings go through the same training and cross-validation procedure. When the context-based scores are incorporated, both Q3 and SOV3 accuracy enhancements are observed in all three secondary structure classes. The overall cross-validated Q3 accuracy is 82.74%, which is higher than the reported accuracies (~80%) in the popular secondary structure prediction servers [35, 36, 38, 41, 60]. It is important to notice that the most significant accuracy improvement (4.02%) is found in β-sheets. This is particularly encouraging because β-sheets are typically harder to predict than helices due to global interactions. 2.55% and 1.47% accuracy improvement are also observed in helices and coils, respectively. Due to the fact that residues in sheets and helices yield stronger correlation to the neighboring residues than those in coils, the context-based scores are more effective on sheets and helices predictions. The overall 7-fold cross-validated SOV3 accuracy reaches 86.25%, which is 2.39% higher than that of using PSSM encoding only.

Table 4 shows the Q3 accuracy and the composition frequency of each amino acid type. The prediction accuracy for Cysteine is the lowest, mainly due to its lowest composition frequency in protein sequences. Moreover, a Cysteine residue may form disulfide bond with another Cysteine residue, which complicates the prediction of their secondary structures.

Table 3 7-fold cross validation Q3 and SOV3 accuracies for 3-state prediction in SCORPION.

	PSSM Only	PSSM + Context-based Score
Q_H	84.74%	87.29%
Q_E	72.72%	76.74%
Q_C	80.53%	82.00%
Q_3	80.31%	82.74%
SOV_H	87.85%	90.34%
SOV_E	81.87%	84.13%
SOV_C	81.19%	83.31%
SOV_3	83.86%	86.25%

Table 4 Q3 accuracy for each amino acid type in SCORPION.

AA	A	R	N	D	C	Q	E	G	H	I
Comp(%)	8.11	5.16	4.33	5.92	1.26	3.88	6.88	6.96	2.33	5.85
Q3(%)	83.59	81.96	81.96	83.67	76.79	83.27	83.03	83.56	81.20	84.29
AA	L	K	M	F	P	S	T	W	Y	V
Comp(%)	9.64	5.82	1.62	4.19	4.52	6.03	5.45	1.42	3.62	7.00
Q3(%)	83.74	82.07	82.69	80.65	84.31	80.72	80.92	80.02	79.96	83.70

Table 5 and Table 6 compare the Q3 and SOV3 accuracies between our method and the popularly used secondary structure prediction servers, including Porter (*ab initio*), PsiPred, ProfPhD, Netsurfp, and Jpred on benchmarks of CB513, CASP9, Manesh215, and Carugo338. To enforce fairness comparison, we generate context-based scores by removing all sequences with 25% or higher sequence identity to the sequences in benchmark from Cull16633 and all homologs with higher than 25% sequence identity to the chains presented in these benchmarks are excluded from Cull7987 when training neural networks. It is interesting to notice that our prediction method has significant higher accuracy in both α-helices and β-sheets than the other servers, with more than 5% improvements in most cases. However, as a tradeoff, the accuracy of coils is around 5% less compared to PsiPred, around 3.6% less compared to Netsurfp and around 2% less compared to JPred. After all, compared to PsiPred with the highest Q3 accuracy, our method's improvement is from 0.5% to 2% on these benchmarks. Although 0.5% to 2% accuracy improvement over PsiPred does not seem very attractive, it is important to

notice that more than 5% improvement in SOV3 accuracy compared to PsiPred. Moreover, the SOV3 accuracy of our server is 4.25% higher than Porter (*ab initio*) and is more than 6% higher compared to Netsurfp, JPred, or PsiPred. This is because the context-based scores incorporating secondary structure information of neighboring residues enhance the coverage of the secondary structure segments.

Table 5 Comparison of Q3 accuracy between SCORPION and other popularly used secondary structure prediction methods including Porter (ab initio), PsiPred, ProfPHD, NetSurfp, and JPred on benchmarks of CB513, CASP9, Manesh215, and Carugo338.

		CB513	CASP9	Manesh215	Carugo338
Porter (ab initio)	Q3	77.53	78.65	77.99	77.5
	QH	81.30	85.45	81.72	80.67
	QE	66.18	67.4	66.73	66.21
	QC	80.49	78.75	80.57	81.1
Psipred	Q3	80.19	81.35	80.67	80.06
	QH	79.28	83.32	78.29	77.09
	QE	68.49	69.68	69.43	67.96
	QC	87.11	85.84	88.63	88.87
Profphd	Q3	76.52	76.91	76.77	76.47
	QH	80.06	84.41	80.02	78.76
	QE	69.18	65.53	68.78	68.82
	QC	77.54	76.48	78.06	78.8
Netsurfp	Q3	77.88	79.35	78.7	78.24
	QH	77.21	82.46	77.39	76.2
	QE	64.36	64.94	66.17	64.65
	QC	85.56	84.27	86.39	87.1
JPred	Q3	78.72	79.24	79.32	78.67
	QH	78.02	79.29	77.72	76.34
	QE	69.04	74.05	71.48	69.37
	QC	84.39	82.09	84.81	85.49
C3-SCORPION	Q3	80.69	83.02	82.66	81.96
	QH	85.27	88.38	86.22	85.51
	QE	72.69	77.66	75.97	74.07
	QC	81.15	81.44	82.95	83.43

Table 6 Comparison of SOV3 accuracy between SCORPION and other popularly used secondary structure prediction servers including Porter (ab initio), PsiPred, ProfPHD, NetSurfp, and JPred on benchmarks of CB513, CASP9, Manesh215, and Carugo338.

		CB513	CASP9	Manesh215	Carugo338
Porter (ab initio)	SOV3	80.21	82.41	80.90	80.03
	SOVH	84.64	88.36	85.07	84.09
	SOVE	76.06	77.24	76.70	76.68
	SOVC	78.76	79.87	79.32	78.61
Psipred	SOV3	78.91	81.24	79.55	77.63
	SOVH	83.61	87.21	84.00	82.40
	SOVE	77.35	78.62	78.56	77.15
	SOVC	75.80	77.19	75.96	74.03
Profphd	SOV3	78.96	79.87	79.62	78.28
	SOVH	83.79	86.42	83.59	82.55
	SOVE	76.19	74.08	76.52	76.21
	SOVC	76.44	77.09	77.64	75.99
Netsurfp	SOV3	77.66	79.74	78.92	77.18
	SOVH	82.21	86.13	82.86	81.62
	SOVE	75.06	75.25	77.02	75.40
	SOVC	75.26	76.38	76.31	74.52
JPred	SOV3	78.82	81.70	79.63	77.98
	SOVH	82.85	83.39	82.88	81.61
	SOVE	77.56	82.52	79.62	78.51
	SOVC	76.11	79.74	76.63	74.71
C3-SCORPION	SOV3	83.98	86.38	85.72	84.45
	SOVH	88.71	89.88	89.52	88.37
	SOVE	80.64	84.57	82.91	82.12
	SOVC	81.84	84.31	83.71	82.57

4.1.3 Discussions

- *Prediction with High Confidence*

The feed-forward neural networks used in SCORPION provide a confidence interval to estimate the uncertainty of the prediction of each residue. When above 90% confidence is obtained, the secondary structure prediction of a residue has rather high

accuracy (98% for helices, 94% for sheets, and 90% for coils in 3-state prediction and 99% for α-helix and 98% for β-sheet in 8-state prediction). Therefore, if consecutive residues in a helix or sheet segment are predicted with high confidence, misprediction of this helix or sheet segment is very unlikely. This is particularly useful in a variety of applications such as assigning secondary structures to NMR constraints or Cryo-EM density maps as well as limiting backbone torsion angle variations to reduce degree of freedoms in template-free predictions [96].

Table 7 compares the total number of residues predicted with over 90% confidence in CB513, Manesh215, Carugo338, and CASP in SCORPION with and without context-based score encoding. Overall there are 153,073 residues in these four benchmarks. For neural networks with PSSM-only encoding, the secondary structures of 60,222 (39.3% of all residues) residues are predicted with over 90% confidence. When context-based scores are incorporated, the total number of residue secondary structure predictions with over 90% confidence is enhanced by 15.5% to 69,537 (45.4% of all residues in benchmarks). Compared to the neural networks using PSSM only encoding, the numbers of residues predicted with over 90% confidence increase by 6.7%, 9.9%, and 42.3% in helices, strands, and coils, respectively.

- *A 3-State Prediction Example*

Figure 17 depicts an example of 3-state secondary structure prediction on protein 3NNQ chain A from CASP9 targets. The Q3 accuracy of PSSM only neural networks is 83.33%. When context-based score encoding is incorporated, the Q3 accuracy is thereby improved to 90.35%. The main prediction difference is on the highlighted α-helix where the PSSM-only neural networks miss.

Table 7 Total number of correct predictions with over 90% confidence on benchmarks of CB513, CASP9, Manesh215, and Carugo338.

	PSSM Only	PSSM+Score
# of residues predicted as H with 90% confidence	33,292	35,533
# of residues predicted as E with 90% confidence	13,298	14,611
# of residues predicted as C with 90% confidence	13,632	19,393
Total # of residues predicted with 90% confidence	60,222	69,537

Nevertheless, the context-based scores of the residues in the highlighted helix segment, as shown in Table 8, indicate that secondary structure of helix is highly favorable, which help the neural networks to identify the major part of this helix.

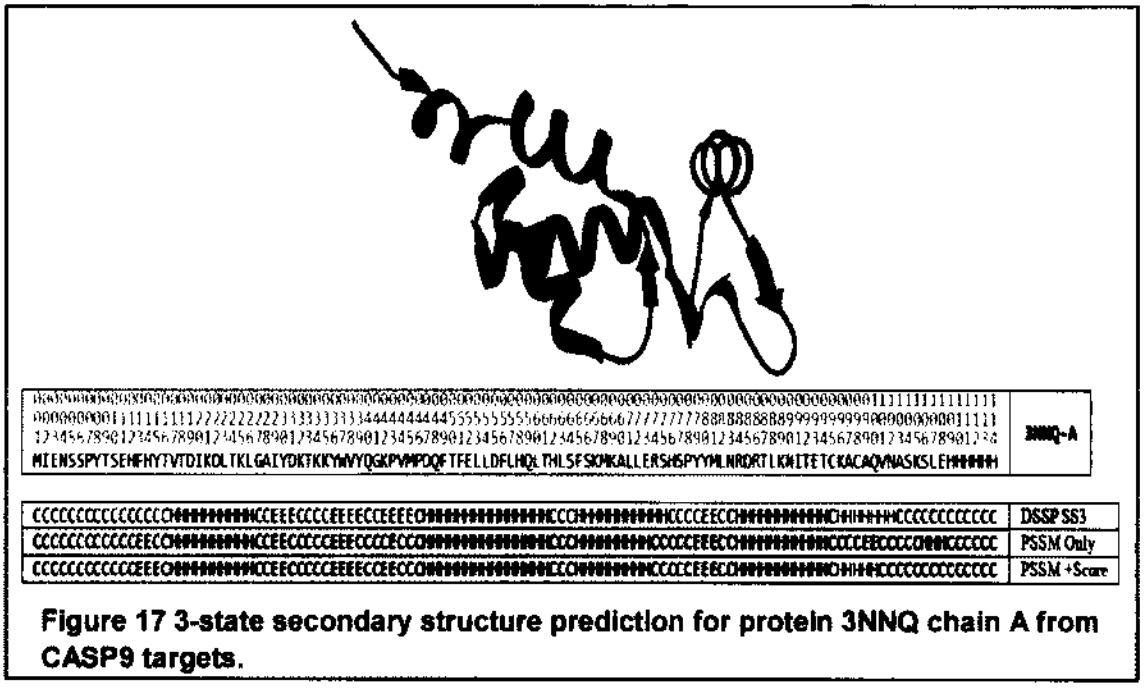


Figure 17 3-state secondary structure prediction for protein 3NNQ chain A from CASP9 targets.

Table 8 Segment of 3NNQA (82-115).

	AA	Context-based scores			Structure Predictions		True Structure
		U(C, ...)	U(E, ...)	U(H, ...)	PSSM+Score	PSSM only	
94	E	1.57	0.73	-1.66	H	H	H
95	T	3.17	0.43	-3.12	H	H	H
96	C	0.12	0.37	-0.08	C	C	C
97	K	5.11	1.01	-5.11	H	C	H
98	A	1.91	0.88	-2.11	H	C	H
99	C	5.5	0.58	-5.31	H	C	H
100	A	0.49	0.5	-0.53	H	E	H
101	Q	2.18	0.22	-2.22	H	E	H
102	V	0.91	-0.1	-0.65	C	C	H
103	N	2.13	0.22	-2.11	C	C	H
104	A	-3.66	0.27	3.68	C	C	C
105	S	-0.51	0.57	0.28	C	C	C

- *Analysis of Misclassifications*

The majority of the prediction errors in the set of benchmarks are resulted from the misclassifications of type E and C with 8.34% and the misclassification of type H and C with 7.98%. The misclassification of H and E is much less common with ~1.14%. Table 9 shows the total number of misclassifications on the benchmark set of CB513, CASP9, Manesh215, and Carugo338. A significant reduction of misclassifications is observed upon the incorporation of context-based score encoding in the neural networks in SCORPION. Misclassifying H to E and E to H has been reduced by 14.08%, misclassifying H to C and C to H has been reduced by 8.83% and the misclassification of E to C and C to E has been reduced by 5.18%.

Table 9 Misclassifications of secondary structure states on benchmark sets.

Misclassifications	PSSM Only	PSSM + Score
H -> E	1864	1717
E -> H	2318	1876
H -> C	12997	11436
C -> H	11550	10943
E -> C	15125	13831
C -> E	10383	10356

4.2 C8-SCORPION

Compared to the general 3-state secondary structure, DSSP program has more detailed classifications of secondary structures to eight states, including 3-10 helix (G), α-helix (H), π-helix (I), β-stand (E), bridge (B), turn (T), bend (S), and others (C). The 8-state secondary structures convey more precise structural information than 3-state, which is particularly important for a variety of protein structure modeling applications. For example, accurate 8-state secondary structures predictions can restrict the variations of backbone torsion angles within a smaller range according to the Ramachandran plot and thus reduce the search space in template-free protein tertiary structure modeling.

Moreover, differentiations among the different helical types (3_{10}-helix, α-helix and π-helix) in secondary structure prediction aid to assign residues and fit protein structure models in cryo-electron microscopy density maps [97].

4.2.1 Method Implementation

In this method, similar to C3-SCORPION using context-based model, we derive the mean-force potential scores to estimate the favorability of a residue in adopting a specific secondary structure state among the 8 states, within its amino acid environment. These mean-force potentials are combined with PSSM data to train a two-stage neural network for 8-state secondary structure prediction.

4.2.2 Results of C8-SCORPION

The overall Q_8 7-fold cross validated accuracy on Cull7987 data set in C8-SCORPION is 71.5%, where the accuracies of predicting 3_{10}-helix (G), α-helix (H), π-helix (I), extended strand (E), isolated bridge (B), bend (S), turn (T) and coil (C) are 22.7%, 92.4%, 0%, 82.9%, 2.4%, 22.3%, 51.6%, and 66.1%, respectively. The accuracy comparison with prediction without using context-based score encoding is listed in Table 10. The prediction accuracies of the eight different secondary structure states vary significantly. In particular, the prediction accuracy of G, I, B, and S are very low, mainly due to the fact of their infrequent appearances in protein data banks (PDB), whose distribution is shown in Figure 5. Hence, the 8-state classification is considered more challenging than the 3-state, due to the extremely unbalanced distribution of the 8-starctural states and their composition in native protein structures. As shown in Table 10, when the context-based scores are incorporated, accuracy enhancements are observed in all eight secondary structure classes, except for π-helix remaining at 0%. The very small fraction of residues adopting π-helix (0.02%) structure makes it almost impossible to predict.

To the best of our knowledge, SSpro8 and RaptorXss8 are the only two reported public accessible servers for 8-state secondary structure prediction. At the time when this dissertation is written, SSpro8 is not available online; however, RaptorXss8 has demonstrated higher accuracy than SSpro8 in [44].

Table 10 7-fold cross validation accuracy for 8-state prediction in SCORPION.

	Q_G	Q_H	Q_I	Q_E	Q_B	Q_S	Q_T	Q_C	Overall
PSSM Only	19.5%	91.8%	0.0%	82.1%	2.3%	19.4%	49.4%	65.5%	70.3%
PSSM+Score	22.7%	92.4%	0.0%	82.9%	2.4%	22.3%	51.6%	66.1%	71.5%

Table 11 and Table 12 compare Q8 and SOV8 accuracies of C8-SCORPION with RaptorXss8 server on CB513, CASP9, Manesh215, and Carugo338, respectively. Similar to 3-state prediction, in order to guarantee fairness, we generate a new set of context-based scores by removing all sequences with 25% or higher sequence identity to the sequences in the benchmarks from Cull16633 and all homologs with higher than 25% sequence identity to the chains presented in the benchmarks are excluded from Cull7987 when training 8-state prediction neural networks. SCORPION has a higher accuracy, in seven states except for π-helix, than RaptorXss8, with ~ 2% improvements in Q8 and ~ 3% improvements in SOV8.

Table 11 Comparison of Q8 accuracy between SCORPION and RaptorXss8 on benchmarks of CB513, CASP9, Manesh215, and Carugo338.

		CB513	CASP9	Manesh215	Carugo338
RaptorXss8	Q_8	65.59	69.31	67.69	66.64
	Q_G	17.54	20.58	18.43	19.20
	Q_H	89.96	92.90	90.22	89.91
	Q_I	0.00	0.00	0.00	0.00
	Q_E	77.68	81.64	79.60	79.45
	Q_B	0.09	0.00	0.32	0.44
	Q_S	15.87	18.11	17.80	17.14
	Q_T	48.02	51.45	51.28	50.11
	Q_C	63.29	59.37	63.73	63.36
C8-SCORPION	Q_8	67.22	71.54	69.71	68.44
	Q_G	21.81	22.46	23.01	22.42
	Q_H	90.95	93.58	91.42	90.55
	Q_I	00.00	0.00	0.00	0.00
	Q_E	80.31	83.95	82.54	81.44
	Q_B	1.43	1.04	1.79	2.22
	Q_S	19.86	23.41	21.99	21.95
	Q_T	49.44	53.87	53.03	52.65
	Q_C	63.54	62.82	64.93	64.69

Table 12 Comparison of SOV8 accuracy between SCORPION and RaptorXss8 on benchmarks of CB513, CASP9, Manesh215, and Carugo338.

		CB513	CASP9	Manesh215	Carugo338
RaptorXss8	SOV_8	64.99	69.84	68.00	66.88
	SOV_G	20.04	21.27	20.81	21.71
	SOV_H	88.95	90.71	89.97	89.24
	SOV_I	0.00	0.00	0.00	0.00
	SOV_E	82.50	84.81	84.15	84.61
	SOV_B	0.09	0.00	0.32	0.44
	SOV_S	17.72	19.74	19.40	18.78
	SOV_T	50.79	53.92	54.73	53.74
	SOV_C	55.13	59.61	58.78	58.01
C8-SCORPION	SOV_8	67.66	73.47	70.79	69.50
	SOV_G	25.39	26.41	27.23	26.01
	SOV_H	92.24	93.66	92.80	91.65
	SOV_I	0.00	0.00	0.00	0.00
	SOV_E	85.25	88.68	87.05	86.57
	SOV_B	1.43	1.04	1.78	2.21
	SOV_S	21.88	25.36	23.70	23.95
	SOV_T	52.98	56.97	56.71	56.89
	SOV_C	56.28	64.28	60.69	60.14

4.2.3 Discussions

We analyze the misclassifications among helices and strands in SCORPION. The benchmark set (CB513, Manesh215, Carugo338 and CASP9) include 10,011 helices ranging from 3- to 18-residue long and 13,877 strands from 1- to 16-residue long. The total number of helices and strands that are correctly predicted as whole structures using SCORPION, are 4,880 and 5,467 respectively (the percentages are 48.75% and 39.40% respectively); leaving the rest of the predictions with at least one residue misclassification in the structure. Table 13 shows a detailed analysis of the misclassifications in helices and strands in this benchmark set.

Figure 18 shows the length distributions of helices and sheets when the whole structures are missed from prediction in SCORPION. The most misclassified helices are 3-10 helices (all of three-residue helices and portion of longer helices) and the most misclassified sheets are isolated β-bridges (residue size of 1). 3-10 helices are much rarer

in PDB than α-helices, which has a very low prediction accuracy (~14%) in literature [98]. Also, most of the 3-10 helices are short with three or four consecutive residues. Similarly, the isolated β-bridges are also rare and short and probably more importantly, the isolated β-bridges are the results of global interactions. Generating statistically meaningful context-based scores to sensitively identify 3-10 helices and β-bridges is rather difficult. Therefore, the context-based scores help but with limited contribution in correctly identifying 3-10 helices and β-bridges.

Table 13 Detailed description of Helix and Strand misclassifications on benchmark set of CB513, CASP9, Manesh215, and Carugo338.

	Helix%	Strand%	Description
missed at begin (only)	5.36	8.28	only the first residue at the beginning of a structure
missed at middle	15.80	13.89	at least one is misclassified anywhere in between the two ends
missed at end (only)	11.59	8.87	only the last residue at the end of a structure
missed at begin & end (only)	1.70	2.08	misclassified at both ends (2 residues)
missed the whole structure	16.80	27.48	the whole structure misclassified

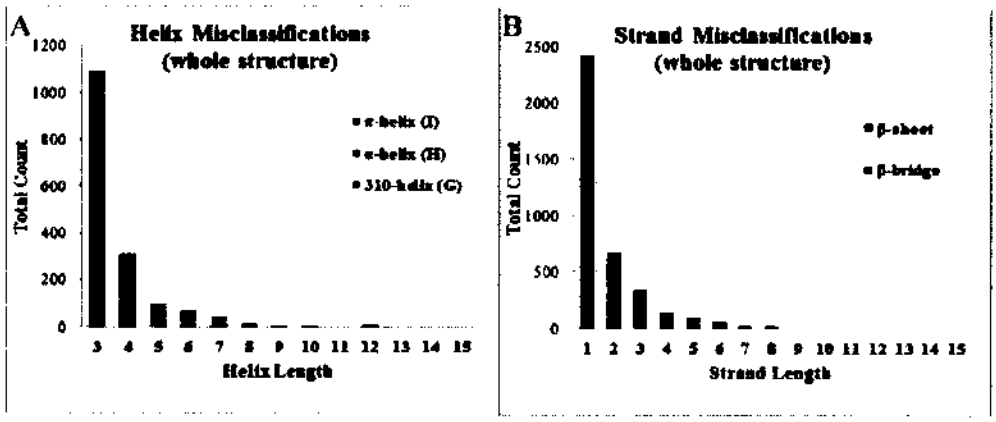


Figure 18 Histograms of the numbers of misclassified helices (A) and strands (B) in SCORPION by their lengths.

4.3 Template-based SCORPION

Most current methods for secondary structure predictions do not rely on similarity to proteins of known structures, in other words, these methods are simply *ab initio*, i.e., from sequence information only to predict the structural features.

However, we cannot neglect the fact that many protein sequences have some degree of similarity among themselves. Actually, over half of all known protein sequences have some detectable similarity (higher than 25%) to one or more sequences of known structures [99, 100]. Around 75% was reported as the percentage of those newly deposited protein structures in the PDB database showing significant similarity to previous deposited structures. Consequently, taking advantage of structural similarity of proteins with sequence similarity may lead to significant improvement of protein structure prediction. In fact, the latest version of Porter has used homology-based templates for 3-state secondary structure prediction [100]. Porter has been reported to achieve prediction accuracy improvement when known structures with >30% sequence similarity are available and even surpass the theoretical upper bound when such sequence similarity is higher than 50%.

Incorporating homology information is very useful in protein structure modeling. Hence, in this approach, we extract structural information from templates constructed for protein chains with known structures that exhibit homology to our training set. The structural information, when available, is incorporated as features together with the PSSM data. In the case where structural information is not available, context-based scores of secondary structures is incorporated instead, as sequence-structure features to train the neural networks for secondary structure prediction.

4.3.1 Method Implementation

- *Template construction*

Figure 19 illustrates the procedure of constructing structural templates. First of all, for a given protein sequence target, PSI-BLAST is used to search against the NR database with Evalue=0.001 and at most 3 iterations to generate the PSSM data. Then, the PSSM is used to search against the Protein Data Bank for alignments with Evalue=10.

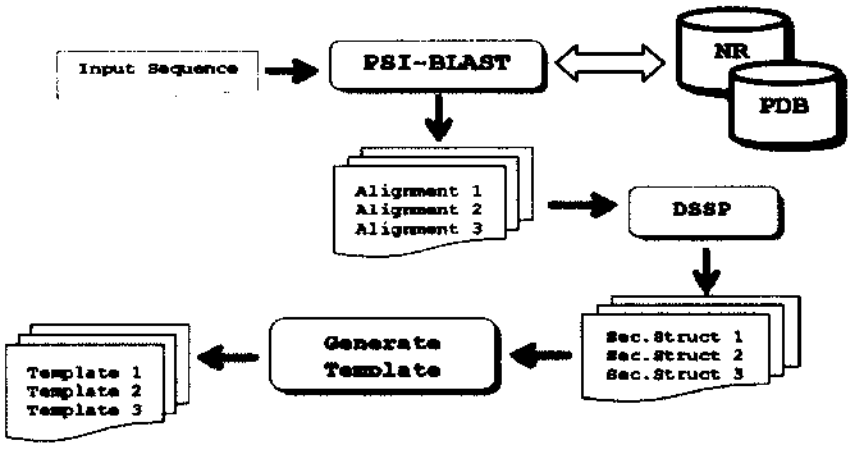


Figure 19 Construction of Templates.

If known structures are available in PDB, their 8-state assignments are determined by the DSSP program and then a structural template is built for the correspondent residue positions. Among the list of templates constructed, we select the top one that is less than 95% sequence similarity, according to PSI-BLAST ranking.

- *Encoding*

We use a window size of 15 residues for input encodings. Each residue is represented with 20 values from the PSSM data, 1 extra input to indicate if the residue window overlaps C- or N-terminal, 1 value for degree of similarity, and 8 values for structural information from template or context-based secondary structure scores. Hence, a total number of 450 values are used to describe each residue.

Figure 20 shows an example of encoding residues in a protein sequence. For a residue with available structural information in the template, the corresponding secondary structure state is set to 1 while the other states are set to 0. At the same time, the degree of similarity is set for the sequence similarity. On the other hand, if the structural information for a residue is not available in the template, the degree of similarity is set to zero and the context-based scores are incorporated instead. These pseudo-potentials are incorporated as context-based scores representing sequence-structure features in neural network training when structural information from templates is not available.

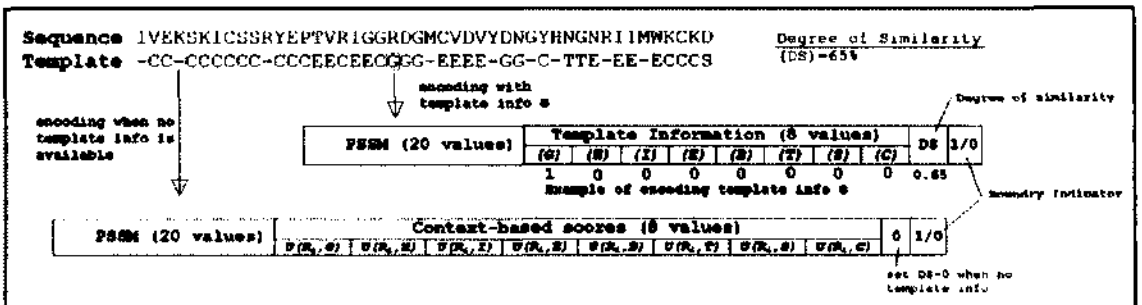


Figure 20 Encoding for template-based 8-state secondary structure prediction.

- *Neural network model*

We incorporate two phases of standard feed-forward neural network training for the prediction method. Figure 21 shows the encoding diagram and the two-phase neural network architecture.

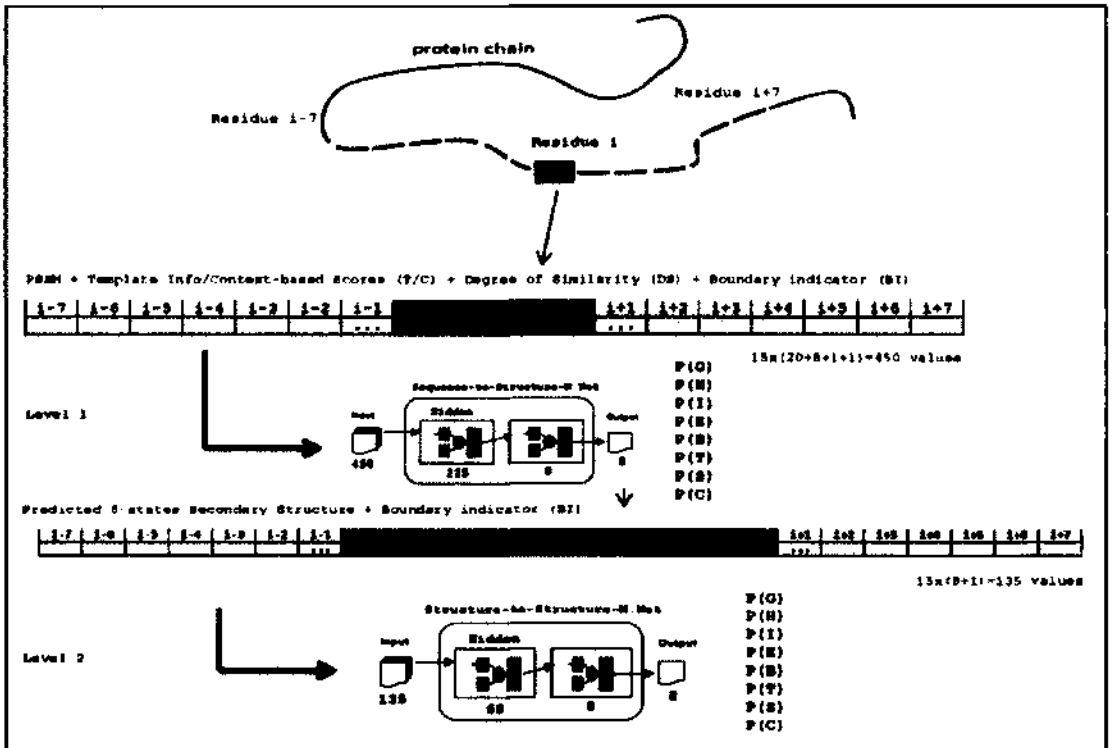


Figure 21 Two phases of template-based 8-state secondary structure prediction (architecture & encoding).

4.3.2 Results of Template-based SCORPION

Upon the selection of the best alignment with similarity less than 95% for all protein chains in the Cull5547 dataset, the final Q8 seven-fold cross validated accuracy after applying the template-based 8-state prediction reaches 78.85%. Table 14 lists the Q8 and SOV8 accuracies of 7-fold cross validation for each state. Table 15 compares the Q8 and SOV8 accuracy of using predictions with and without templates on benchmarks of CB513, CASP9, Manesh215, and Carugo338. Clearly, when homology structural information is available, the 8-state prediction accuracy is significantly improved. It is also interesting to find that when structural templates are used, the 8-state prediction accuracy improvement in CASP9 is much less than the other benchmark sets. This is due to the fact that in the CASP9 experiment, targets are deliberately selected to have relatively low similarity to sequences with existing structures in PDB.

Figure 22 shows the distribution of the prediction accuracy as a function of sequence similarity in levels in CB513, CASP9, Manesh215, Carugo338 as well as Cull5574 in cross-validation. Without surprise, the better templates with higher sequence similarity level, the more accurate the prediction results. More importantly, even templates with only 20%~30% sequence similarity can improve the prediction accuracy by near 5% in various benchmark sets compared to predicted results without templates.

Table 14 7-fold cross-validation accuracy in template-based 8-state prediction.

	G	H	I	E	B	S	T	C	Overall
Q_8	43.99	92.48	0.00	88.30	27.86	43.46	64.18	75.51	78.85
SOV_8	47.96	95.19	0.00	92.77	27.57	45.32	66.64	71.45	80.10

Table 15 Comparison between 8-state predictions with and without template on CB513, CASP9, Manesh215, and Carugo338.

	Q_8		SOV_8	
	No-Template	With-Template	No-Template	With-Template
CB513	67.22	79.39	67.66	80.64
CASP9	71.54	76.36	73.47	78.15
Manesh215	69.71	81.10	70.79	82.99
Carugo338	68.44	80.39	69.50	81.95

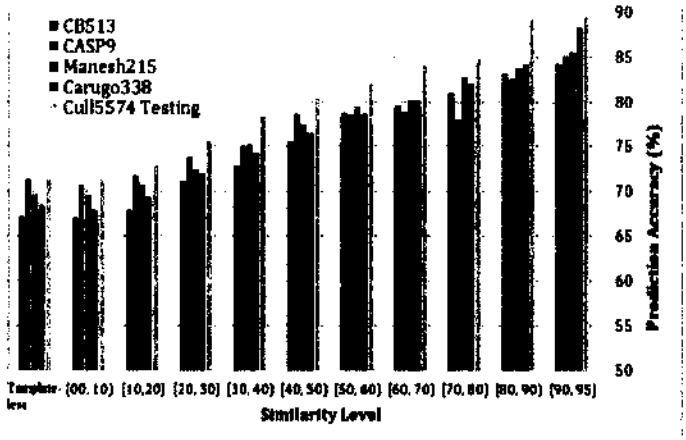


Figure 22 Distribution of 8-state secondary structure prediction accuracy (Q8) as a function of sequence similarity- the first group of bars corresponds to template-less predictions.

Figure 23 uses the A chain of protein 1BTN as an example to demonstrate the effectiveness of template-based 8-state secondary structure prediction. Prediction without template has 73.6% Q8 accuracy.

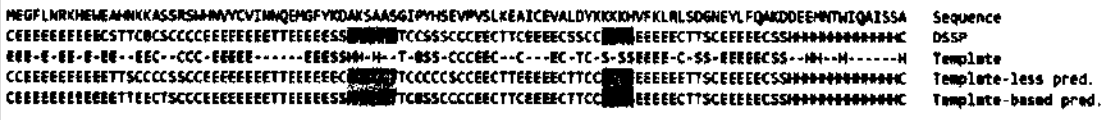


Figure 23 Comparison between template-less and template-based predictions on 1BTN chain A.

The best template found in PDB has 61% sequence similarity. Under the guidance of the structural template, the mispredicted helix segment and bend segment in template-less prediction (highlighted in Figure 23) are corrected, which leads to overall 89.6% Q8 accuracy.

4.3.3 Discussions

As shown in Table 14, the prediction accuracies for different states vary largely due to the very unbalanced appearing frequencies of the eight states in protein structures. In this work, we are particularly interested in the effectiveness of structural templates in improving the prediction accuracies of those states with low accuracy in prediction without templates. From Cull5547, we create five subsets of chains that have structural templates with similarity level in intervals of (0%, 10%], (10%, 20%], (20%, 40%], (40%, 70%], and (70%, 95%], respectively. Then, 7-fold neural network trainings are carried out for each subset and the average cross validation prediction accuracy for each state is reported in Table 16.

For α-helices (H), the prediction accuracy using templates with very low sequence similarity (0%, 10%] is already rather high (92.05%), mainly because there are sufficient number of α-helix samples available and the formation of α-helix is mainly result from local interactions. Anyway, the structural templates help refine the α-helix predictions with slight accuracy improvements. When structural templates with 40% or better similarity are available, the prediction accuracy of β-sheets (E) is also improved to above 90%, reaching the theoretical upper bound in secondary structure prediction. 40%+ similarity templates also significantly improve the accuracies of 3-10 helices (G) and bends (S) from 20%+ to 50%+. Similar but not as significant improvements are found in turns (T) and coils (C). However, the prediction results for bridges (B) and π-helices (I) are disappointing. Only when templates with very high similarity (>70%) are available, we can obtain 44% prediction accuracy in bridges (B). The prediction accuracy for π-helices (I) is still 0%. This is mainly due to the facts that π-helices are extremely rare (0.02%) and π-helices (I) are often misclassified into α-helices (H).

Table 16 Comparison of 7-fold cross validation prediction accuracies in eight states when templates with different sequence similarities are used.

	(0, 10]	(10, 20]	(20, 40]	(40, 70]	(70, 95]
# of chains	4,426	4,215	3,204	1,437	1,133
Q_H	92.05	92.70	93.60	94.97	95.94
Q_G	22.07	23.93	35.09	55.03	69.44
Q_I	0.00	0.00	0.00	0.00	0.00
Q_E	83.37	84.53	86.59	90.16	93.61
Q_B	1.53	3.59	7.24	22.30	44.26
Q_T	53.35	55.34	60.89	69.66	77.06
Q_S	22.83	26.41	35.19	54.09	73.40
Q_C	66.55	67.84	71.81	79.56	86.80
Q_8	71.33	73.01	76.29	82.11	88.01

CHAPTER 5

DISULFIDE BONDING PREDICTION

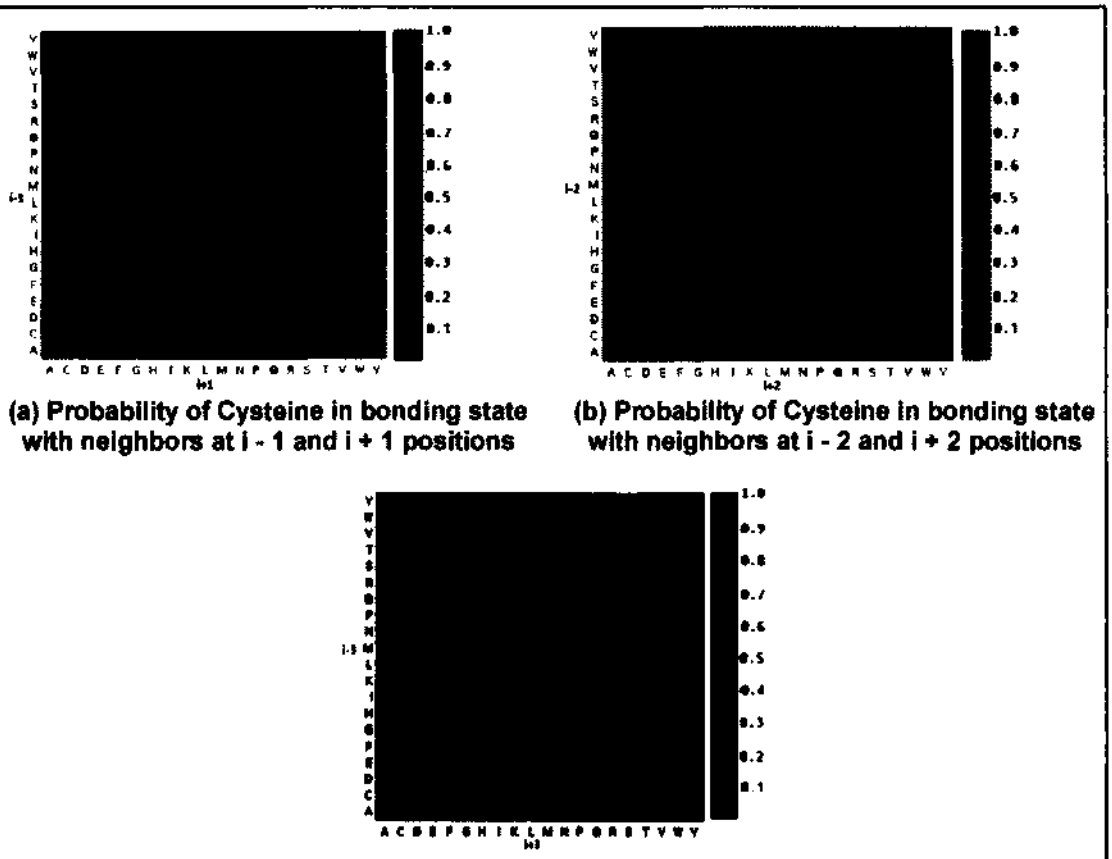
Disulfide bonding prediction is accomplished in two stages. The first stage is the bonding state prediction, whose goal is to determine whether each Cysteine residue in a protein chain is involved in forming a disulfide bond or not. Afterward, the second stage carries out the connectivity prediction, where Cysteine pairs likely to form disulfide bonds are identified.

Our approach of generating context-based features and then combining them with PSSM data is applied to develop our method for predicting disulfide bonds. Our disulfide prediction algorithm is implemented on a web server named "DINOSOLVE" available at: http://hpcr.cs.odu.edu/dinosolve [101].

5.1 DINOSOLVE

The neighboring residues have strong and probably deterministic influence to the chemical property of Cysteine residues in forming disulfide bond [102]. Actually, Cysteine often forms particular motifs of biochemical functions with neighboring residues, such as Cys-X-X-Ser [103], Cys-X-X-Cys [104], Leu-X-Cys-X-Glu [105], Cys-X-X-Asp-X-X-Cys [106], etc. Figure 24(A), (B), and (C) show the probability of Cysteine at position i in disulfide bonding state with the neighboring residues at i-1 and i+1, i-2 and i+2, and i-3 and i+3 positions, respectively. One can notice that the neighboring residues separated by two residues in the middle still have strong influences on the bonding state of the center Cysteine residue.

Hence, capturing the correlations among residues, where Cysteine residues are present, and then incorporate these correlations as features into the learning process of a disulfide bond predictor can enhance the prediction accuracy.



(a) Probability of Cysteine in bonding state with neighbors at i - 1 and i + 1 positions

(b) Probability of Cysteine in bonding state with neighbors at i - 2 and i + 2 positions

(b) Probability of Cysteine in bonding state with neighbors at i - 3 and i + 3 positions

Figure 24 Probability of Cysteine in disulfide bonding state with neighbors at different positions.

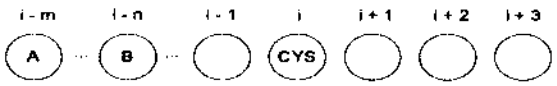
5.1.1 Method Implementation

- *Context-based statistics*

We derive the first-order and second-order mean-force potentials according to the amino acid environment around the Cysteine residues from large number of Cysteine samples, collected from Cull16633. The mean-force potentials are integrated as context-based scores to estimate the favorability of a Cysteine residue in disulfide bonding state as well as a Cysteine pair in disulfide bond connectivity. These context-based scores are then incorporated as features together with other sequence and evolutionary information to train neural networks for disulfide bonding state prediction and connectivity prediction.

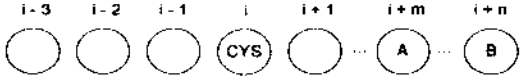
The first-order statistics estimate the correlations between a Cysteine residue and one of its neighboring residues while the second-order statistics estimate the correlations between a Cysteine residue and the coexistence of two neighboring residues. Both first-order and second-order statistics are extracted from protein chains in the Cull16633 dataset. For a Cysteine sample with window size of K, there are K-1 position combinations for first-order statistics in total. Figure 25 shows the three possible situations of two neighbors relative to a Cysteine residue when extracting second-order statistics, including (a) both neighbors on the left; (b) two neighbors on both sides; and (c) both neighbors on the right. Therefore, considering a window size of K for a Cysteine sample, there are totally $\binom{K-1}{2}$ position combinations for the second-order statistics of a Cysteine residue in bonding state.

Similar to the bonding state statistics, the first-order and second-order statistics of a disulfide bonded Cysteine pair related to its neighboring residues are also extracted from the Cull dataset. These statistics are used to estimate the probability of a Cysteine pair in forming disulfide connectivity.



(a) Both neighbors on the left hand side of Cysteine

(b) Two neighbors on both sides of Cysteine



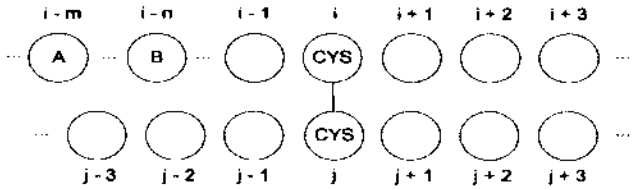
(c) Both neighbors on the right hand side of Cysteine

Figure 25 Three possible positions of two neighbors to a Cysteine residue.

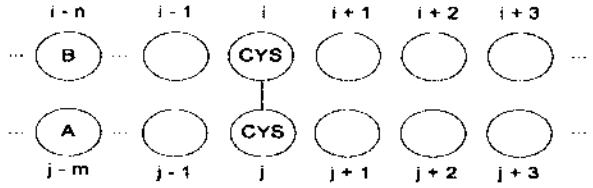
Compared to the statistics in estimating a Cysteine residue in a bonding state, the main difference lies in the different number of position combinations in second-order statistics since the two neighboring residues may belong to two different Cysteine residues. Figure 26(a) shows the situation that both neighboring residues belong to one Cysteine residue and Figure 26(b) shows the situation that the two neighboring residues belong to different Cysteine residues. Therefore, considering a window size of K for both Cysteine residues connected in a disulfide bond, there are totally $\binom{2K-2}{2}/2$ position combinations for the second-order statistics of a bonding Cysteine pair.

Similar to our method in predicting secondary structures, to obtain more precise neighboring correlation statistics to disulfide bonding states, we consider the divergence of a protein sequence in its structural family by using the PSSM data specifying the frequency of each amino acid type in a protein multiple sequence alignment.

Let R_i denote residue R at position i in a protein sequence and let $R_{(j)}$ denote residue R at relative position j to a cysteine residue. In the first-order statistics, the observed probability, $P_{obs}(Bonded|R_{(k)})$, of residue type R with relative distance k to a bonded cysteine in a specific protein data set is estimated as



(a) Both residues are the neighbors of one Cysteine residue



(b) One residue is the neighbor of Cysteine i and the other is the neighbor of Cysteine j

Figure 26 Possible positions of two neighbors to a Cysteine residue pair in disulfide bond.

$$P_{obs}(Bonded|R_{(k)}) = \frac{\sum_{protein}\sum_{CYS_i\ is\ bonded} PSSM(R_{i+k}) * PSSM(CYS_i)}{\sum_{protein}\sum_{CYS_i\ is\ bonded} PSSM(CYS_i)},$$

where $PSSM(R_i)$ is the PSSM frequency of residue type R at position i in a protein sequence. Similarly, in the second-order statistics, the observed probability, $P_{obs}(Bonded|R_{(k_1)}, R_{(k_2)})$, of the coexistence of residues $R_{(k_1)}$ and $R_{(k_2)}$ to a bonded cysteine is estimated as

$$P_{obs}(Bonded|R_{(k_1)}, R_{(k_2)}) = \frac{\sum_{protein}\sum_{CYS_i\ is\ bonded} PSSM(R_{i+k_1}) * PSSM(CYS_i) * PSSM(R_{i+k_2})}{\sum_{protein}\sum_{CYS_i\ is\ bonded} PSSM(CYS_i)}.$$

The neighboring correlation statistics to the disulfide bonding pair are obtained in a similar manner as bonding state.

- *Context-based potential*

In this method, we consider the first-order and the second-order mean-force potentials only. Currently, there is insufficient number of available protein structures in PDB to derive meaningful statistics for estimating higher order interactions.

According to the inverse-Boltzmann theorem, we introduce the first-order mean-force potential $U(R_{(k)}, Bonded)$ to treat the interaction between residue $R_{(k)}$ and cysteine in forming a disulfide bond,

$$U(R_{(k)}, Bonded) = -RTln\frac{P_{obs}(Bonded|R_{(k)})}{P_{ref}(Bonded|R_{(k)})}$$

Here R is the gas constant, T is the temperature, and $P_{ref}(Bonded|R_{(k)})$ is the reference state, which is estimated as

$$P_{ref}(Bonded|R_{(k)}) = \frac{\sum_{protein}\sum_{CYS_i} PSSM(R_{i+k}) * PSSM(CYS_i)}{\sum_{protein}\sum_{CYS_i} PSSM(CYS_i)}.$$

Similarly, the second-order mean-force potential $U(R_{(k_1)}, R_{(k_2)}, Bonded)$ is calculated as

$$U(R_{(k_1)}, R_{(k_2)}, Bonded) = -RTln\frac{P_{obs}(Bonded|R_{(k_1)}, R_{(k_2)})P_{ref}(Bonded|R_{(k_1)})P_{ref}(Bonded|R_{(k_1)})}{P_{ref}(Bonded|R_{(k_1)}, R_{(k_2)})P_{obs}(Bonded|R_{(k_1)})P_{obs}(Bonded|R_{(k_1)})}$$

with the second-order reference state,

$$P_{ref}\left(Bonded|R_{(k_1)},R_{(k_2)}\right) = \frac{\sum_{protein}\sum_{CYS_i} PSSM\left(R_{i+k_1}\right) * PSSM(CYS_i) * PSSM\left(R_{i+k_2}\right)}{\sum_{protein}\sum_{CYS_i} PSSM(CYS_i)}.$$

Influenced by all of its neighboring residues, the overall mean-force potential for the interactions of a cysteine residue in bonding state is the summation of all first-order and second-order potentials while the higher-order interactions are ignored

$$U(CYS_i, Bonded) = \sum_{k}^{k\neq 0} U\left(R_{(k)}, Bonded\right) + \sum_{k_1}^{k_1\neq 0}\sum_{k_2}^{k_2\neq 0} U\left(R_{(k_1)}, R_{(k_2)}, Bonded\right).$$

The potential $U(CYS_i, CYS_j, Connected)$ for a bonded cysteine pair CYS_i and CYS_j can be obtained in a similar way. These potentials are used as context-based scores to be encoded in neural network training for bonding state and connectivity predictions.

- *Neural network model*

We adopt the standard feed-forward back-propagation neural network architecture for both disulfide bonding state prediction and connectivity prediction. The neural network for bonding state prediction uses a window size of 15 residues for input encodings. Each residue is represented with 20 values from the PSSM data and 1 extra input to indicate if the window overlaps C-terminal or N-terminal. When incorporating the context-based scores in training the neural network predictor, two more inputs specifying the scores of the cysteine residue being in free and bonding state are added. Hence, a total number of 317 values are used to describe each cysteine residue. 100 hidden nodes are used in the neural network for bonding state prediction.

The neural network for connectivity prediction incorporates two windows, each with size of 15 residues, for input encoding. Each window encodes the amino acid environment of a cysteine residue in a cysteine pair. Each residue is encoded with 20 PSSM values and 1 boundary indicator. The predicted results (bonded or free) from the bonding state prediction for both cysteine residues and the context-based scores for connectivity are also encoded as input. As a result, there are totally 636 input values for each cysteine pair. 150 hidden nodes are used in the neural network for connectivity prediction. Figure 27 and Figure 28 illustrate the encoding and neural network architecture for disulfide bonding state and connectivity prediction, respectively.

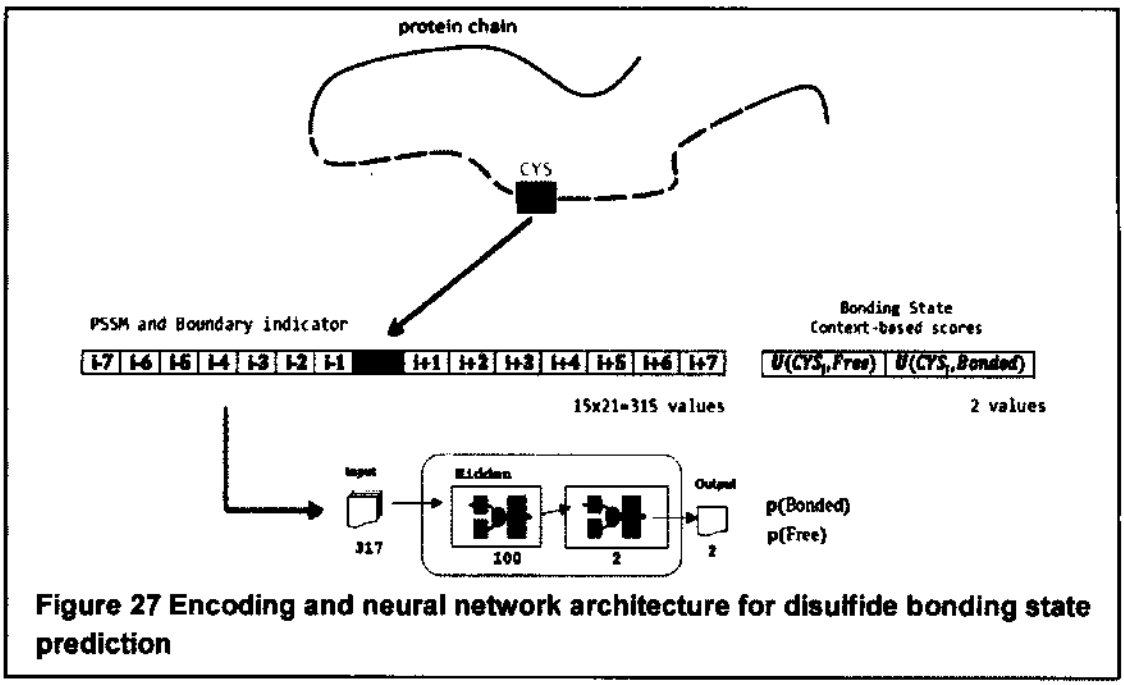


Figure 27 Encoding and neural network architecture for disulfide bonding state prediction

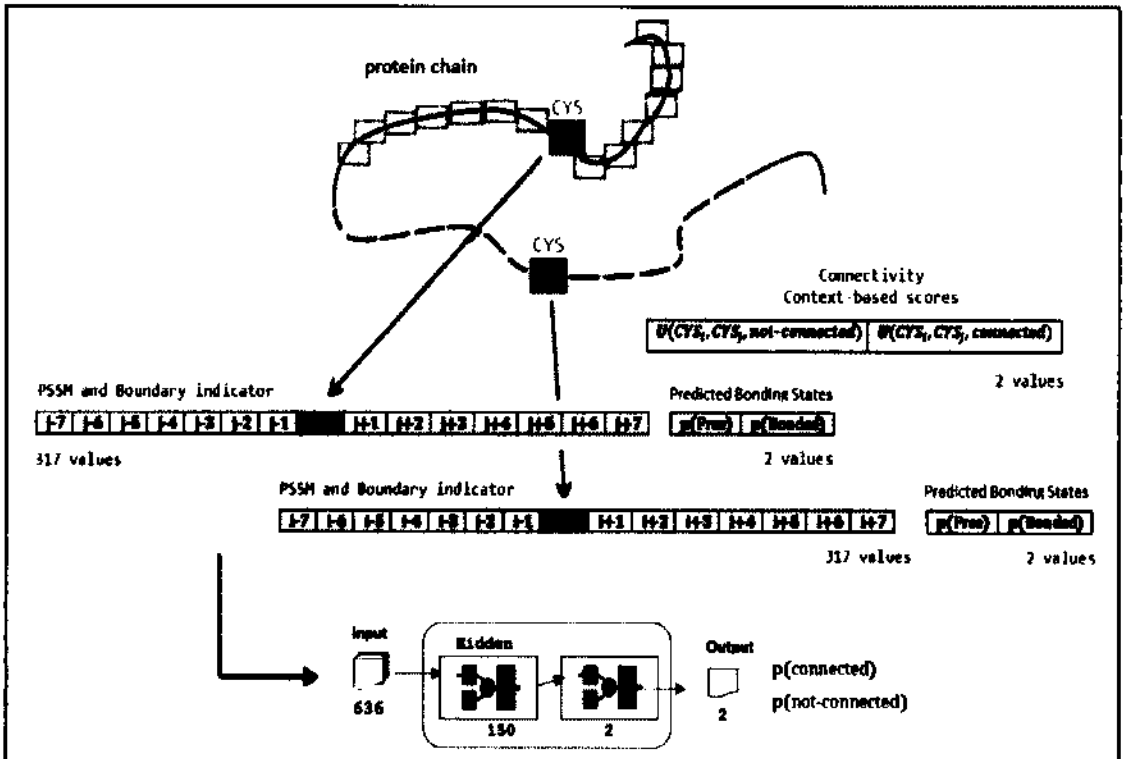


Figure 28 Encoding and neural network architecture for disulfide connectivity prediction.

5.1.2 Results of DINOSOLVE

- *The bonding state prediction*

Table 17 compares the prediction qualities of bonding states with PSSM-only encoding and PSSM with context-based scores encoding after 10-fold cross validation. Compared to the one trained with PSSM data only, the neural network using context-based scores as additional features results in improvements in all performance indexes, including Sn, Sp, Qc, Qp, and Mcc. The residue-level prediction accuracy (0.908) and protein-level prediction accuracy (0.856) are higher than the reported accuracies in popular disulfide bond prediction servers. Table 17 also compares the prediction qualities when Cull25 and Cull50 are used as training sets. Cull50 has more than twice cysteine samples as Cull25, which leads to better prediction performance than Cull25.

- *Connectivity prediction*

Table 18 compares the computational results of 10-fold cross validation for disulfide bond connectivity predictions on Cull50 using PSSM-only and PSSM with context-based scores for neural network encoding. Similar to bonding state prediction, one can find that incorporating the context-based scores as features in neural network training enhances the connectivity prediction accuracy, where sensitivity (Sn), specificity (Sp), and overall accuracy (Qc) are improved from 73.07%, 91.03%, and 86.91% to 73.42%, 91.61%, and 87.34%, respectively, compared to PSSM only encoding. These prediction results are also higher than the reported disulfide connectivity accuracies in the popular disulfide bond prediction servers.

Table 17 Comparison of prediction performance of bonding states using PSSM only and PSSM with context-based scores on Cull25 and Cull50 using 10-fold cross validation.

	Cull25		Cull50	
	PSSM Only	PSSM+Score	PSSM Only	PSSM+Score
S_n	0.554	0.616	0.655	0.720
S_p	0.945	0.956	0.947	0.959
Q_c	0.870	0.888	0.885	0.908
Q_p	0.719	0.751	0.829	0.856
Mcc	0.574	0.646	0.734	0.801

Table 19 lists the prediction results on protein chains in Manesh215, Carugo338, and CASP9, which include at least one disulfide bond. The percentage of chains where all disulfide bonds are correctly predicted is 87.8%.

Table 18 Computational results of 10-fold cross validation on Cull50 using PSSM only and PSSM + Score in neural network encoding.

	PSSM only			PSSM + Score		
fold	Sn	Sp	Qc	Sn	Sp	Qc
1	73.90	91.60	87.50	74.90	91.60	87.70
2	72.80	93.00	88.10	71.70	93.10	88.00
3	70.70	91.90	86.50	71.40	92.40	87.10
4	78.80	82.30	82.20	77.80	84.10	82.60
5	75.20	91.40	87.60	74.10	92.00	87.80
6	71.40	92.30	87.70	71.30	93.00	88.10
7	74.50	92.40	88.50	76.00	92.40	88.80
8	66.80	93.60	87.40	70.40	93.30	88.00
9	69.00	90.20	85.20	68.40	91.50	86.10
10	77.60	91.60	88.40	78.20	92.70	89.20
Average	73.07	91.03	86.91	73.42	91.61	87.34

Table 19 Prediction performance on protein chains in Manesh215, Carugo338, and CASP9.

	Manesh215		Carugo338		CASP9		All	
# of disulfide bonds	# of chains	# of correctly predicted	# of chains	# of correctly predicted	# of chains	# of correctly predicted	# of chains	# of correctly predicted
1	14	13	23	23	1	1	38	37
2	12	11	21	21	0	0	33	32
3	9	7	19	16	1	1	29	24
4	3	2	13	12	0	0	16	14
5	3	3	6	5	0	0	9	8
6	1	0	2	2	0	0	3	2
7	1	1	2	1	0	0	3	2
8	2	1	2	2	0	0	4	3
9	0	0	3	0	0	0	3	0
10	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	1	0	1	0
Summary					139		122 (87.8%)	

Figure 29 depicts an example of the disulfide connectivity prediction on protein 153L chain 'A' listed in Manesh215. The native 153L(A) structure has four cysteine residues: CYS(4), CYS(18), CYS(29), and CYS(60). CYS(4) is connected to CYS(60) and CYS(29) is connected to CYS(60) by disulfide bonds. In the bonding state prediction, the predicted bonding probabilities for CYS(4), CYS(18), CYS(29), and CYS(60) are 0.82, 0.84, 0.95, and 0.94, respectively, which are all higher than 0.5 indicating that they are all bonded. In the connectivity prediction, the predicted bonding probabilities for the potential disulfide bonds are listed in Table 20.

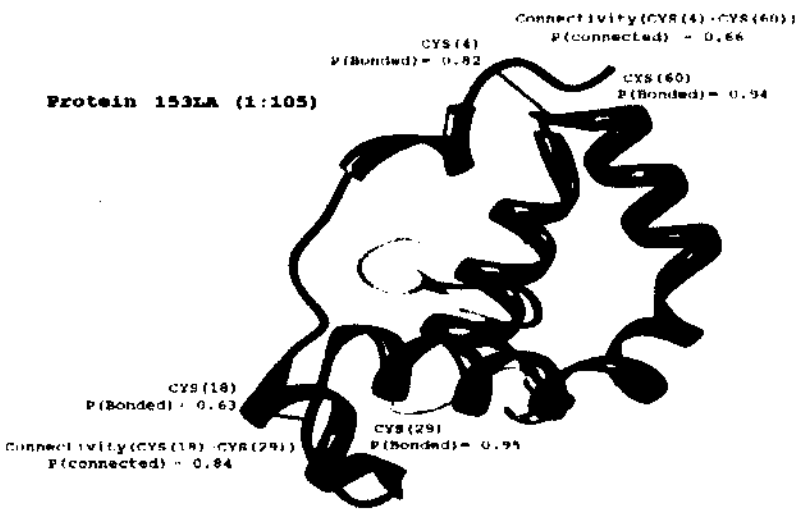


Figure 29 Disulfide connectivity prediction on protein 153L A chain.

Table 20 Predicted bonding probability for potential disulfide bonds in 153L(A).

Potential Disulfide Bonds	Predicted Bonding Probability
CYS(4)-CYS(18)	0.37
CYS(4)-CYS(29)	0.32
CYS(4)-CYS(60)	0.66
CYS(18)-CYS(29)	0.84
CYS(18)-CYS(60)	0.90
CYS(29)-CYS(60)	0.34

Form Table 20, one can find that CYS(18) and CYS(60) are most likely to be connected due to their highest predicted connectivity probability (0.90). However, if CYS(18) and CYS(60) are connected, CYS(4) and CYS(29) are unlikely to be connected due to their low predicted connectivity probability (0.32), which violates the predicted results during bonding state prediction. Therefore, an alternative connectivity pattern is selected with CYS(18)-CYS(29) and CYS(4)-CYS(60). This prediction result matches the disulfide connectivity pattern in the native structure of 153L(A).

5.1.3 Discussions

The context-based scores are effective features to enhance the neural network training process. When context-based scores are incorporated, the bonding state prediction accuracies are improved on all three benchmarks compared to those using PSSM data only. Table 21 compares the residue-level accuracies on the popularly used public benchmarks, including Manesh215, Carugo338, and CASP9. Similar to the computational results of 10-fold cross-validation, one can find that the Cull50 training set yields better prediction performance than Cull25.

Moreover, incorporating the context-based scores as features in neural network training enhances the connectivity prediction accuracy, where sensitivity (Sn), specificity (Sp), and overall accuracy (Qc) are improved from 73.07%, 91.03%, and 86.91% to 73.42%, 91.61%, and 87.34%, respectively, compared to PSSM only encoding.

One important question for generating the context-based statistics is how faraway the neighbors in sequence need to be involved.

Table 21 Comparison of residue-level accuracies (Qc) on benchmarks of Manesh215, Carugo338, and CASP9 using Cull25 and Cull50 as training sets.

	Cull25		Cull50	
	PSSM Only	PSSM+Score	PSSM Only	PSSM+Score
Manesh215	0.830	0.848	0.879	0.900
Carugo338	0.808	0.821	0.872	0.884
CASP9	0.950	0.951	0.955	0.963

Figure 30 compares the 10-fold cross validated accuracies when context-based features with different window sizes are used for neural network training. One can find that the context-based features with window sizes 3 and 5 slightly improve the prediction accuracy compared to using PSSM only. However, the context-based features with window size 7 yield the optimal performance. This is mainly due to the fact that the context-based features with window size 7 take the important i – i+3 residue correlations into account, where such correlations are often found in many motifs where cysteine is involved, such as Cys-X-X-Cys, Cys-X-X-Ser, Cys-X-X-His, Cys-X-X-Pro, Cys-X-X-Asp, etc. Another reason is, when the window size 7 is used, the residue-residue correlations in secondary structures are implicitly estimated, because helices, strands, and coils are strongly correlated at relative positions i-3 – i – i+3, i-2 – i – i+2, and i-1 – i – i+1, respectively [28]. It is also interesting to find that the prediction accuracy drops when the context-based features with window size 9 are employed. This is because the context-based scores with window size 9 integrate almost twice as many mean-force potential terms as scores with window size 7 – these additional terms measure the long distance inter-residue correlations of i – i+4, which are not as important as the shorter inter-residue correlations but accumulate the statistical sampling noise.

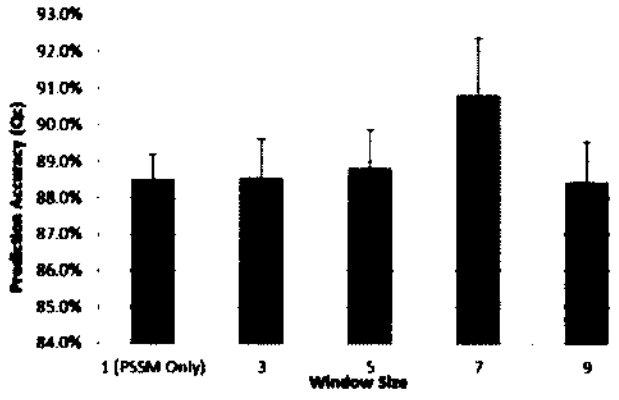


Figure 30 10-fold cross validated accuracies using context-based scores generated with different window sizes.

CHAPTER 6

SOLVENT ACCESSIBILITY

Correctly predicting the solvent accessibility of residues can not only reduce the conformational space to aid modeling protein structures in three dimensions, but also help predict important protein functions.

Our approach of generating context-based features and then combining them with PSSM data is applied to develop our method for predicting solvent accessibility. The prediction algorithm is implemented on a web server named "CASA" available at: http://hpcr.cs.odu.edu/casa [107].

6.1 CASA

We use the context-based model to derive statistics for singlets, doublets, and triplets in a sequence window from experimentally determined structures in PDB [10]. Then scores measuring the pseudo-energy of a residue adopting a certain accessibility state are calculated using potentials of mean force approach. The fundamental idea is based on the fact that the residue's solvent accessibility exhibit strong local dependency. These scores are then incorporated as features together with the multiple sequence alignment data to train neural networks for solvent accessibility prediction. We apply our approach to predict solvent accessibility in 2-state.

We test CASA on protein benchmarks, Manesh215 [16], Carugo338 [86], and CASP9 [108] targets. Lastly, we compare CASA with a set of popular methods for solvent accessibility prediction, including NETASA [58], Sable [59], Netsurf [60], SPINE [56], ACCpro [61] and SANN [64].

6.1.1 Method Implementation

The solvent accessibility values are determined by the DSSP program [12]. Relative values for residues' solvent accessibility are calculated as the ratio between the absolute solvent accessibility value and that in an extended tripeptide (Ala-X-Ala) conformation. Table 22 shows the extended state value of each amino acid reported by

Ahmad et al. [16] and used in many prediction methods. A threshold of 0.25 is used to define the 2-state solvent accessibility (Buried when the relative solvent accessibility value is less than 0.25, and Exposed otherwise).

- *Context-based statistics*

Presumably, the neighboring residues have strong influence to the chemical property of a residue in its accessibility to solvent. Figure 31 shows the probability of residue K at position i in buried accessibility state with the neighboring residues at i - 1 and i + 1, i - 2 and i + 2, and i - 3 and i + 3 positions, respectively. One can notice that the residues separated by two residues in the middle still have strong influences on the state of the center residue.

In this work, similar to the previous work in SCORPION and DINOSOLVE, we extract statistics of singlets (R_i), doublets (R_iR_{i+k}), and triplets ($R_iR_{i+k_1}R_{i+k_2}$) residues at different relative positions in protein sequences, which is further used to generate pseudo-potentials to be incorporated as new features in neural network training. The statistics of singlets, doublets, and triplets represent estimations of the probabilities of residues adopting a specific solvent accessibility state when none, one, or two of their neighbors in context are taken into consideration, respectively.

Table 22 Extended state values of amino acids.

A.A	Extended State ($Å^2$)		A.A	Extended State ($Å^2$)
Ala	110.2		Met	200.1
Asp	144.1		Asn	146.4
Cys	140.4		Pro	141.9
Glu	174.7		Gln	178.6
Phe	200.7		Arg	229.0
Gly	78.7		Ser	117.2
His	181.9		Thr	138.7
Ile	185.0		Val	153.7
Lys	205.7		Trp	240.5
Leu	183.1		Tyr	213.7

- *Context-based potential*

Similarly, the context-dependent pseudo-potentials are generated based on the potentials of mean force method. We calculate the mean-force potentials $U_{singlet}(R_i, C_i)$, $U_{doublet}(C_i, R_i R_{i+k})$ and $U_{triplet}(C_i, R_i R_{i+k_1} R_{i+k_2})$ for a residue R_i adopting solvent accessibility state C_i. Then, the context-dependent pseudo-potential for R_i under its amino acid environment is

$U(C_i, \ldots R_{i-1} R_i R_{i+1} \ldots) =$

$U_{singlet}(C_i, R_i) + \sum_k U_{doublet}(C_i, R_i R_{i+k}) + \sum_{k_1,k_2} U_{triplet}(C_i, R_i R_{i+k_1} R_{i+k_2})$.

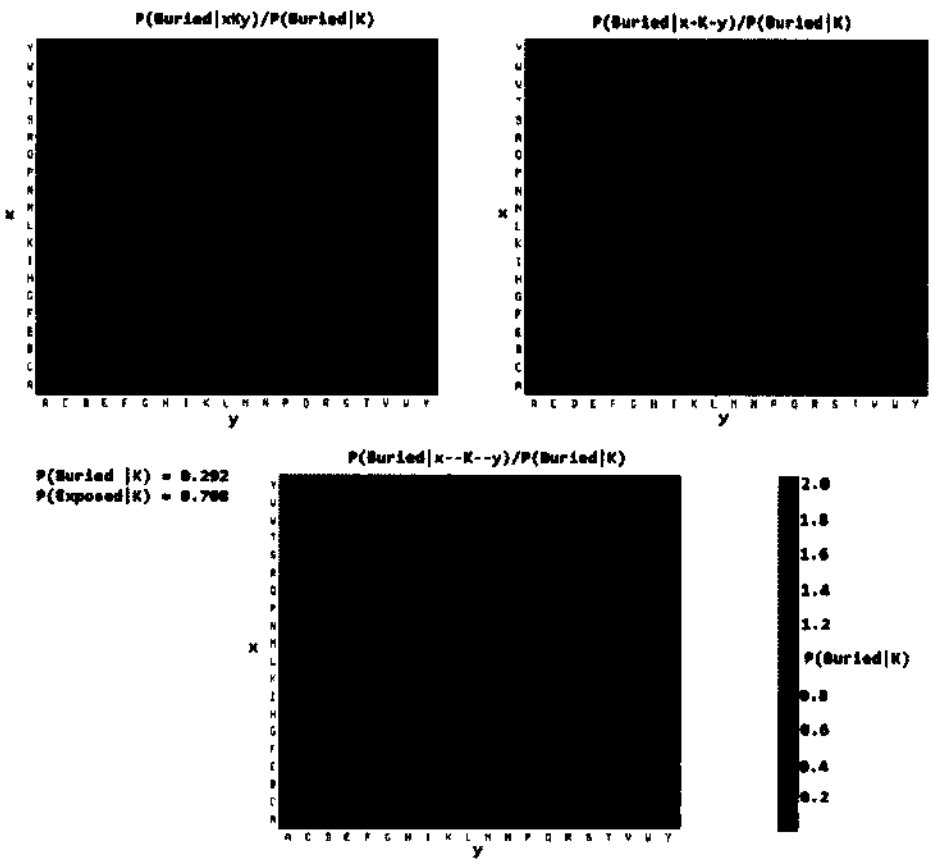


Figure 31 The probability of Lysine (K) as the middle residue of a triplet with neighboring residues at 1~3 positions away when adopting buried accessibility state. x, y, and - represent the left neighbor, right neighbor, and gap, respectively. The neighboring residues are ordered alphabetically.

- *Neural network model*

Our method for predicting protein solvent accessibility incorporates two phases of neural network training. Similar to SCORPION and DINOSOLVE, we adopt the standard feed-forward back-propagation neural network architecture. Figure 32 depicts the encoding and neural network architecture for CASA.

6.1.2 Results of CASA

We use Q_2 to measure the quality of our prediction method. We also use Q_B and Q_E to measure the quality of predicting the buried state and the exposed state, respectively. Table 23 compares the prediction qualities of solvent accessibility with PSSM-only encoding and PSSM with context-based scores encoding after 7-fold cross validation. Compared to the one trained with PSSM data only, the neural network using context-based scores as additional features results in improvements in the Q_2 accuracy, which is higher than the reported accuracies, 72-79%.

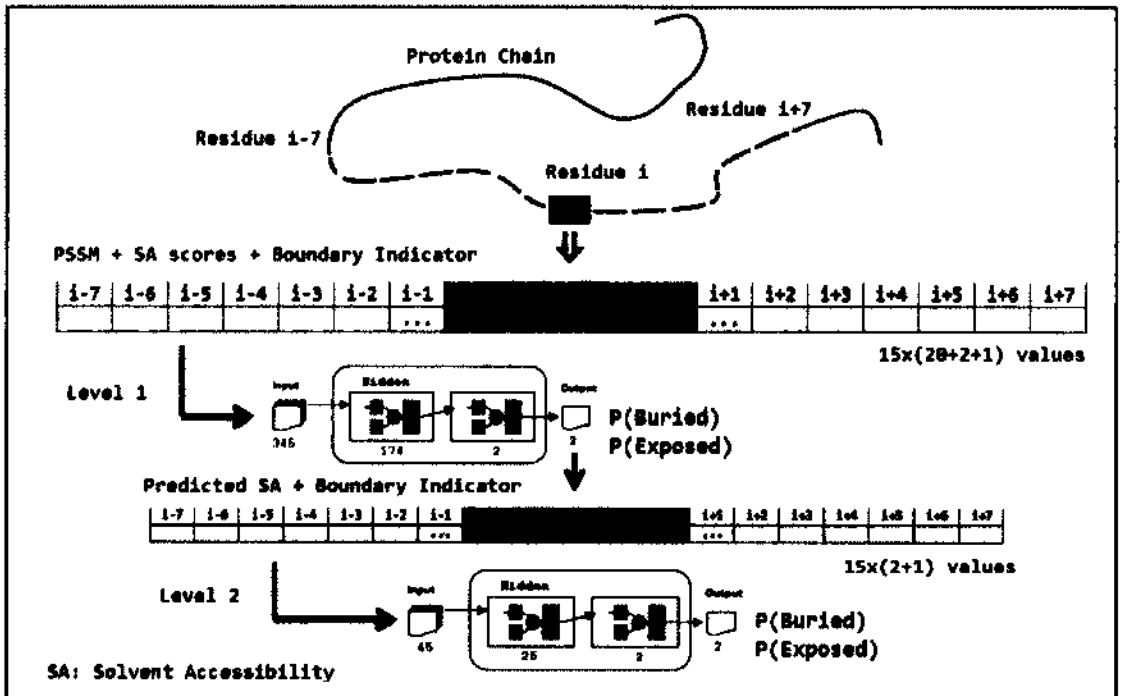


Figure 32 Encoding and neural network architecture for 2-state solvent accessibility prediction.

Table 23 Comparison of prediction performance of Solvent Accessibility using PSSM only and PSSM with context-based scores on Cull7987 using 7-fold cross validation.

	Q_B	Q_E	Q_2
PSSM Only	78.44%	80.61%	79.50%
PSSM+Score	79.21%	82.00%	80.76%

Table 24 compares the Q2 accuracy between our method and the popularly used solvent accessibility prediction servers including NETASA [58], Sable [59], Netsurf [60], SPINE [56], and ACCpro [61] on benchmarks of CASP9, Manesh215, and Carugo338. To guarantee fairness, we generate a new set of context-based scores by removing all sequences with 25% or higher sequence identity to the sequences in benchmark from Cull16633.

Table 24 Comparison of the accuracy between our method (CASA) and other popularly used solvent accessibility prediction servers including NETASA, Sable, Netsurf, SPINE, and ACCpro on benchmarks of CASP9, Manesh215, and Carugo338.

		CASP9	Manesh215	Carugo338
NETASA	Q_2	69.32	71.09	69.7
	Q_B	70.86	72.1	72.04
	Q_E	67.59	69.9	67.22
Sable t=0.2	Q_2	78.47	79.83	78.68
	Q_B	78.27	80.2	78.48
	Q_E	78.69	79.4	78.91
Sable t=0.3	Q_2	75.13	77.04	75.94
	Q_B	89.55	91.08	90.29
	Q_E	59.58	60.35	60.33
Netsurf	Q_2	79.15	80.83	80.04
	Q_B	80.04	83.35	81.27
	Q_E	78.19	78.49	78.13
SPINE	Q_2	77.86	80.5	79.68
	Q_B	83.22	85.3	85.33
	Q_E	72.08	74.8	73.53
ACCpro	Q_2	76.18	78.87	77.99
	Q_B	81.15	83.19	83.12
	Q_E	70.81	73.76	72.41
CASA	Q_2	80.82	81.93	81.14
	Q_B	81.46	84.27	83.65
	Q_E	80.13	79.14	78.39

The predictions of the benchmark sets are performed in 2-state for each method. Sable method provides 10-state predictions, with 10% difference among the states of solvent accessibility. Hence the results reported in Table 24, for sable method, are using 0.2 and 0.3 thresholds. We also compare our method with SANN [64] on benchmark of CASP9. The Q_2 accuracy of SANN on CASP9 is 77.86% such that the Q_B and Q_E are 69.68% and 86.66%, respectively.

We observe that CASA outperforms the other methods, where the Q_2 performance is higher in these benchmarks. However, when considering the predictions of the accessibility states individually, SPINE predictions of the buried state (Q_B) is better than CASA, with an average of 1.49% improvement. On the other hand, CASA provides a much higher exposed state prediction (Q_E) with an average of 5.75% difference compared to SPINE. Similarly, SANN predictions of the exposed state (Q_E) is also higher than CASA with 6.53% difference, but CASA provides a much higher buried state prediction (Q_B) with 11.78% difference and the overall Q_2 predictions with ~3% improvement over SANN.

6.1.3 Discussions

The context-based scores are effective features to enhance the neural network training process. When context-based scores are incorporated, the solvent accessibility prediction accuracy is improved on all three benchmarks compared to those using PSSM data only.

Figure 33 depicts an example of solvent accessibility prediction on protein 3NRF, chain 'A' listed in CASP9 targets. The first row, underneath the native structure in Figure 33, is the amino acid sequence, the second row is the DSSP assignments of each residue, the third row is the predicted solvent accessibility state when using PSSM information for encoding, and the last row is the prediction when incorporating context based scores with PSSM information. An improvement of 6.61% is achieved in this prediction example upon the incorporation of context based scores with PSSM information.

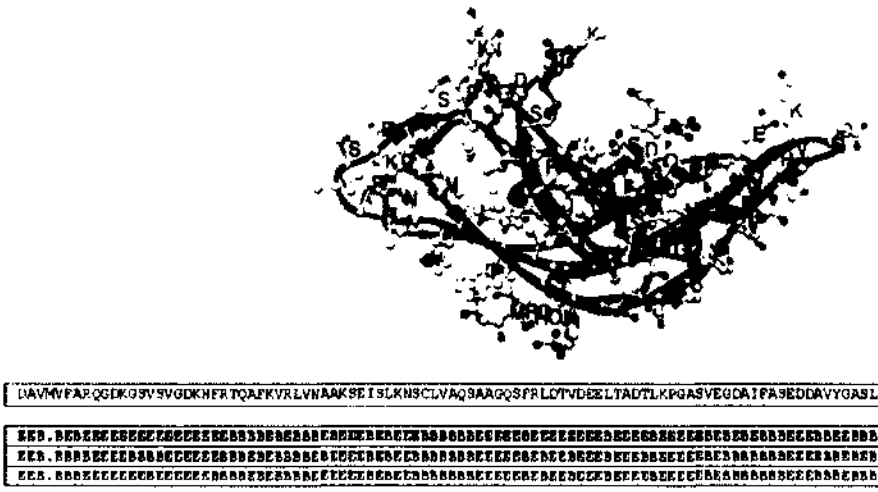


Figure 33 Solvent Accessibility Prediction on protein 3NRF(A).

Although the overall improvement of our method is relatively small, from tertiary structure prediction point of view, reducing inaccuracy, even just a few percent, would be very helpful in modeling efficiency, because the search space for finding a tertiary structure goes up superlinearly with the fraction of inaccuracy. Furthermore, our method of generating context-based statistics relies on the number of known protein structures in PDB. As the number of protein structures available in PDB continues to increase rapidly, we will be able to obtain more accurate context-based statistics for solvent accessibility. Hence, our method has potential to achieve further accuracy improvement in the future.

CHAPTER 7

GPU-ACCELERATION OF MANY-BODY POTENTIALS

In this chapter, we present approaches to accelerate the calculations of pairwise and high-order interactions by taking advantage of the emerging high performance computing architectures in GPU. In section 7.1, we use a 2-body knowledge-based energy function, DFIRE, as an example to illustrate our GPU implementation and to show the efficiency of the proposed approach in accelerating pairwise interaction calculations. In section 7.2, we use two potential energy functions with 3-body terms, including the Axillord-Teller Potential and the Context-based Secondary Structure Potential (CSSP) as examples to demonstrate the effectiveness of our approach in accelerating 3-body interactions.

The main contribution in this work is the design of workload distribution schemes to achieve perfect or nearly perfect load balancing among GPU threads in the symmetric 2-body and 3-body problems. Moreover, the evaluation of DFIRE, in particular, exhibits a few floating-point operations but intensive memory accesses instead. Accordingly, we reorder the protein atom sequence by types to improve cache efficiency in latest NVIDIA Fermi GPU architecture.

Other standard CUDA programming techniques are implemented in order to fully take advantage of the power of the GPU architecture. Examples include coalesced memory access, fine-grain threads, parallel sum reduction, loop unrolling, and taking advantage of GPU memory hierarchy.

7.1 Accelerating 2-body Potentials

Reducing the energy evaluation time is the key to accelerate many modeling and simulation programs. The major part of most energy evaluation involves estimating interactions between pair-wise atoms, which is typically an *N*-body problem. For a system involving *N* particles, conventional evaluation of the potential energy function by estimating every pair-wise interactions requires $O(N^2)$ operations. As a result, for

relatively large systems, calculation of all pairs of interactions requires substantial computational time.

In this section, we present an approach to accelerate the calculations of knowledge-based energy functions popularly used in computational protein structure modeling by taking advantage of the GPU architecture. We use an all-atom knowledge-based energy function, DFIRE [26], as an example to illustrate our GPU implementation. Our key contribution is the design of a workload distribution scheme to achieve perfect or nearly perfect load balancing among GPU threads in the symmetric *N*-body problem. Moreover, unlike many *N*-body simulations [80-82, 84] where a large number of floating point operations are involved, the evaluation of knowledge-based energy functions exhibits a different computing pattern with few floating-point operations but intensive memory accesses instead. Accordingly, we reorder the protein atom sequence by types to improve cache efficiency in latest NVIDIA Fermi GPU architecture. We name the GPU implementation of DFIRE energy function "GPU–DFIRE" while the original serial CPU version is referred to as "CPU–DFIRE". GPU–DFIRE is implemented on the recent Fermi architecture using CUDA programming environment [109]. A Monte Carlo sampling program and a local optimization program are used to demonstrate the efficiency of GPU–DFIRE in all-atom protein structure modeling.

7.1.1 GPU-DFIRE Implementation Details

We use DFIRE [110] potential energy function as an example to illustrate our GPU implementation of memory intensive knowledge-based energy functions. The GPU implementation of DFIRE (GPU–DFIRE) can be adopted into a variety of protein modeling algorithms, such as Monte Carlo (MC) methods [111], Local Energy Minimization [112], Molecular Dynamics [113], Genetic Algorithms (GA) [96], Evolutionary Computing (EC) [114], etc., where repeatedly assessing the potential energy of protein conformations is required.

7.1.1.1 DFIRE Potential

DFIRE is an all-atom potential energy function derived from "the structures of single-chain proteins by using a physical state of uniformly distributed points in finite spheres as the zero interaction reference state" [26]. A large three-dimensional DFIRE

array (first atom type, second atom type, and distance bin) is used to store the statistical potential energy values of each possible atom pair based on their Euclidean distance. In this work, we consider the symmetric version of DFIRE, which has demonstrated better accuracy than the asymmetric one [26]. In symmetric DFIRE, for a protein sequence starting from N-terminal to C-terminal, DFIRE(ATOMS[i], ATOMS[j], d) = DFIRE(ATOMS[j], ATOMS[i], d), for atom pair i and j in distance d and ATOMS[i] denotes the atom information of the i^{th} atom in the protein. The DFIRE program takes the protein PDB file as input and parses it into the ATOMS array, including atom type in DFIRE, sequence number, and spatial coordinate values (XYZ).

The computation of DFIRE energy is a near *N*-body calculation. Starting from the first atom in the ATOMS array, for every atom in the protein, the DFIRE program retrieves the energy terms between the current atom and the rest of the atoms not in the same residue. The energy term is obtained by calculating the pair-wise atom distance, converting it into a distance bin, and then looking up the large DFIRE array for the appropriate energy term value. DFIRE calculation has a distance cutoff; if the distance between two atoms is bigger than the cutoff, the interaction between these two atoms is deemed to be small enough to be ignored. Figure 34 shows the pseudocode of DFIRE subroutine (calcDFIRE) for pair-wise atom interaction calculation. Finally, all energy terms are accumulated to generate the overall DFIRE potential energy value. The major operations in calculating DFIRE energy are memory accesses, i.e., looking up the large DFIRE array for every atom pair.

```
calcDFIRE(atomi, atomj, d)
{
    if(d < CUTOFF)
        return (DFIRE[atomi.DFIREtype, atomj.DFIREtype, d] );
    else
        return 0.0;
}
```

Figure 34 DFIRE subroutine for pair-wise atom interaction calculation.

The serial calculation of symmetric *N*-body interaction in DFIRE is straightforward, whose pseudo code is shown in Figure 35.

7.1.1.2 Workload Distribution

Unlike the asymmetric *N*-body problem, where exactly (*N*-1) 2-body interactions are calculated for each particle, in the symmetric *N*-body problem the number of 2-body interaction calculations in the inner loop varies gradually from *N*-1 to 1. Consequently, directly mapping the calculations in the inner loops to GPU threads will lead to unbalanced workload distribution.

To balance workload distribution among GPU threads, we design the following novel load assignment scheme. The pseudocode of workload assignment for a thread in GPU-DFIRE is shown in Figure 36.

```
score = 0.0;                                    // initialization
for i=1 to N-1 {                                // outer loop
    for j=i+1 to N {                                // inner loop
        d = dist(ATOMS[i], ATOMS[j]);           //distance bin btw. atoms i, j
        score = score + calcDFIRE(ATOMS[i], ATOMS[j], d);
    }
}
```

Figure 35 Pseudocode of symmetric N-body calculation in serial DFIRE.

```
Thread(i)
{
    score = 0.0;                                // initialization
    if(N mod 2 == 0 AND i > N/2)
        count = N/2 - 1;                        // even number and second half
    else
        count = N/2;                            // odd number or even number/first half

    for k=1 to count {                          // atom-atom calcualtion loop
        j = (i + k) mod N;                      // next atom number
        d = dist(ATOMS[i], ATOMS[j]);           // distance bin btw. atoms i, j
        score = score + calcDFIRE(ATOMS[i], ATOMS[j], d);
    }
}
```

Figure 36 Pseudocode of workload assignment in GPU-DFIRE.

For simplicity in illustration, we assume the one-thread-per-atom assignment. Each thread i carries out M atom-atom interaction calculations between atom i and atom $(i+1)$ mod L, $(i+2)$ mod L, . . . , $(i+M)$ mod L. If L is an odd number, M is $(L-1)/2$ and perfect load balancing in doublets calculations can be obtained. On the other hand, when L is an even number, for the first $L/2$ threads, M is $L/2$ and for the second $L/2$ threads, M is $L/2-1$. Figure 37 shows the thread load distribution in 2-body calculations for an odd N, where perfect load balancing can be achieved. Whereas, Figure 38 shows the thread load distribution for an even N, where nearly perfect load balancing can be achieved-the first half of the threads carry out one more atom-atom interaction calculation each than the second half of the threads.

7.1.1.3 Cache Efficiency

In the latest NVIDIA GPU Fermi architecture, the device memory is cached with L1/L2 cache. L1 cache is shared in one multiprocessor while L2 cache is shared among all multiprocessors. Correctly organizing data can take advantage of the cache coherence in GPU. Generally, atoms in a PDB file are grouped by their residues in the protein.

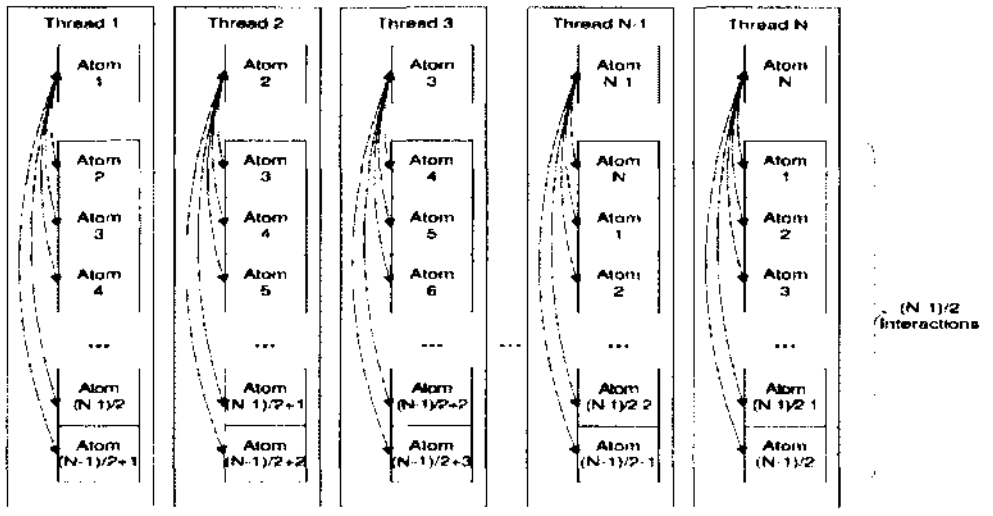


Figure 37 Perfect balancing when N is an odd number. Each thread carries out (N – 1)/2 atom–atom iterations.

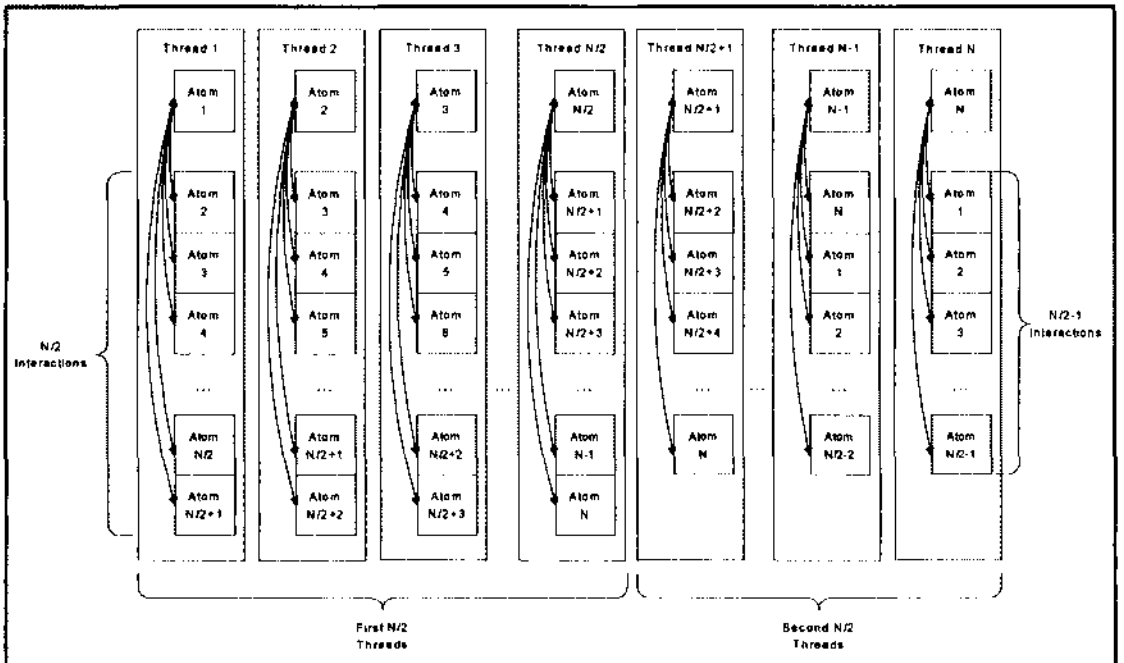


Figure 38 Nearly perfect balancing when N is an even number. The first N/2 threads carry out N/2 atom-atom interactions. The second N/2 threads carry out N/2-1 interactions.

Figure 39 shows the sorting of a 6-residue fragment of protein, where atoms of the same types are clustered after reordering. Clustering atoms of the same types together can potentially improve the cache hit rate. Another advantage of reordering the atom sequence by atom types is, in case of cache misses, the requested global memory addresses will have a good chance to fall within fewer cache-lines compared to unsorted atom sequence, which can lead to better bus utilization.

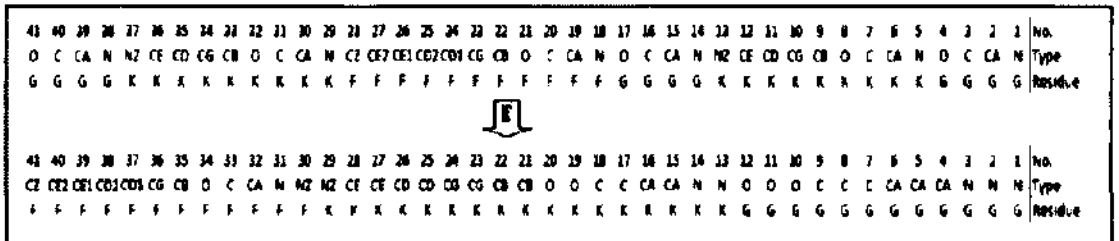


Figure 39 Reordering atoms in a 6-residue protein fragment according to atom types. Atoms of the same type are clustered after sorting.

Table 25 compares the GPU-DFIRE performance using sorted and unsorted atom sequences. The performance data is obtained by NVIDIA Compute Visual Profiler 3.2 [115] on an NVIDIA M2050 (Fermi architecture). One can find that when the atom sequences are sorted by the atom types, the number of L1 hits increases while that of L1 misses decreases, which indicates L1 cache efficiency improvement. However, surprisingly, we only observe GPU time decreasing in relatively small proteins with less than 500 residues; for proteins over 500 residues, the GPU times with sorted atom sequence are even higher than those without sorting.

Our further analysis finds that this is mainly caused by the divergent branches in the GPU-DFIRE program. The DFIRE energy evaluation requires testing the atom-atom distance against the DFIRE cutoff—if the atom–atom distance is higher than the DFIRE cutoff, the atom–atom interaction will be ignored. When the threads handling atom–atom interactions within the DFIRE cutoff as well as those exceeding cutoff co-reside in the same warp, divergent branches may occur in runtime. As shown in table 25, the number of divergent branches in GPU-DFIRE with sorted atom sequence grows significantly as the protein size increases.

Table 25 Performance between sorted and unsorted atom sequences in proteins of various sizes. Performance data are obtained by NVIDIA Compute Visual Profiler 3.2 [115].

PDB	#of res	#of atoms	GPU time			L1 hits		L1 misses		Divergent branches	
			Sorted (μs)	Unsorted (μs)	Sorted/unsorted	Sorted	Unsorted	Sorted	Unsorted	Sorted	Unsorted
1PRB	53	419	49	76	0.65	4,687	3,069	2,305	3,493	160	107
1G6U	96	718	56	85	0.66	8,289	6,007	2,833	4,575	623	291
1DFR	162	1,294	158	242	0.65	17,306	14,581	9,451	12,285	1,962	977
2WJK	210	1,631	220	360	0.61	36,743	24,340	15,980	22,859	1,982	1,018
1YQS	345	2,654	479	729	0.66	111,469	56,012	20,477	27,621	5,226	3,058
1OWH	455	3,619	893	1,011	0.88	139,518	118,291	35,757	55,947	15,247	5,322
2XGY	597	4,669	1,429	1,416	1.01	220,512	184,191	28,154	48,821	24,133	3,603
2YB9	696	5,595	1,851	1,827	1.01	339,468	301,044	63,379	67,121	24,579	2,147
3NLJ	818	6,642	1,056	2,686	1.13	563,190	422,003	87,131	99,743	27,487	5,432
3QDG	1 068	7,992	3,729	3,514	1.06	710,890	613,904	85,466	248,178	42,726	7,608
3NKY	1 124	8,788	3,985	3,548	1.12	758,835	691,194	79,113	96,515	59,320	8,377
3QHR	1 156	9,284	5,737	4,887	1.17	826,776	729,634	76,225	127,422	67,655	8,110
3NUK	1,314	10,787	7,529	7,131	1.06	1,174,630	1,033,720	66,588	234,798	74,980	15,650
3MBC	1,470	11,222	7,901	7,438	1.06	1,197,540	1,071,820	71,483	184,837	72,870	16,131
2XQY	1,796	13,633	9,504	9,317	1.02	1,863,810	1,650,320	120,925	272,308	79,668	11,929
2WPF	1,955	14,888	12,380	11,799	1.05	2,115,280	1,892,400	108,928	277,929	90,850	14,063
3LUO	3,258	25,407	34,819	33,116	1.05	7,464,650	3,849,770	354,802	381,078	188,759	21,997
2WAQ	3,334	26,454	36,419	34,802	1.05	6,885,780	3,839,500	364,048	666,217	179,784	33,541
3E0O	4,596	34,623	51,778	50,158	1.03	11,196,500	9,431,000	588,343	1,076,800	278,080	38,483
1GYT	6,036	46,152	93,899	91,680	1.02	19,160,200	17,412,200	523,193	1,382,620	364,695	62,972
2PMZ	6,058	48,098	98,554	94,963	1.04	21,747,500	18,818,900	594,047	1,370,870	400,250	50,119
3E3N	6,482	52,706	114,301	111,298	1.03	25,535,300	21,516,200	564,874	1,275,850	367,785	69,461
1UF2	7,434	58,130	151,318	146,181	1.04	31,772,400	27,670,000	743,424	1,647,330	531,273	78,664
3H0C	7,904	62,852	170,996	166,339	1.03	36,193,800	32,331,200	830,885	1,261,420	485,525	74,027
3MQT	9,063	71,547	204,809	200,529	1.02	45,718,400	41,279,800	989,423	2,702,170	642,562	99,490
3KIF	10,400	83,080	283,510	275,717	1.03	63,210,500	57,158,800	1,529,580	2,757,380	744,937	107,584
3HF9	11,907	90,054	342,698	332,235	1.03	72,799,100	70,040,100	1,100,350	2,448,980	696,403	102,063

In small proteins with less than 500 residues, the numbers of divergent branches in GPU-DFIRE with sorted atom sequences are 1.5–3 times of those without sorting. In contrast, in larger proteins with more than 500 residues, the ratio of divergent branches with sorted and unsorted atom sequences increases to 4–8.

This is due to the fact that sorting atom sequences according to atom types increases the chance of divergent branches occurrences, because the clustered together atoms of the same type may be from nearby or faraway residues. In larger protein molecules, the occurrence chance of divergent branch becomes higher after sorting. Consequently, in large proteins, the GPU time gains of the cache efficiency by sorting are counteracted by the increasing number of divergent branches. Nevertheless, in practice, the DFIRE energy function is often used in protein modeling applications on small proteins typically with less than 300 residues. Sorting atom sequences according to atom types are effective in GPU-DFIRE evaluation in small proteins with less than 500 residues due to improved cache efficiency, where the overall GPU times are reduced 11%–45%.

7.1.1.4 Additional Improvements

In addition to asymmetric *N*-body load balancing and reordering atom sequence according to atom types in GPU-DFIRE we also use the following "standard" CUDA programming techniques in our GPU implementations to fully take advantage of the power of the GPU architecture.

(1) Coalesced global memory access

In GPU architecture, global memory access by all threads in the half-warp (non-Fermi) or full-warp (Fermi) of a block can be coalesced into efficient memory transactions [109]. In GPU-DFIRE, we reorganize the data arrays used in the DFIRE program to facilitate coalesced memory access.

For example, instead of viewing atoms information as an array of structures, we reconstruct the ATOMS array from "array of structure" to "structure of array". The array reconstruction posts a one-time cost at startup, which has trivial impact to the overall application performance, particularly when the DFIRE potential energy is evaluated many times for different protein conformations. Moreover, threads per block are chosen to be a multiple of warp size in our GPU implementation.

(2) Fine-grained threads

For an N-body problem with small N, achieving good performance on GPU is difficult, as shown in the literature [80]. In GPU-DFIRE, similar to the technique used in [80], we adopt fine grained threads by increasing the number of active threads by assigning multiple threads to compute interactions of an atom in a small protein. We firstly calculate the number of threads (T_n) that can be assigned to handle interactions computation of an atom by dividing the maximum number of threads (T_{max}) that a GPU device can launch over the total number of atoms N. ($T_n = T_{max}/N$.)

Then, we distribute the workload of interaction computation to T_n threads. As a result, large number of threads whose total number is near the maximum number of threads that the GPU hardware can launch are created. With sufficient number of threads, the memory access latency can be effectively masked. Figure 40 show the effectiveness of fine-grained threads in GPU-DFIRE in small proteins.

Our computational results of GPU-DFIRE on Tesla M2050 show that the turning point between linear increasing speedup and constant speedup is when N = 2000, as shown in Figure 40. Fine-grained threads are particularly effective in proteins with less than 2000 atoms.

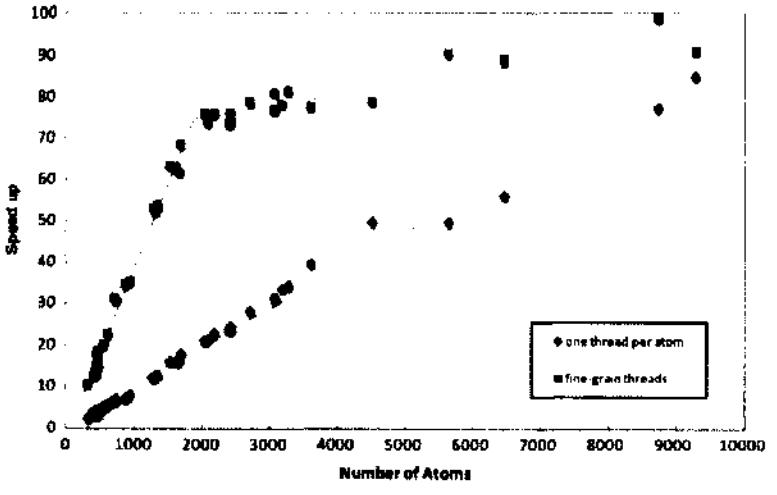


Figure 40 Effectiveness of GPU-DFIRE with fine-grained threads in small proteins (Tesla M2050).

It is important to notice that particularly more significant performance improvements are found in the small proteins, whose speedups are promoted to be close to those of the large proteins. This is due to the fact that sufficiently large number of threads is produced to use the GPU hardware to its fullest. As the number of particles increases, the speedup curves of fine-grain threads and one-thread-per-particle start to merge because the increasing number of threads in one-thread-per-particle strategy makes more efficient use of the GPU architecture.

(3) Parallel sum reduction

We adopt the parallel sum reduction algorithm [116], a tree based approach, to compute the overall energy from the partial energy sums generated by the GPU threads in GPU-DFIRE. The algorithm involves a local sum reduction step and a global sum reduction step. In the local sum reduction step, the tree based approach [116] is used to accumulate the partial sums from threads within each block. Then, an additional kernel is launched to add up the partial sums from each block in the previous kernel using the tree-based approach again.

Since fined-grained threads are used for small proteins, the computation time of parallel sum reduction is nearly constant for proteins of various sizes, which is approximately 0.1 *ms* on Tesla M2050. This is less than the computation time of simply using CPU to sum up the partial sums, which ranges from 0.14 *ms* (proteins with ~300 atoms) to 3.0 *ms* (proteins with ~10,000 atoms). After all, compare to the overall GPU–DFIRE energy evaluation time, even for the small proteins which typically takes 1.0–3.0 *ms*, the parallel sum reduction time is relatively small.

(4) Take advantage of GPU memory hierarchy

Our GPU–DFIRE implementation requires transferring the DFIRE table and atom information arrays (ATOMS) including atom type array, residue number array, and atom coordinates arrays, from the host memory to the GPU device memory and retrieving the calculated overall energy from the device memory.

We reorganize the data arrays in DFIRE program to different memory locations in the GPU. First of all, we take advantage of the shared memory to reduce the number of accesses to the global memory. In GPU-DFIRE, all threads in a block shares the ATOMS array; hence, the data in the ATOMS array for a thread block can be loaded in the shared

memory. Unfortunately, for proteins with large number of atoms, the high-speed shared memory has limited size, which may not be able to accommodate all data in the input array. We have to break the input array data into tiles, where a tile represents a fixed dimension of the atom information data. Then, the tile is loaded into the shared memory and used by all threads in a block. Once all threads are done with one tile, the next tile will be loaded into the shared memory and override the previous one. This process is repeated until all computations in the thread block are completed. Moreover, the latest NVIDIA Fermi architecture introduces a cache hierarchy for caching local and global memory accesses. For example, Tesla 2050M GPUs have 64 kB of RAM per streaming multiprocessor, which can be partitioned into shared memory and L1 cache. A CUDA programmer can select combinations of "48 kB shared memory +16 kB L1 cache" or "16 kB shared memory +48 kB L1 cache" according to his/her program needs. Our GPU-DFIRE implementation on Tesla M2050 uses the "48 kB shared memory +16 kB L1 cache" combination, which yields slightly better performance than the other combination. Figure 41 shows the overall implementation of GPU–DFIRE on GPU memory hierarchy (Fermi).

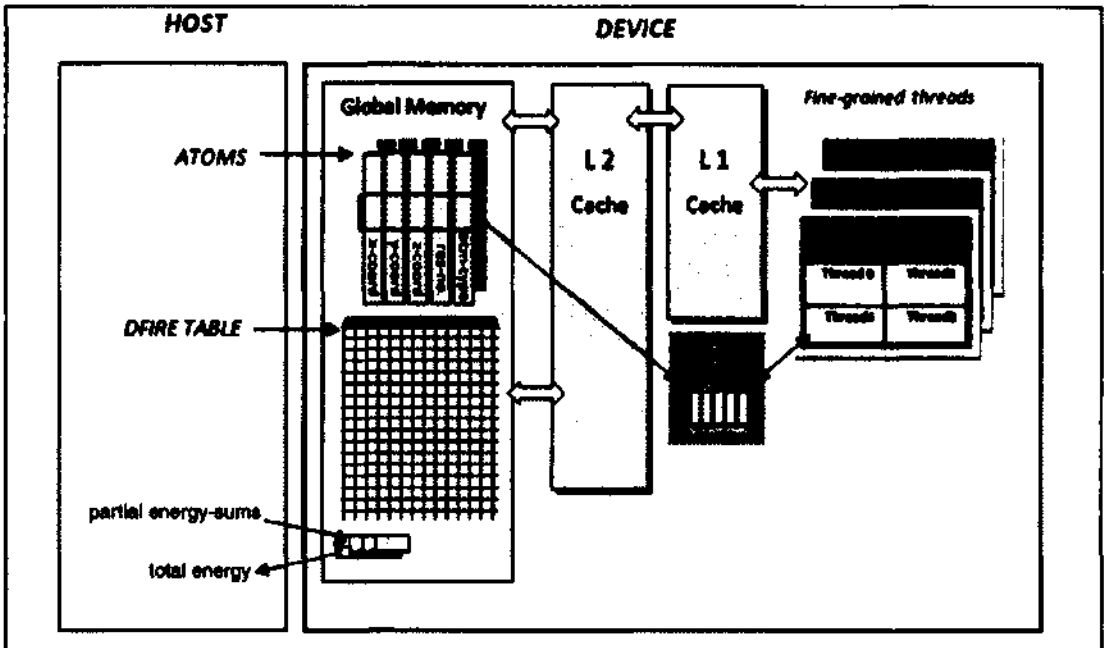


Figure 41 Implementation of GPU–DFIRE on GPU memory hierarchy (Fermi architecture).

(5) Loops unrolling

Loop unrolling is an optimization technique that can be applied to GPU programming by replacing the body of a loop with multiple copies of itself [80]. For example, in GPU-DFIRE we unroll the atom-atom calculation loop in the pseudocode shown in Figure 36. Figure 42 compares the GPU-DFIRE performance of 2 times loop unrolling and 4 times loop unrolling by showing their average percentages relative to the computation time of GPU-DFIRE without loop unrolling. One can find that replicating the loop body for 2 times yields best performance in GPU-DFIRE. Loop unrolling is more effective in large proteins with averagely ~4% computation time reduction because they contain more loop iterations.

7.1.2 Computational Results

The GPU–DFIRE programs are tested on a server with Quadro FX3800M as well as a server with Tesla M2050 GPU. The Tesla M2050 GPU (Fermi architecture) has 14 multiprocessors with 32 cores each, 3 G of global memory, 64 kB of RAM which can be configured between Shared Memory and L1 cache and 32 kB of registers per multiprocessor.

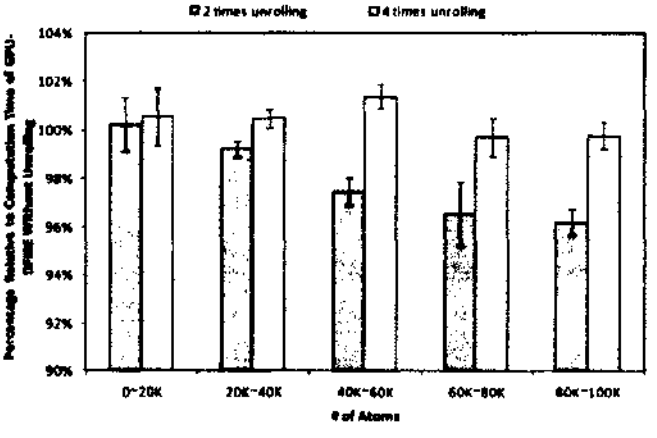


Figure 42 Effect of loop unrolling in GPU–DFIRE performance (Tesla M2050) on a set of 84 proteins ranges from 318 atoms to 99,595 atoms. The computation time measures are based on the average of 10 runs.

The Quadro FX3800M GPU (non-Fermi architecture) has 16 multiprocessors with 8 cores each, 1 G of global memory and 16 K of RAM. The CPU version of DFIRE (CPU–DFIRE) runs on a server with an Intel i7 CPU 920 @ 2.67 GHz, 8 MB cache, and 6 G memory. We firstly benchmark GPU–DFIRE on a set of proteins of various sizes. The GPU time we measured includes the time of transferring the protein information (ATOMS) arrays to GPU device memories, GPU execution time, and the time of retrieving the calculated overall DFIRE energy from GPU. Then, we apply GPU–DIRE to a Monte Carlo program for protein conformation sampling and a program for protein energy local minimization. We use the gcc compiler with the default "-O3" optimization flag specified in DFIRE package for CPU–DFIRE. For GPU–DFIRE, nvcc compiler in CUDA 2.0 is used with "-O3" flag.

7.1.2.1 Overall Speedup

Figure 43 shows the overall speedup of GPU-DFIRE using pairwise interaction on NVIDIA Tesla M2050 and Quadro FX3800M on proteins of various sizes with respect to CPU-DFIRE.

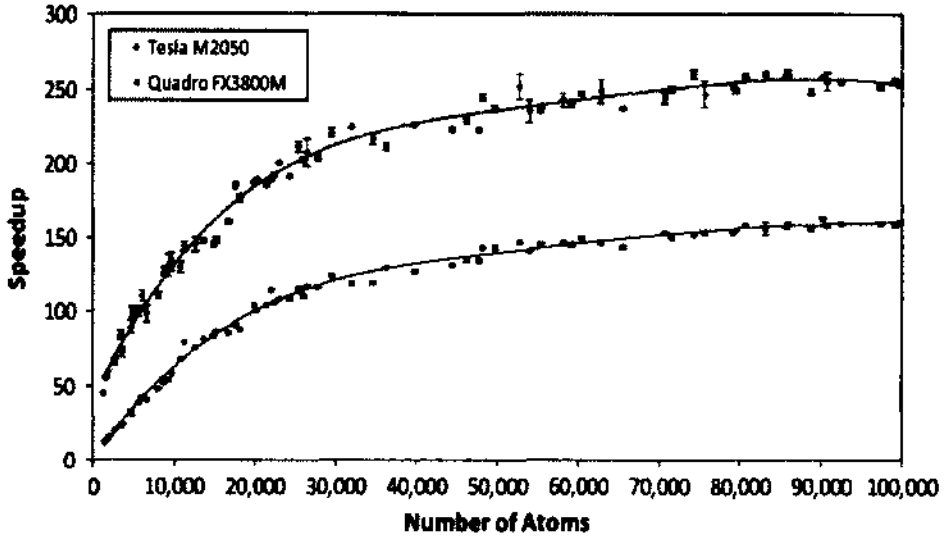


Figure 43 Overall speedup of GPU–DFIRE on Tesla M2050 and Quadro FX3800M on a set of proteins with various sizes with respect to CPU–DFIRE. The computation time measures are based on the average of 10 runs for each protein.

Due to not having enough computations in each thread, there are certain inefficiencies in GPU-DFIRE for proteins with less than 20,000 atoms. However, the performance is consistently high for proteins with more than 20,000 atoms. For very large proteins with more than 50,000 atoms, the maximum speedups of GPU-DFIRE can converge to ~150 and ~250 using Quadro FX3800M and Tesla M2050, respectively. Moreover, it is important to notice that Tesla M2050 yields significant higher speedups than Quadro FX3800M in small proteins with less than 5000 atoms. This is due to the fact that Quadro FX3800M is non-Fermi architecture without L1cache, where the strategy of sorting atom sequence by atom types to achieve cache efficiency cannot take effect.

7.1.2.2 Applications of GPU-DFIRE in Protein Structure Modeling

GPU-DFIRE can be adapted to a variety of protein structure modeling algorithms requiring assessing protein molecule energy or feasibility. We use GPU-DFIRE in a Monte Carlo protein conformation sampling program and a local structure optimization program (MINIROT) where DFIRE is the target energy function. The measured computation time is the application execution time, which includes CPU time, GPU time, and the data transferring time between host memory and device memory.

In this work, we only consider how the acceleration in energy function evaluation using DFIRE can affect the overall performance of the protein structure modeling program. However, it is important to notice that if more parallel computations, e.g., randomly proposing new atom positions in Monte Carlo algorithm and the matrix operations in MINIROT, are moved to the GPU, more aggressive performance improvements may be obtained. After all, the key advantage of using GPU-DFIRE is that almost no programming modification of the original protein structure modeling program is necessary since GPU-DFIRE can provide the same programming interface as CPU-DFIRE.

- *Monte Carlo sampling*

We use GPU-DFIRE in a Monte Carlo sampling program provided by the TINKER package [117]. DFIRE is used as the target potential energy function. The protein conformational search is carried out by using Cartesian all atoms move where the position of every atom in the protein molecule is changed by a small random perturbation

during every Monte Carlo trial. The Monte Carlo sampling program uses the Metropolis algorithm [112] and the DFIRE energy is evaluated in every iteration step to determine the acceptance of the proposed new conformation.

Adopting GPU-DFIRE, the Monte Carlo sampling algorithm is implemented as a heterogeneous CPU-GPU program. The evaluation of the DFIRE energy is carried out on the GPU while the rest of the Monte Carlo computations are executed on the CPU. We carry out the Monte Carlo optimization program using GPU-DFIRE on Tesla M2050 on a protein 3GDG with 7992 atoms. Our computational results show that the number of Monte Carlo steps per second is increased from 2.57 in CPU-DFIRE to 212.67 in GPU-DFIRE. The acceleration in the energy function evaluation significantly improve the performance of the Monte Carlo computation with an average speedup of 82.67, where the original computation time for 105 iterations is reduced from more than an hour to less than one minute.

The Cartesian all atom move in this Monte Carlo example requires evaluation of interactions between every atom pair. However, for Monte Carlo methods employing local conformational moves by changing a few torsion angles or positions of small number of atoms in a Monte Carlo trial, energy re-evaluations are only necessary for the atom pairs with relative position changes and thereby the computation times of both CPU-DFIRE and GPU-DFIRE may be further reduced.

- *Local structure optimization*

We adopt GPU-DFIRE in the MINIROT program [118] provided by the TINKER package [117] where DFIRE is the target energy function. The MINIROT program performs local energy minimization of an initial protein structure over dihedral angle space using a limited memory Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. DFIRE energy is evaluated at every iteration step to determine the descending gradient until convergence is reached. Similar to the implementation of the Monte Carlo program, the evaluation of the DFIRE energy is on the GPU and the rest computations are on CPU. Because evaluation of the descending gradient on each atom is needed in MINIROT, we use the asymmetric scheme instead of the symmetric scheme in GPU-DFIRE.

Table 26 compares the percentages of energy function evaluation times in the Monte Carlo sampling program and the MINIROT program using the CPU-only implementation and the heterogeneous CPU–GPU implementation with energy calculations on GPU. Compared to the Monte Carlo program where 99.9% of its computation time is in energy evaluation, the MINIROT program has significant more computations on CPU due to its matrix operations in linear search. Therefore, the energy evaluation occupies 75.8% of the overall computation time in the MINIROT program, which is still in majority but is much less than that of the Monte Carlo sampling program.

Table 27 shows the performance of the MINIROT program on 6 initial structures of protein 2ERL using GPU-DFIRE and CPU-DFIRE. Using GPU-DFIRE by migrating the energy evaluation computation to the GPU, the percentage of computational time spent in energy evaluation in the overall MINIROT program is reduced from 75.8% to 24.1%. As a result, although not as significant as that of the Monte Carlo program using GPU-DFIRE, MINIROT with GPU-DFIRE can achieve an average speedup of ~4.5.

Table 26 Percentages of energy evaluation times in Monte Carlo sampling program and MINIROT using the CPU-only implementations and the heterogeneous CPU–GPU implementations with energy function evaluation on GPU. The energy evaluation times in heterogeneous CPU.

	Percentage of energy evaluation time (CPU-only implementation) (%)	Percentage of energy evaluation time (heterogeneous CPU-GPU implementation with energy evaluation on GPU) (%)
MC	99.9	89.2
MINIROT	75.8	24.1

Table 27 Performance comparison of MINIROT program on 6 initial structures of 2ERL (319 atoms) using GPU–DFIRE and CPU–DFIRE. GPU–DFIRE is carried out on Tesla M2050 server.

	RMSD (A)	Num. of steps	Execution time (s)	
			CPU-DFIRE	GPU-DFIRE (Tesla M2050)
Initial 1	3.19	338	64.76	14.19
Initial 2	2.06	225	43.39	9.73
Initial 3	2.24	480	91.70	19.89
Initial 4	3.19	222	43.23	10.02
Initial 5	2.85	692	130.35	26.81
Initial 6	3.62	434	82.21	17.27

7.2 Accelerating 3-body Potential

Three-body effects play an important role for obtaining quantitatively high accuracy in a variety of molecular simulation applications. However, evaluation of three-body potentials is computationally costly, generally of $O(N^3)$ where N is the number of particles in a system. Although the N-body interactions can be carried out in a straightforward way on a serial processor, efficient parallel implementation to fully take advantage of GPU architectures requires deliberate considerations.

In this section, we extend our previous GPU-based 2-body load-balancing workload distribution scheme, presented in section 7.1, to explicitly calculating 3-body terms. A load-balancing workload distribution scheme is presented for calculating 3-body interactions by taking advantage of the GPU architectures. Perfect load-balancing is achieved if N is not divisible by 3 and nearly perfect load-balancing is obtained if N is divisible by 3. The workload distribution scheme is particularly suitable for the GPU's Single Instruction Multiple Threads (SIMT) architecture, where particles data access by threads can be coalesced into efficient memory transactions. We use two potential energy functions with 3-body terms, including the Axillord-Teller Potential [119] and the Context-based Secondary Structure Potential (CSSP) [94] as examples to demonstrate the effectiveness of our workload distribution scheme.

7.2.1 3-body Effects

Although many molecular simulations are typically confined to evaluating interactions between molecular pairs, recent studies show that three-body or even higher order effects play an important role for quantitatively accurate computation in a variety of molecular simulation applications [120-123]. For example, the three-body effects strongly influence solid-liquid and vapor-liquid equilibria of fluids [124-127]. The context-based secondary structure potential taking three-body statistical terms into account leads to significant accuracy enhancement in evaluating protein secondary structures [94]. A three-body potential incorporating interaction between a DNA base and a protein residue with regard to the effect of a neighboring DNA base outperforms two-body potentials in specific protein-DNA site recognition [128]. A four-body residue-

residue contact potential has demonstrated its effectiveness compared with pairwise potentials in discriminating native protein conformations [129]. Inclusion of three-body effects in the additive CHARMM protein CMAP potential also results in enhanced cooperativity of α-helix and β-hairpin formation [130].

Despite the advantages of calculating three-body interactions explicitly in molecular simulations, the main obstacle of a potential energy involving three-body or higher order terms is its high computational cost. In general, when external influences are not presented, the potential energy of a system with N particles can be evaluated as

$$E = \sum_{i \neq j} U(p_i, p_j) + \sum_{i \neq j \neq k} U(p_i, p_j, p_k) + \sum_{i \neq j \neq k \neq l} U(p_i, p_j, p_k, p_l) + \cdots,$$

where $U(p_i, p_j)$ is the two-body term involving two particles p_i and p_j, $U(p_i, p_j, p_k)$ is the three-body term, and $U(p_i, p_j, p_k, p_l)$ is the four-body term, and so on. Computing time increases largely when higher order terms are included. Generally, two-body terms require $O(N^2)$ operations, while $O(N^3)$ for three-body terms, $O(N^4)$ for four-body terms, and so forth. Computation reduction approaches such as Barnes-Hut method, fast multipole, and particle-mesh [131-134] can significantly reduce the overall computation complexity by simplifying interactions between far apart particles. Nevertheless, simulation using computation involving three- or higher-body terms is still unrealistic for a system with relatively large number of interacting molecules until recent improvements in computer systems.

In this section, we extend our GPU-based two-body load-balancing workload distribution scheme to explicitly calculating three-body terms. The effectiveness of our approach is demonstrated in the computation of the Axillord-Teller potential [119], a physics-based three-body potential function, and the Context-based Secondary Structure Potential (CSSP), a knowledge-based three-body potential energy function to evaluate protein conformation adopting certain secondary structure pattern [94].

7.2.2 GPU-based Load-balancing Scheme for Computing 3-body Interactions

Our load-balancing scheme assumes that the three-body interaction terms are independent of the order of the three particles. In other words, the order permutation of the three particles does not change the potential value. Moreover, for simplicity in

illustration, we assume one-thread-per-particle assignment, i.e., each thread keeps one particle information unchanged and then shift the second or the third particles information at each iteration to obtain all combinations of triplets. A fine-grained assignment other than the one-thread-per-particle assignment to enhance GPU performance on systems with small number of particles is presented in the next section.

7.2.2.1 Serial Implementation

The general implementation of three-body interaction computation in serial is straightforward. All one needs to do is to enumerate all triplet combinations of three different particles and then calculate the three-body energy of the triplet. The corresponding pseudo code is illustrated in Figure 44. For a molecular system with N particles and assuming that each pair of particles are interacting, there are totally $N*(N-1)*(N-2)/6$ triplet computations and each particle is involved in exactly $(N-1)*(N-2)/2$ interaction computations. However, for each particle in the outer loop in Figure 44, its number of three-body interaction calculations in the middle and inner loops varies gradually from $(N-1)*(N-2)/2$ to 1. Consequently, directly mapping the calculations in the middle and inner loops to GPU threads will lead to a highly unbalanced workload distribution.

```
sumE ← 0.0;                          // initialization
for i ← 1 to N-2 {                   // first particle, outer loop
   particle1 = i;
   for j ← i+1 to N-1 {              // second particle, middle loop
      particle2 = j;
      for k ← j+1 to N {             // third particle, inner loop
         particle3 = k;
         sumE ← sumE + TripleE(particle1, particle2, particle3);
      }
   }
}
```

Figure 44 Pseudocode of calculating three-body terms in a system with N particles.

7.2.2.2 Rotational Symmetry

Our implementation of load-balancing workload distribution scheme for three-body interaction computation on GPU is based on the concept of rotational symmetry. Assuming that the N particles in the system are stored in a cyclic array, we use i, j, and k to denote the indices of the three particles in clockwise order and d_{ij}, d_{jk}, and d_{ki} to denote the position separations between particles i and j, j and k, and k and i, respectively. Here d_{ij} is calculated as

$$d_{ij} = \begin{cases} j - i, & i < j \\ j - i + N, & i > j \end{cases}$$

and d_{jk} and d_{ki} are calculated in a similar way. Clearly, we can have the following two properties

1). $i \neq j \neq k$, and

2). $d_{ij} + d_{jk} + d_{ki} = N$.

Then, we study the position separation pattern of

of a triplet (P_i, P_j, P_k). Considering two position separation patterns

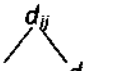
and

,

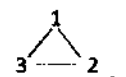
and 

are rotational symmetric if $d_{ij} = d_{ij}'$ and $d_{jk} = d_{jk}'$ and $d_{ki} = d_{ki}'$ or $d_{ij} = d_{jk}'$ and $d_{jk} = d_{ki}'$ and $d_{ki} = d_{ij}'$ or $d_{ij} = d_{ki}'$ and $d_{jk} = d_{ij}'$ and $d_{ki} = d_{jk}'$. Or equivalently, if 

can turn into 

via cyclic rotations, then 

and

are rotational symmetric. Given an example of a system where $N = 6$, 

,

, and

are rotational symmetric but

and

are not. In our GPU implementation, assuming one-thread-per-particle and all threads share the same computation pattern due to GPU's SIMT architecture, rotational symmetric position separation patterns indicate that some threads will calculate certain triplets in overlap, which will lead to waste of computational power and, more seriously, erroneous

results if not handling correctly. Figure 45 illustrates an example where $N = 6$. Thread 1 starts from particle P1 and triplet (P1, P2, P3) has position separation pattern 

. If a rotational symmetric position separation pattern of 

, for instance, 

is adopted, then thread 1 will carry out three-body interaction calculation on triplet (P1, P2, P6), which is the same as the three-body interaction calculation on triplet (P6, P1, P2) in thread 6 starting at P6 with position separation pattern 

. In summary, the fundamental idea of our GPU-based algorithm is to uniquely enumerate all position separation patterns that neither pair is rotational symmetry.

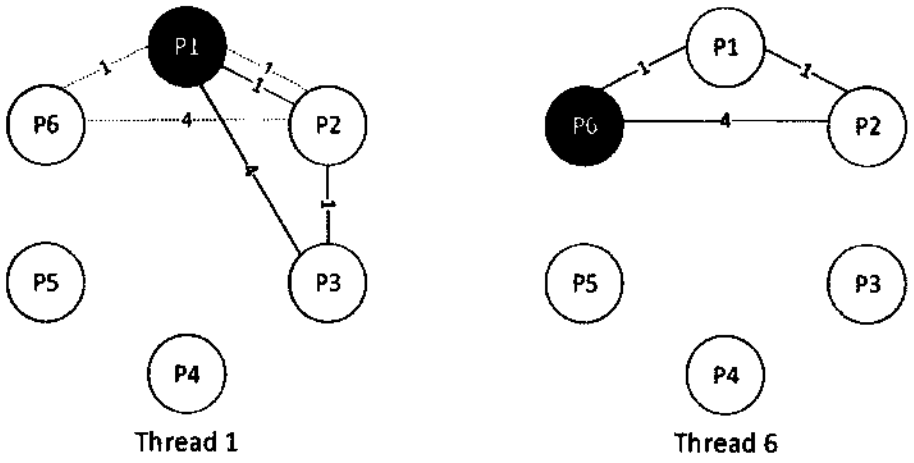
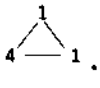


Figure 45 An example with N = 6 with thread 1 starting from P1 and thread 6 from P6. The highlighted particle is the first particle that a thread handles. For separation pattern 

, threads 1 and 6 will calculate three-body interactions of triplets (P1, P2, P3) and (P6, P1, P2), respectively. If thread 1 adopts a position separation pattern 

, which is rotational symmetric to 

, will compute three-body interaction of triplet (P1, P2, P6), which overlaps the computation of thread 6 with position separation pattern 

.

To balance workload distribution among GPU threads, we design a novel workload distribution scheme by enumerating all position separation patterns without sharing rotational symmetry. The workload distribution scheme is shown in Figure 46. For thread i, the index of the first particle is always i specified by the passed parameter and the second and third particles are selected according to the enumerated position separation patterns. The algorithm enumerates all position separation patterns satisfying

$$d_{ij} \leq d_{jk} \text{ and } d_{ij} < d_{ki}. \quad (1)$$

It is easy to show that any position separation patterns that do not satisfy the above condition are rotational symmetric to one of the position separation patterns satisfying this condition, because we can always rotate the smallest position separation to d_{ij}. The above condition also indicates that d_{ij} is bounded by $\lfloor (N-1)/3 \rfloor$. Hence, our algorithm iterates the second particle index from $(i+1) \bmod N$ to $(i + \lfloor (N-1)/3 \rfloor) \bmod N$. Then, the third particle is iterated to satisfy (1).

```
Thread(i, N)
{
   localE ← 0.0;                        // initialization
   particle1 ← i;                       // particle 1
   range ← ⌊(N-1)/3⌋;                   // range of particle 2
   for dij ← 1 to range {
      djk ← dij;
      dki ← N - djk - dij;
      particle2 ← (i + dij) mod N;      // particle 2
      while (dki > dij) {               // range of particle 3
        particle3 ← (i + dij + djk) mod N;      // particle 3
        localE ← localE + TripleE(particle1, particle2, particle3);
        djk ← djk + 1;
        dki ← dki - 1;
      }
   }
   local_sum_reduce(localE);  // sum partial scores in local block
}
```

Figure 46 Pseudocode of enumerating all non-rotational symmetric position separation patterns except for the order three symmetric one and calculating three-body interaction of the corresponding triplet particles in a GPU thread.

Figure 47 illustrates an example of enumerating triplets by the first thread (thread 1) in a system with 10 particles, using the algorithm shown in Figure 46. For thread 1, the first particle is always P1. The second particle iterates from P2 to P4. When the second particle is P2 (d_{ij} = 1), three-body interactions of triplets (P1, P2, P3), (P1, P2, P4), (P1, P2, P5), (P1, P2, P6), (P1, P2, P7), (P1, P2, P8), and (P1, P2, P9) with position separation patterns $\overset{1}{\underset{8}{\triangle}}\,1$, $\overset{1}{\underset{7}{\triangle}}\,2$, $\overset{1}{\underset{6}{\triangle}}\,3$, $\overset{1}{\underset{5}{\triangle}}\,4$, $\overset{1}{\underset{4}{\triangle}}\,5$, $\overset{1}{\underset{3}{\triangle}}\,6$, and $\overset{1}{\underset{2}{\triangle}}\,7$ are calculated, respectively. When the second particle is P3 (d_{ij} = 2), the three-body interactions of triplets (P1, P3, P5), (P1, P3, P6), (P1, P3, P7), (P1, P3, P8) with respective separation patterns $\overset{2}{\underset{6}{\triangle}}\,2$, $\overset{2}{\underset{5}{\triangle}}\,3$, $\overset{2}{\underset{4}{\triangle}}\,4$, and $\overset{2}{\underset{3}{\triangle}}\,5$ are accumulated. When the second particle is iterated to P4 (d_{ij} = 3), only one triplet triplets (P1, P4, P7) with separation pattern $\overset{3}{\underset{4}{\triangle}}\,3$ can satisfy (1). The completion of the algorithm allows thread 1 to carry out three-body interactions of 12 triplets with different position separation patterns that are not rotational symmetric. Assuming that one particle per thread, the total number of three-body interactions is 12*10 = 120 = 10*9*8/6.

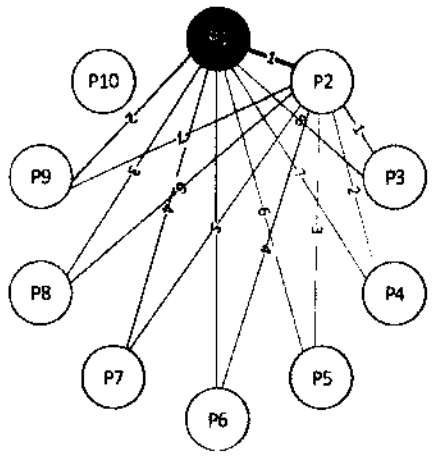


Figure 47 (a) d_{ij} = 1, the second particle is P2 and three-body interactions of triplets (P1, P2, P3), (P1, P2, P4), (P1, P2, P5), (P1, P2, P6), (P1, P2, P7), (P1, P2, P8), (P1, P2, P9) with patterns $\overset{1}{\underset{8}{\triangle}}\,1$, $\overset{1}{\underset{7}{\triangle}}\,2$, $\overset{1}{\underset{6}{\triangle}}\,3$, $\overset{1}{\underset{5}{\triangle}}\,4$, $\overset{1}{\underset{4}{\triangle}}\,5$, $\overset{1}{\underset{3}{\triangle}}\,6$, and $\overset{1}{\underset{2}{\triangle}}\,7$ are calculated, respectively.

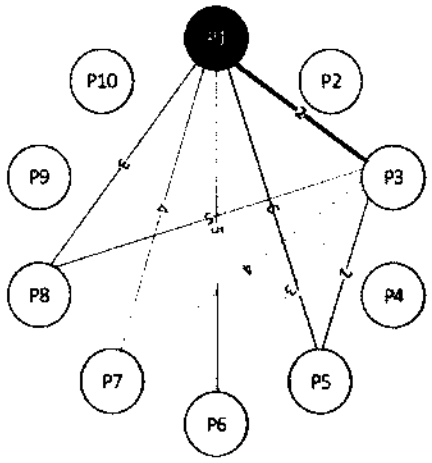


Figure 47 (b) dij = 2, the second particle is P3 and three-body interactions of triplets (P1, P3, P5), (P1, P3, P6), (P1, P3, P7), (P1, P3, P8) with separation patterns $\overset{2}{\triangle}_{6\ 2}$, $\overset{2}{\triangle}_{5\ 3}$, $\overset{2}{\triangle}_{4\ 4}$, **and** $\overset{2}{\triangle}_{3\ 5}$ **are calculated, respectively.**

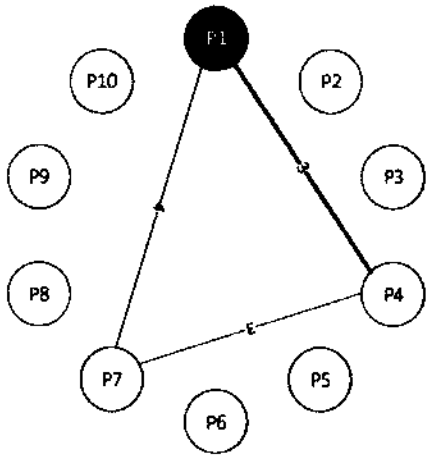


Figure 47 (c) $d_{ij} = 3$**, the second particle and only one three-body interaction of triplet (P1, P4, P7) with separation pattern** $\overset{3}{\triangle}_{4\ 3}$ **is calculated.**

Figure 47 An example of enumerating triplets without sharing rotational symmetry in position separation patterns by the first thread (thread 1) in a system with 10 particles.

The only position separation patterns that the algorithm shown in Figure 46 cannot iterate are order three rotational symmetric patterns in case of $d_{ij} = d_{jk} = d_{kl}$, when N is divisible by 3. The order three rotational symmetric pattern will cause the triplets with equal separation distances to be calculated repeatedly by different threads. Figure 48 shows an example of a system with 6 particles, where an order three rotational symmetric pattern  exists. As a result, threads 1, 3, 5 over calculates triplet (P1, P3, P5) while threads 2, 4, 6 over calculates triplet (P2, P4, P6). The deeper reason is, if N is divisible by 3, $(N - 1)*(N - 2)$ is no longer divisible by 6 and thus the total number of $N*(N - 1)*(N - 2)/6$ interaction computations cannot be equally distributed to N threads. Consequently, order three rotational symmetric patterns require special handling.

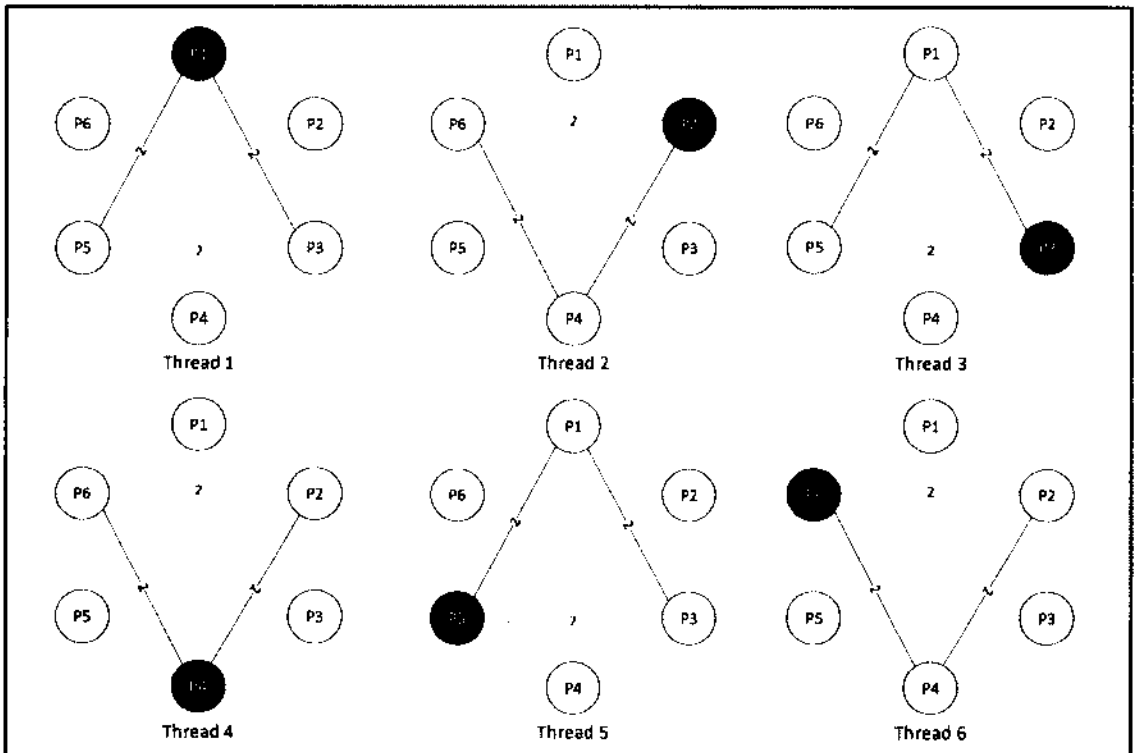


Figure 48 Order three rotational symmetry in a system with 6 particles. The highlighted particle is the first particle a thread handles. Threads 1, 3, 5 over calculate triplet (P1, P3, P5) while threads 2, 4, 6 over calculate triplet (P2, P4, P6).

7.2.2.3 Load-balancing Workload Distribution Scheme

Figure 49 shows the complete workload distribution algorithm with special handling of N divisible by 3 based on the pseudocode provided in Figure 46. Only the first $N/3$ threads will carry out the three-body interaction computation of triplets with order three rotational symmetric position separation pattern to avoid over calculation. When N is not divisible by 3, each thread carries out exactly $(N-1)*(N-2)/6$ three-body interaction operations, where perfect load-balancing is achieved. When N is divisible by 3, the first $N/3$ threads carry out an additional iteration of three-body interaction computation for triplets with order three rotational symmetric pattern. When N is relatively big and thereby a lot of iterations are needed, this additional iteration has little impact to the overall system performance and hence we can claim that nearly perfect load-balancing is obtained.

```
Thread(i, N)
{
   localE ← 0.0;                                  // initialization
   particle1 ← i;                                 // particle 1
   range ← ⌊(N-1)/3⌋;                             // range of particle 2
   for dij ← 1 to range {
      djk ← dij;
      dki ← N - djk - dij;
      particle2 ← (i + dij) mod N;                // particle 2
      while (dki > dij) {                         // range of particle 3
         particle3 ← (i + dij + djk) mod N; // particle 3
         localE ← localE + TripleE(particle1, particle2, particle3);
         djk ← djk + 1;
         dki ← dki - 1;
      }
   }
   //Special handling 3-way rotational symmetric
   if (N mod 3 == 0) {          // N divisible by 3
      if (i <= N/3) {
        dij = djk = N/3;
        particle2 ← (i + dij) mod N;              // particle 2
        particle3 ← (i + dij + djk) mod N;  // particle 3
        localE ← localE + TripleE(particle1, particle2, particle3);
      }
   }
   local_sum_reduce(localE);    // sum partial scores in local block
}
```

Figure 49 Pseudocode of load-balancing workload distribution scheme. Perfect load-balancing is achieved when N is not divisible by 3. When N is divisible by 3, an additional iteration is needed for the first N/3 threads.

Figure 50(a), (b), and (c) show the workload distribution when N = 7, 8, and 9, respectively, with perfect load-balancing when N = 7 and 8 and nearly perfect load-balancing when N = 9. In addition to load balancing, the workload distribution scheme is particularly suitable for the GPU's SIMT architecture [109]. This is due to the fact that, at each iteration step, each thread reads data from different particles with the same stride, which can be coalesced into efficient memory transactions.

7.2.2.4 Additional Performance Improvement Implementations on GPU

The above load-balancing workload distribution scheme assumes the one-thread-per-particle on GPU architecture. Nevertheless, for molecular systems with small N value, the one-thread-per-particle scheme with N threads may not produce enough threads to fully utilize all resources in GPU. To address this issue, we implement fine-grained threads by dividing the workload originally assigned to one thread to multiple threads so that sufficient threads are produced when N is small. In addition to fine-grained treads, other standard CUDA programming techniques, including parallel sum reduction, loop unrolling, and coalesce memory access are implemented in order to fully take advantage of the GPU architecture [109].

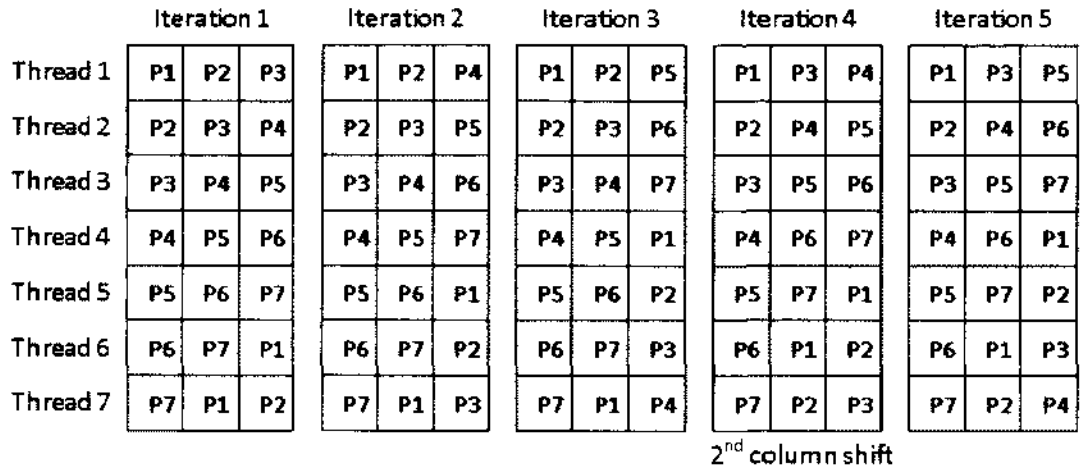


Figure 50 (a) Perfect Balancing (*N* = 7).

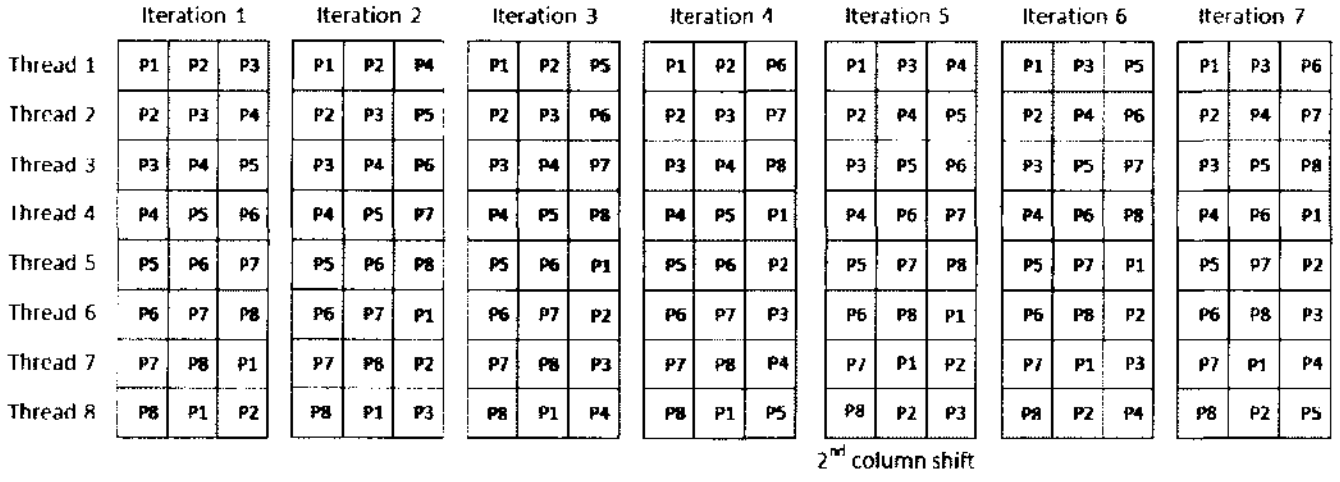


Figure 50 (b) Perfect Balancing (N = 8).

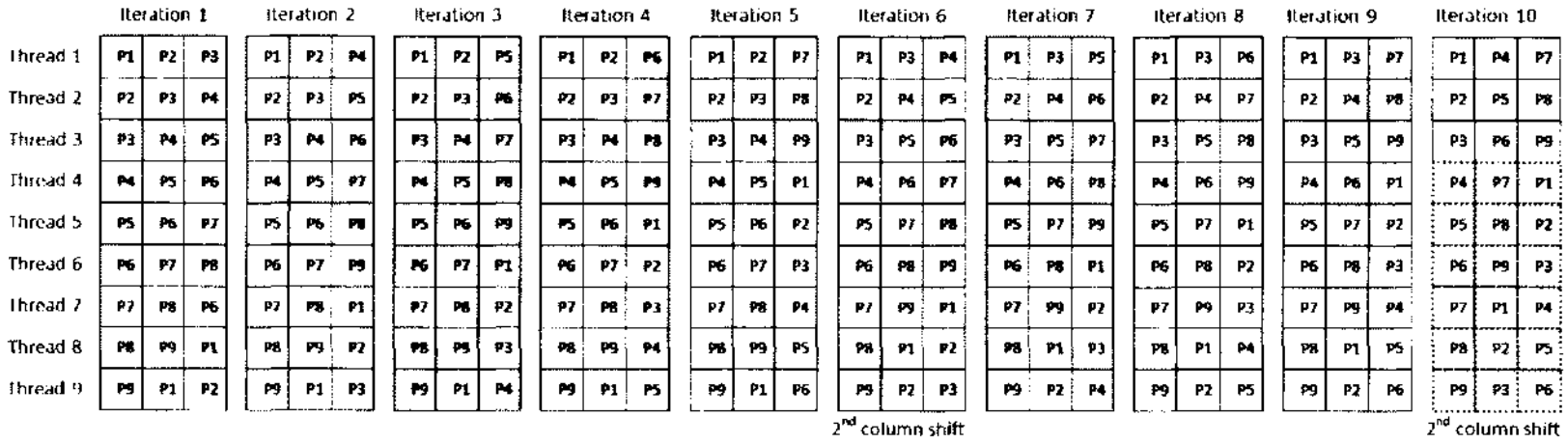


Figure 50 (c) N = 9, due to order three rotational symmetry, only the first three threads handles triplets (1, 4, 7), (2, 5, 8), and (3, 6, 9), respectively, at iteration 10.

Figure 50. Workload Distribution Scheme when N = 7, 8, 9. Perfect load balancing is achieved when N = 7 and 8. Near-perfect load balancing is obtained when N = 9.

7.2.3 Computational Results

Two potentials involving three-body terms, including the Axillord-Teller potential and the CSSP potential are used to demonstrate the effectiveness of our load-balancing workload distribution scheme on GPU. We name the GPU implementation of Axillord-Teller and CSSP potential energy functions "GPU-AxT" and "GPU-CSSP," respectively. The original serial CPU versions are referred to as "CPU-AxT" and "CPU-CSSP." The load-balancing workload distribution scheme for three-body interactions is adopted in GPU-AxT and GPU-CSSP. As for the two-body interactions in GPU-CSSP, we used our approach, described in section 7.1, to balance the workload [135]. Furthermore, the standard CUDA programming techniques for performance improvement are implemented in both potentials.

The GPU-AxT and GPU-CSSP programs are tested on a server with Tesla C2070 GPU. The Tesla C2070 GPU (Fermi architecture) has 14 multiprocessors with 32 cores each, 6 GB of global memory, 64 KB of RAM which can be configured between Shared Memory and L1 cache and 32 KB of registers per multiprocessor. We also tested GPU-CSSP program on two other servers with NVIDIA Tesla C1060 on one server and Tesla C870 on the other server. Tesla C1060 has 30 multiprocessors with 8 cores each, 4 GB of global memory, and 16 KB of registers per multiprocessor. Tesla C870 has 16 multiprocessors with 8 cores each, 2 GB of global memory, and 16 KB of registers per multiprocessor. Both Tesla C1060 and Tesla C870 are non-Fermi architecture with no L1 cache.

CPU-AxT and CPU-CSSP run on a server with an Intel(R) Xeon(R) CPU @ 2.40 GHz, 1.6 GB cache, and 70 GB memory. We benchmark GPU-AxT and GPU-CSSP on a set of systems of various sizes. The GPU time we measured includes the time of transferring the system information (particles) arrays to GPU device memories, GPU execution time, and the time of retrieving the calculated overall potential energy from GPU. We use the gcc compiler with "-O3" optimization flag for the CPU

implementations and nvcc compiler in CUDA 2.0 with "-O3" flag for GPU implementations.

7.2.3.1 Computational Results of Axillord-Teller Potential

- *Axillord-Teller Potential*

The Axillord-Teller potential is an intermolecular potential for the interaction of the van der Waals type between three particles [119]. Considering particles i, j, and k, the Axillord-Teller potential u_{ijk} is calculated as,

$$u_{ijk} = v[\frac{1}{r_{ij}^3 r_{ik}^3 r_{jk}^3} + \frac{3(-r_{ij}^2 + r_{ik}^2 + r_{jk}^2)(r_{ij}^2 - r_{ik}^2 + r_{jk}^2)(r_{ij}^2 + r_{ik}^2 - r_{jk}^2)}{8(r_{ij}^5 r_{ik}^5 r_{jk}^5)}]$$

where v is a non-additive coefficient and r_{ij}, r_{jk}, and r_{ik} are Euclidean distances between particles i and j, j and k, and k and i, respectively.

- *GPU-AxT Speedup over CPU-AxT*

We employed the Axillord-Teller potential in a molecular dynamics simulation for Argon gas. A simulation box of length L is initialized with N number of argon particles (atoms). In order to demonstrate the effectiveness of Axillord-Teller potential implementation on the GPU (GPU-AxT), we run the simulation for 10 steps in boxes of various sizes (L ranges from 10 for 125 particles to 58 for 24389 particles). The execution time of Axillord-Teller potential evaluation is averaged over the number of simulation steps.

Figure 51 shows the overall speedup of GPU-AxT on NVIDIA Tesla C2070 on systems of various sizes with respect to CPU AxT. For very large systems with more than 3,000 particles, the maximum speedups of GPU–AxT can reach ~340. Figure 51 also shows that there are certain inefficiencies in GPU–AxT for systems with less than 2,500 particles, due to insufficient number of threads to fully take advantage of the GPU architecture. To improve the performance of systems with small number of particles, we adopt a fine-grained implementation by evenly splitting the workload of three-body interaction computations belonging to one thread to multiple threads so that nearly the maximum number of threads that a GPU device can launch is created.

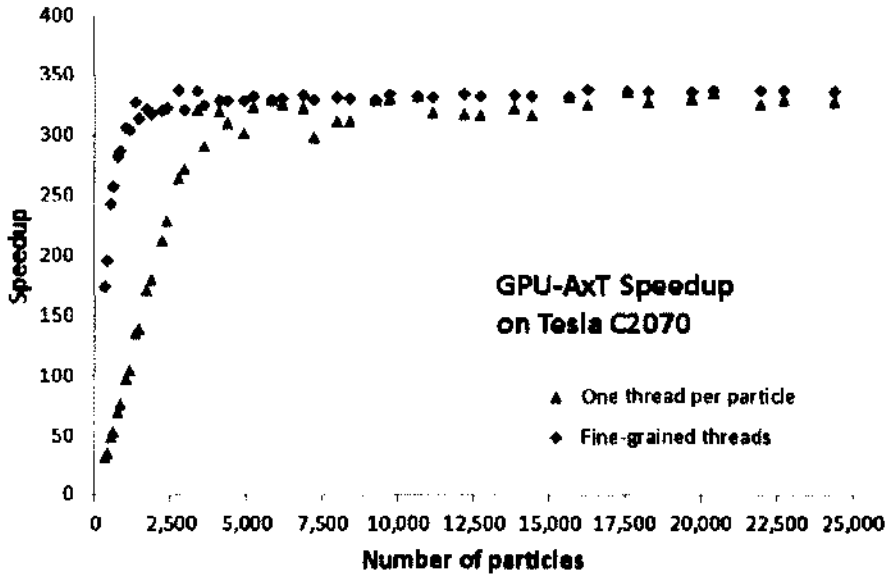


Figure 51 Overall speedup of GPU–AxT on Tesla C2070 on a set of systems with various sizes with respect to CPU–AxT.

Significant performance improvements are found in systems with small number of particles when fine-grained threads are employed, whose speedups are promoted to be closer to those with a lot of particles. For systems with small number of particles, sufficiently large number of threads is produced to use the GPU hardware to its fullest. As the number of particles increases, the speedup curves of fine-grain threads and one-thread-per-particle start to merge because the increasing number of threads in one-thread-per-particle strategy makes more and more efficient use of the GPU architecture. After all, with sufficient number of threads, the memory access latency can be effectively masked.

- *Load Balancing Scheme vs. Direct Mapping Scheme*

Theoretically, assuming every three particles are interacting with each other, if the serial algorithm is directly mapped to GPU implementation, the longest thread needs to carry out $(N-1)*(N-2)/2$ three-body interaction calculations. When the load-balancing scheme is used, each thread handles at most $(N-1)*(N-2)/6+1$ three-body interactions. Therefore, the theoretical speedup of the load-balancing scheme over direct

mapping scheme is approximately 3. Figure 52 shows that, when no distance cutoff is applied, the GPU-AxT implementation using the load-balancing scheme is around 3 times faster than that of direct mapping. This agrees well with our theoretical analysis. Nevertheless, in practice, interactions between particles separated over a certain distance are weak enough to be ignored in computation. Using the distance cutoff can significantly reduce the overall three-body interaction computation. Figure 52 also shows that the speedup of the GPU-AxT implementation using the load-balancing scheme when half of box width is used as cutoff distance over that of direct mapping is reduced to approximately 1.8.

The performance reduction of the balanced GPU-AxT with distance cutoff is mainly caused by the divergent branches in the program. Evaluation of the Axillord-Teller potential with distance cutoff requires testing pairwise particle distances — if any one of the pairwise particle distances is higher than the cutoff distance, the three-body interaction computation will not be carried out.

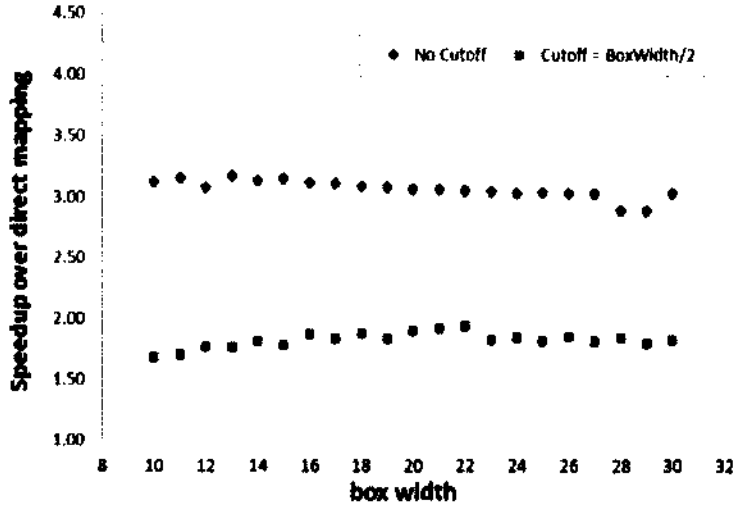


Figure 52 Speedup of GPU-AxT implementation using load-balancing scheme over that of direct mapping. When no cutoff distance is applied, speedup is approximately 3, agreeing well with the theoretical analysis. When half box width cutoff is adopted, speedup is reduced to ~1.8.

When the threads handling three-body interactions within the distance cutoff as well as those exceeding cutoff co-reside in the same GPU warp, divergent branches will occur in runtime. Table 28 compares the number of divergent branches of GPU-AxT with half box width cutoff with GPU-AxT without distance cutoff. The performance data is obtained by NVIDIA Compute Visual Profiler 3.2 [115]. When no distance cutoff is adopted, GPU-AxT does not suffer from branch divergence because pairwise particle distances are not necessarily checked against cutoff distance. In contrast, when distance cutoff is applied, branch instructions (if statements) are inserted to compare the pairwise particle distances with the cutoff distance, which potentially leads to divergent branches. When half box width cutoff is used, the divergent branches in GPU-AxT is approximately 15% of the total number off branch instructions, which results in speedup reduction.

Table 28 Comparison of the Number of divergent branches in GPU-AxT with half box cutoff distance and GPU-AxT without cutoff.

	# of Branch Instruction		# of Divergent Branch	
Box Width	no Cutoff	w. Cutoff	no Cutoff	w. Cutoff
10	2,628	6,722	0	1,319
12	7,817	16,954	0	1,338
14	39,338	91,987	0	11,766
16	87,558	207,909	0	27,614
18	266,098	640,687	0	89,429
20	500,511	1,182,504	0	152,506
22	1,181,940	2,858,410	0	435,072
24	2,489,780	5,809,260	0	730,359
26	4,829,030	11,573,100	0	1,635,830
28	8,787,680	20,750,600	0	2,717,490
30	15,192,000	36,138,900	0	5,058,260

7.2.3.2 *Computational Results of Context-based Secondary Structure Potential (CSSP)*

- *Context-based Secondary Structure Potential (CSSP)*

CSSP is a statistical potential integrating inter-residue interaction potentials for assessing predicted protein secondary structures [94]. Consider a protein chain with L amino acid residues and fragment size of S ($S < L$), the CSSP potential of a protein molecule is calculated as,

$$U_{protein} = \sum_{i=0}^{L-S+1} \left(\sum_{i}^{S} U(R_i) + \sum_{i \neq j}^{S} U(R_i, R_j) + \sum_{i \neq j \neq k}^{S} U(R_i, R_j, R_k) \right) - \sum_{i=1}^{L-S-1} \left(\sum_{i}^{S-1} U(R_i) + \sum_{i \neq j}^{S-1} U(R_i, R_j) + \sum_{i \neq j \neq k}^{S-1} U(R_i, R_j, R_k) \right),$$

where R_i denotes residue i in the protein chain and $U(R_i)$, $U(R_i, R_j)$, and $U(R_i, R_j, R_k)$ are singlet, doublet, and triplet potential terms, respectively.

- *Performance of Load-balancing Scheme*

Unlike the Axillord-Teller potential, which is a pure three-body potential, the CSSP potential includes three-body terms together with two- and single-body terms. In GPU-CSSP implementation, the single-body terms are calculated using direct mapping and the two-body terms are calculated using the pairwise load-balancing scheme described in [135], which has a theoretical speedup of ~2.0 over direct mapping scheme. The three-body terms are calculated using the load-balancing workload distribution scheme described in this paper with theoretical speedup of ~3.0 over direct mapping. Figure 53 shows the speedup of the load-balancing GPU-CSSP over that of direct-mapping on a set of proteins ranged from tens to hundreds of residues, where an average speedup of 2.6 is obtained. This speedup is consistent for small and large proteins.

Figure 54 shows the overall speedup of the final GPU-CSSP implementation with fine-grained threads on NVIDIA Tesla C2070, Tesla C1060, and Tesla C870 on proteins of size ranging from tens to thousand residues with respect to CPU-CSSP. The computation time measures are based on the average of the times on proteins of sizes within the ranges specified in the figure. For large proteins, the maximum speedups of GPU–CSSP can reach up to ~480, ~190 and ~55 using Tesla C2070, Tesla C1060 and

Tesla C870, respectively. It is important to notice that Tesla C2070 yields significantly higher speedups than Tesla C1060 and Tesla C870. This is due to the fact that Tesla C2070 has 448 CUDA cores whereas Tesla C1060 and Tesla C870 have 240 and 128 CUDA cores, respectively. Moreover, Tesla C1060 and Tesla C870 are non-Fermi architecture without L1 cache.

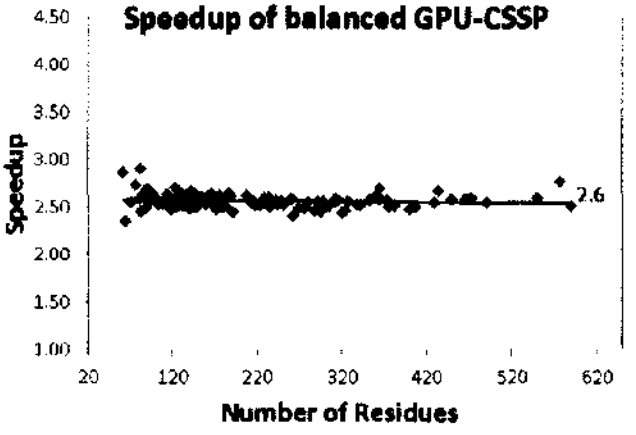


Figure 53 Performance of the balanced GPU-CSSP with respect to the unbalanced GPU-CSSP on Tesla C2070.

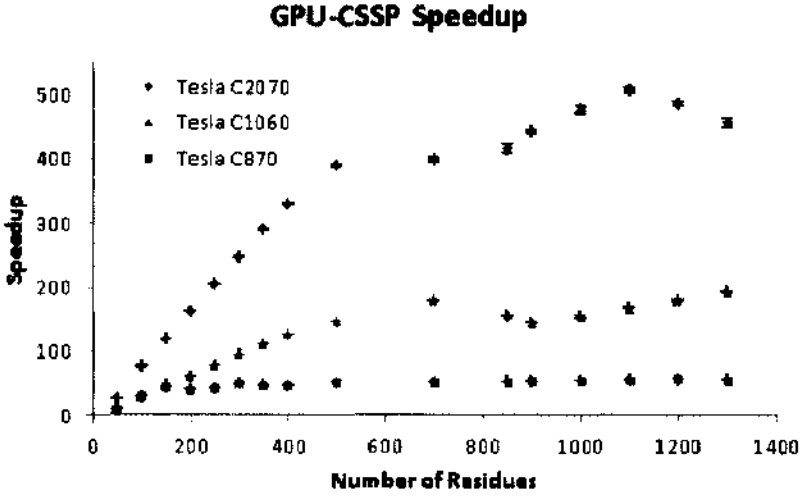


Figure 54 Overall speedup of GPU–CSSP on Tesla C2070, Tesla C1060 and Tesla C870 on a set of proteins with various sizes with respect to CPU–CSSP.

CHAPTER 8

SUMMARY AND POSTDISSERTATION REASEARCH

An approach of deriving context-based scores based on the potentials of mean force method for characterizing the favorability of residues in adopting a structural state according to their amino acid environment is developed in this work. These context-based scores are incorporated as features together with other sequence and evolutionary information in neural network training for different structural features predictions, including secondary structure in 3-state and 8-state, disulfide bonds, and solvent accessibility. Furthermore, efficient load balancing schemes to accelerate the calculations of many-body potentials by taking advantage of the GPU architecture are also developed in this work.

The effectiveness of using context-based scores has been demonstrated in our computational results in *N*-fold cross validation as well as on protein benchmarks, where enhancements of prediction accuracies are observed. A comparison of our methods with a set of popular structural features prediction methods was made such that our methods demonstrate higher accuracies. Furthermore, the efficiency of our proposed load-balancing approach in accelerating many-body potentials has been demonstrated in the corresponding results as well.

Web servers implementing our prediction methods are currently available:

- DINOSOLVE, available at http://hpcr.cs.odu.edu/dinosolve.
- C3-SCORPION, available at http://hpcr.cs.odu.edu/c3scorpion.
- C8-SCORPION, available at http://hpcr.cs.odu.edu/c8scorpion.
- CASA, available at http://hpcr.cs.odu.edu/casa.

This dissertation raises many interesting opportunities for us to continue our study in our future post-dissertation research. The following is a list of some of these possible research directions:

- **Toward the theoretical upper bound**

In order to close the gaps between the theoretical upper bounds and our current prediction accuracies of secondary structures, disulfide bonds and solvent accessibility, our future efforts will include:

1) Deriving better context-based scores for calculating high-order interactions.

The context-based scores estimate the favorability of a residue adopting certain structural conformation within its amino acid environment. The contribution of the context-based scores in structural features prediction depends on their accuracy.

In order to show the sensitivity to context-based scores accuracy, we hereby use our SCORPION method for predicting secondary structures in 3-state as an example. Figure 55 shows the distribution of the context-based scores accuracy and the corresponding accuracy improvements over the PSSM-only predictions for 3-state secondary structure in CB513, CASP9, Manesh215, and Carugo338 benchmarks. When the accuracies are lower than 40%, the context-based scores are close to random and thus their contributions to secondary structure prediction are marginal. The more accurate in the scores, the higher accuracy improvement over the PSSM-only predictions is achieved. When the accuracy of the context-based scores exceeds 80%, the average accuracy improvement over PSSM-only predictions reaches 2.5%. Unfortunately, the quality of the context-based scores is limited by number of samples existing in the PDB, particularly when calculating the high-order interactions. Therefore, it is not often that the context-based scores are highly accurate and the average of the context-based scores accuracy in SCORPION training and prediction is around 60%. Hence, more samples from the PDB will be collected in order to calculate the residues' high-order interactions.

2) Obtaining more precise PSSM substitution matrices

The evolutionary information revealed by multiple sequence alignments is a major component of almost all modern prediction methods. The quality of this information depends on the alignment algorithm used. Hence, our research direction involves investigating better sequence alignment algorithms on increasingly large sequence databases.

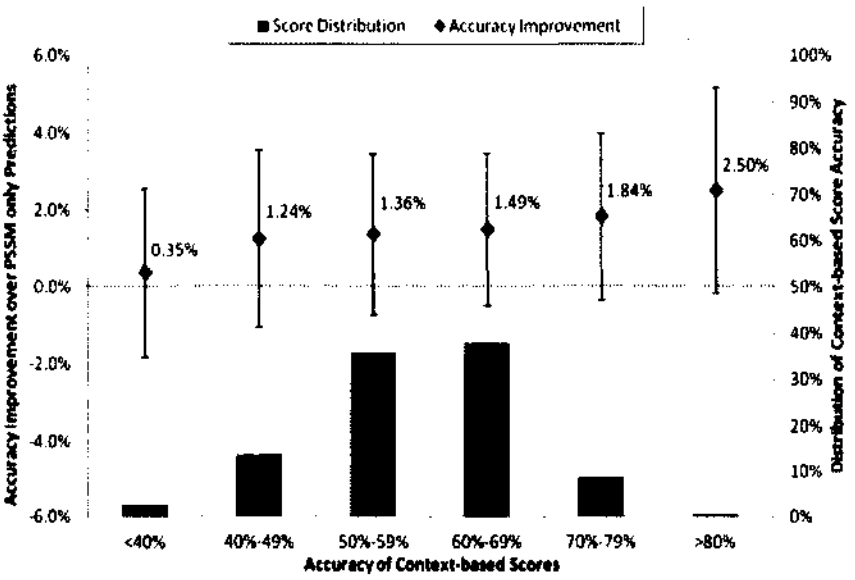


Figure 55 Distribution of context-based scores accuracy and its correlation with accuracy improvement over PSSM only prediction in CB513, CASP9, Manesh215, and Carugo338 benchmarks. The accuracy of context-based scores is measured by calculating the percentage of residues whose lowest score secondary structure conformations (H, E, or C) agree with the DSSP.

3) Developing advanced machine learning algorithms

More machine learning algorithms will be explored; specifically, algorithms that can capture residue-residue interactions in longer range and handle increasingly large number of known protein structures.

- **Secondary structure prediction and intrinsically disordered protein regions**

An important application of secondary structure prediction is to predict intrinsically disordered protein regions. Disordered proteins typically have a low content of secondary structures. In fact, the predicted secondary structures are often incorporated as important information in disorder region predictors such as DISOPRED [136] and SPINE-D [137]. Figure 56 shows an example of the secondary structures predicted by SCORPION on Thylakoid soluble phosphoprotein TSP9 [138], an intrinsic disordered protein reacting to light condition changes from the photosynthetic membrane. The majority of the N-terminal α-helix is predicted correctly with high confidence (8+) except

for two residues at the end. For the unstructured coils, 85.3% of the 75 residues are correctly identified while the rest 14.7% are misclassified as helices or strands but with low prediction confidence (6-). Coupled with other residue feature predictors such as solvent accessibility, B-factor, and disulfide bonding state, the accuracy improvement in secondary structure prediction using context-based scores has the potential to enhance determination of intrinsically disordered regions.

- **More protein structural features to predict.**

Features including contact-map, torsion angles, disordered regions, and B-factor will be considered as part of our future research.

- **A Framework for tertiary structure prediction**

Prediction methods, once available, will serve as a framework by which important information can be generated in order to be used in tertiary structure predictions. Even though this dissertation focuses specifically on protein structural features prediction, we are planning to work on protein tertiary structure prediction, where the tools and techniques developed in this work can be efficiently applied in order to enhance protein 3D prediction.

- **General GPU-accelerator**

Our future research directions regarding GPU-accelerations of many-body potentials will include investigating the development of a general GPU-accelerated framework that can be easily employed to accelerate complicated energy potential functions.

```
1   SAAKGTAETK QEKSPVIWIL  IF TKEDQFY ETDPILRGGD VKSSGSTSGK   Sequence
    CCCCCCCCCC CCCCHHHHHH HHHCCCCCCC CCCCCCCCCC CCCCCCCCCC   DSSP Assignment
    CCCCCCCCCH HHCCHHHHHH HCCCCCCCCE CCCCCCCCCC CCCCCCCCCC   SCORPION Prediction
    9987887646 6579899997 5667766444 4897667986 5688888888   Prediction Confidence

51  KGGTTSGKKG TVSIPSKKKN GNGGVPGGLF AKKD
    CCCCCCCCCC CCCCCCCCCC CCCCCCCCCC CCCC
    CCCCCCCCCC CHCCCCCCCC CCCCEEECEE ECCC
    8898898888 5647887668 9997654445 4789
```

Figure 66 Secondary Structures Predicted by SCORPION on Thylakoid soluble phosphoprotein TSP9.

REFERENCES

1. Lesk AM, Lo Conte L, Hubbard TJP: **Assessment of novel fold targets in CASP4: Predictions of three-dimensional structures, secondary structures, and interresidue contacts**. *Proteins* 2001:98-118.

2. Taylor WR: **The Classification of Amino-Acid Conservation**. *J Theor Biol* 1986, **119**(2):205-&.

3. Livingstone CD, Barton GJ: **Protein-Sequence Alignments - a Strategy for the Hierarchical Analysis of Residue Conservation**. *Comput Appl Biosci* 1993, **9**(6):745-756.

4. Dill KA, Bromberg S, Yue KZ, Fiebig KM, Yee DP, Thomas PD, Chan HS: **Principles of Protein-Folding - a Perspective from Simple Exact Models**. *Protein Sci* 1995, **4**(4):561-602.

5. **A protein sequence culling server** [http://dunbrack.fccc.edu/Guoli/pisces_download.php]

6. Sanger F, Thompson EO, Kitai R: **The amide groups of insulin**. *Biochem J* 1955, **59**(3):509-518.

7. Fersht AR: **Max Ferdinand Perutz OM FRS - Obituary**. *Nat Struct Biol* 2002, **9**(4):245-246.

8. Clegg W: **Crystal structure determination**. Oxford: Oxford University Press; 1998.

9. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE: **The Protein Data Bank**. *Nucleic Acids Res* 2000, **28**(1):235-242.

10. Sussman JL, Lin DW, Jiang JS, Manning NO, Prilusky J, Ritter O, Abola EE: **Protein Data Bank (PDB): Database of three-dimensional structural information of biological macromolecules**. *Acta Crystallogr D* 1998, **54**:1078-1084.

11. **Nuclear magnetic resonance spectroscopy** [http://en.wikipedia.org/wiki/Nuclear_magnetic_resonance_spectroscopy]

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

13. Segre ABaAM: **Ab Initio Methods for Protein Structure Prediction: A New Technique based on Ramachandran Plots**. In.: The European Research Consortium for Informatics and Mathematics (ERCIM); 2000.

14. Chuang CC, Chen CY, Yang JM, Lyu PC, Hwang JK: **Relationship between protein structures and disulfide bonding patterns**. *Proteins-Structure Function and Genetics* 2003, **53**(1):1-5.

15. Fass D: **Disulfide Bonding in Protein Biophysics**. *Annu Rev Biophys* 2012, **41**:63-79.

16. Ahmad S, Gromiha MM, Sarai A: **Real value prediction of solvent accessibility from amino acid sequence**. *Proteins-Structure Function and Genetics* 2003, **50**(4):629-635.

17. Chan HS, Dill KA: **Origins of structure in globular proteins**. *Proc Natl Acad Sci U S A* 1990, **87**(16):6388-6392.

18. Ooi T, Oobatake M, Nemethy G, Scheraga HA: **Accessible surface areas as a measure of the thermodynamic parameters of hydration of peptides**. *Proc Natl Acad Sci U S A* 1987, **84**(10):3086-3090.

19. Ehrlich L, Reczko M, Bohr H, Wade RC: **Prediction of protein hydration sites from sequence by modular neural networks**. *Protein Eng* 1998, **11**(1):11-19.

20. Anfinsen CB: **Principles That Govern Folding of Protein Chains**. *Science* 1973, **181**(4096):223-230.

21. Levintha.C: **Are There Pathways for Protein Folding**. *J Chim Phys Pcb* 1968, **65**(1):44-&.

22. Moult J, Fidelis K, Kryshtafovych A, Rost B, Tramontano A: **Critical assessment of methods of protein structure prediction-Round VIII**. *Proteins* 2009, **77**:1-4.

23. Ginalski K: **Comparative modeling for protein structure prediction**. *Curr Opin Struc Biol* 2006, **16**(2):172-177.

24. Tress M, Ezkurdia L, Grana O, Lopez G, Valencia A: **Assessment of predictions submitted for the CASP6 comparative modeling category**. *Proteins* 2005, **61**:27-45.

25. John B, Sali A: **Comparative protein structure modeling by iterative alignment, model building and model assessment**. *Nucleic Acids Res* 2003, **31**(14):3982-3992.

26. Zhou HY, Zhou YQ: **Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction**. *Protein Sci* 2002, **11**(11):2714-2726.

27. Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M: **Charmm - a Program for Macromolecular Energy, Minimization, and Dynamics Calculations**. *J Comput Chem* 1983, **4**(2):187-217.

28. Wessam Elhefnawy LC, Yun Han, Yaohang Li: **ICOSA: A Distance-dependent, Orientation-specific Coarse-grain Contact Potential for Protein Structure Modeling**. *Journal of Molecular Biology, under review* 2014.

29. Rohl CA, Strauss CEM, Misura KMS, Baker D: **Protein structure prediction using rosetta**. *Method Enzymol* 2004, **383**:66-+.

30. Chou PY, Fasman GD: **Prediction of Protein Conformation**. *Biochemistry-Us* 1974, **13**(2):222-245.

31. Fiser A, Cserzo M, Tudos E, Simon I: **Different Sequence Environments of Cysteines and Half Cystines in Proteins Application to Predict Disulfide Forming Residues**. *Febs Lett* 1992, **302**(2):117-120.

32. Qian N, Sejnowski TJ: **Predicting the Secondary Structure of Globular-Proteins Using Neural Network Models**. *J Mol Biol* 1988, **202**(4):865-884.

33. Rost B, Sander C: **Prediction of Protein Secondary Structure at Better Than 70-Percent Accuracy**. *J Mol Biol* 1993, **232**(2):584-599.

34. Rost B, Sander C: **Secondary structure prediction of all-helical proteins in two states**. *Protein Eng* 1993, **6**(8):831-836.

35. Rost B, Sander C: **Combining evolutionary information and neural networks to predict protein secondary structure**. *Proteins* 1994, **19**(1):55-72.

36. Jones DT: **Protein secondary structure prediction based on position-specific scoring matrices**. *J Mol Biol* 1999, **292**(2):195-202.

37. Pollastri G, Przybylski D, Rost B, Baldi P: **Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles**. *Proteins-Structure Function and Genetics* 2002, **47**(2):228-235.

38. Pollastri G, McLysaght A: **Porter: a new, accurate server for protein secondary structure prediction**. *Bioinformatics* 2005, **21**(8):1719-1720.

39. Lin K, Simossis VA, Taylor WR, Heringa J: **A simple and fast secondary structure prediction method using hidden neural networks**. *Bioinformatics* 2005, **21**(2):152-159.

40. Dor O, Zhou YQ: **Achieving 80% ten-fold cross-validated accuracy for secondary structure prediction by large-scale training**. *Proteins* 2007, **66**(4):838-845.

41. Cole C, Barber JD, Barton GJ: **The Jpred 3 secondary structure prediction server**. *Nucleic Acids Res* 2008, **36**:W197-W201.

42. Rost B, Eyrich VA: **EVA: large-scale analysis of secondary structure prediction**. *Proteins* 2001, **Suppl 5**:192-199.

43. Rost B: **Review: Protein secondary structure prediction continues to rise**. *J Struct Biol* 2001, **134**(2-3):204-218.

44. Wang Z, Zhao F, Peng J, Xu J: **Protein 8-class secondary structure prediction using conditional neural fields**. *Proteomics* 2011, **11**(19):3786-3792.

45. Fariselli P, Riccobelli P, Casadio R: **Role of evolutionary information in predicting the disulfide-bonding state of cysteine in proteins**. *Proteins-Structure Function and Genetics* 1999, **36**(3):340-346.

46. Fiser A, Simon I: **Predicting the oxidation state of cysteines by multiple sequence alignment**. *Bioinformatics* 2000, **16**(3):251-256.

47. Mucchielli-Giorgi MHM, Hazout S, Tuffery P: **Predicting the disulfide bonding state of cysteines using protein descriptors**. *Proteins-Structure Function and Genetics* 2002, **46**(3):243-249.

48. Ceroni A, Frasconi P, Passerini A, Vullo A: **Predicting the disulfide bonding state of cysteines with combinations of kernel machines**. *J Vlsi Sig Proc Syst* 2003, **35**(3):287-295.

49. Martelli PL, Fariselli P, Malaguti L, Casadio R: **Prediction of the disulfide bonding state of cysteines in proteins with hidden neural networks**. *Protein Engineering* 2002, **15**(12):951-953.

50. Song JN, Wang ML, Li WJ, Xu WB: **Prediction of the disulfide-bonding state of cysteines in proteins based on dipeptide composition**. *Biochem Bioph Res Co* 2004, **318**(1):142-147.

51. Fariselli P, Casadio R: **Prediction of disulfide connectivity in proteins**. *Bioinformatics* 2001, **17**(10):957-964.

52. Ceroni A, Passerini A, Vullo A, Frasconi P: **DISULFIND: a disulfide bonding state and cysteine connectivity prediction server**. *Nucleic Acids Res* 2006, **34**:W177-W181.

53. Ferre F, Clote P: **DiANNA: a web server for disulfide connectivity prediction**. *Nucleic Acids Res* 2005, **33**:W230-W232.

54. Cheng JL, Saigo H, Baldi P: **Large-scale prediction of disulphide bridges using kernel methods, two-dimensional recursive neural networks, and weighted graph matching**. *Proteins* 2006, **62**(3):617-629.

55. Vincent M, Passerini A, Labbe M, Frasconi P: **A simplified approach to disulfide connectivity prediction from protein sequences**. *Bmc Bioinformatics* 2008, **9**.

56. Faraggi E, Xue B, Zhou Y: **Improving the prediction accuracy of residue solvent accessibility and real-value backbone torsion angles of proteins by guided-learning through a two-layer neural network**. *Proteins* 2009, **74**(4):847-856.

57. Dor O, Zhou YQ: **Real-SPINE: An integrated system of neural networks for real-value prediction of protein structural properties**. *Proteins* 2007, **68**(1):76-81.

58. Ahmad S, Gromiha MM: **NETASA: neural network based prediction of solvent accessibility**. *Bioinformatics* 2002, **18**(6):819-824.

59. Adamczak R, Porollo A, Meller J: **Accurate prediction of solvent accessibility using neural networks-based regression**. *Proteins* 2004, **56**(4):753-767.

60. Petersen B, Petersen TN, Andersen P, Nielsen M, Lundegaard C: **A generic method for assignment of reliability scores applied to solvent accessibility predictions**. *Bmc Struct Biol* 2009, **9**.

61. Pollastri G, Baldi P, Fariselli P, Casadio R: **Prediction of coordination number and relative solvent accessibility in proteins**. *Proteins-Structure Function and Genetics* 2002, **47**(2):142-153.

62. Yuan Z, Burrage K, Mattick JS: **Prediction of protein solvent accessibility using support vector machines**. *Proteins-Structure Function and Genetics* 2002, **48**(3):566-570.

63. Kim H, Park H: **Prediction of protein relative solvent accessibility with support vector machines and long-range interaction 3D local descriptor**. *Proteins* 2004, **54**(3):557-562.

64. Joo K, Lee SJ, Lee J: **Sann: Solvent accessibility prediction of proteins by nearest neighbor method**. *Proteins* 2012, **80**(7):1791-1797.

65. Sim J, Kim SY, Lee J: **Prediction of protein solvent accessibility using fuzzy k-nearest neighbor method**. *Bioinformatics* 2005, **21**(12):2844-2849.

66. Naderi-Manesh H, Sadeghi M, Arab S, Movahedi AAM: **Prediction of protein surface accessibility with information theory**. *Proteins-Structure Function and Genetics* 2001, **42**(4):452-459.

67. Thompson MJ, Goldstein RA: **Predicting solvent accessibility: Higher accuracy using Bayesian statistics and optimized residue substitution classes**. *Proteins-Structure Function and Genetics* 1996, **25**(1):38-47.

68. Rost B, Sander C: **Conservation and Prediction of Solvent Accessibility in Protein Families**. *Proteins-Structure Function and Genetics* 1994, **20**(3):216-226.

69. Cuff JA, Barton GJ: **Application of multiple sequence alignment profiles to improve protein secondary structure prediction**. *Proteins-Structure Function and Genetics* 2000, **40**(3):502-511.

70. **SAS prediction server** [http://140.113.239.214/~weilun/index.php]

71. **NVIDIA** [http://www.nvidia.com/page/home.html]

72. Owens JD, Luebke D, Govindaraju N, Harris M, Kruger J, Lefohn AE, Purcell TJ: **A survey of general-purpose computation on graphics hardware**. *Comput Graph Forum* 2007, **26**(1):80-113.

73. Owens JD, Houston M, Luebke D, Green S, Stone JE, Phillips JC: **GPU computing**. *P Ieee* 2008, **96**(5):879-899.

74. Stone JE, Hardy DJ, Ufimtsev IS, Schulten K: **GPU-accelerated molecular modeling coming of age.** *J Mol Graph Model* 2010, **29**(2):116-125.

75. Zhu WH: **Massively parallel differential evolution-pattern search optimization with graphics hardware acceleration: an investigation on bound constrained optimization problems.** *J Global Optim* 2011, **50**(3):417-437.

76. Zhu WH: **Nonlinear optimization with a massively parallel Evolution Strategy-Pattern Search algorithm on graphics hardware.** *Appl Soft Comput* 2011, **11**(2):1770-1781.

77. Zhou Y, Tan Y: **GPU-based Parallel Particle Swarm Optimization.** *Ieee C Evol Computat* 2009:1493-1500.

78. You Y: **Parallel Ant System for Traveling Salesman Problem on GPUs.** In: *ACM Genetic and Evolutionary Computation Conference.* 2009.

79. Bakhtiari M, Malhotra H, Jones MD, Chaudhary V, Walters JP, Nazareth D: **Applying graphics processor units to Monte Carlo dose calculation in radiation therapy.** *J Med Phys* 2010, **35**(2):120-122.

80. Lars Nyland MH, Jan Prins: **Fast N-Body Simulation with CUDA.** In: *GPU Gems 3.* NVIDIA; 2007.

81. M. J. Stock AG: **Toward efficient GPU-accelerated N-body simulations.** In: *46th AIAA Aerospace Sciences Meeting and Exhibit.* 2008.

82. Anderson JA, Lorenz CD, Travesset A: **General purpose molecular dynamics simulations fully implemented on graphics processing units.** *J Comput Phys* 2008, **227**(10):5342-5359.

83. Friedrichs MS, Eastman P, Vaidyanathan V, Houston M, Legrand S, Beberg AL, Ensign DL, Bruns CM, Pande VS: **Accelerating Molecular Dynamic Simulation on Graphics Processing Units.** *J Comput Chem* 2009, **30**(6):864-872.

84. Belleman RG, Bedorf J, Zwart SFP: **High performance direct gravitational N-body simulations on graphics processing units II: An implementation in CUDA.** *New Astron* 2008, **13**(2):103-112.

85. Vondervisztt F, Matrai G, Simon I: **Characteristic Sequential Residue Environment of Amino-Acids in Proteins.** *Int J Pept Prot Res* 1986, **27**(5):483-492.

86. Carugo O: **Predicting residue solvent accessibility from protein sequence by considering the sequence environment.** *Protein Engineering* 2000, **13**(9):607-609.

87. Kinch LN, Shi S, Cheng H, Cong Q, Pei JM, Mariani V, Schwede T, Grishin NV: **CASP9 target classification.** *Proteins* 2011, **79**:21-36.

88. Sippl MJ: **Calculation of Conformational Ensembles from Potentials of Mean Force - an Approach to the Knowledge-Based Prediction of Local Structures in Globular-Proteins**. *J Mol Biol* 1990, **213**(4):859-883.

89. Zemla A, Venclovas C, Fidelis K, Rost B: **A modified definition of Sov, a segment-based measure for protein secondary structure prediction assessment**. *Proteins-Structure Function and Genetics* 1999, **34**(2):220-223.

90. **Matthews correlation coefficient** [http://en.wikipedia.org/wiki/Matthews_correlation_coefficient]

91. Yaseen A, Li Y: **Context-based features enhance protein secondary structure prediction accuracy**. *Journal of chemical information and modeling* 2014, **54**(3):992-1002.

92. Ashraf Yaseen YL: **Template-based C8-SCORPION: a protein 8-state secondary structure prediction method using structural information and context-based features**. *Bmc Bioinformatics* 2014.

93. Rata IA, Li YH, Jakobsson E: **Backbone Statistical Potential from Local Sequence-Structure Interactions in Protein Loops**. *J Phys Chem B* 2010, **114**(5):1859-1869.

94. Li Y, Liu H, Rata I, Jakobsson E: **Building a Knowledge-Based Statistical Potential by Capturing High-Order Inter-residue Interactions and its Applications in Protein Secondary Structure Assessment**. *Journal of chemical information and modeling* 2013, **53**(2):500-508.

95. Samudrala R, Moult J: **An all-atom distance-dependent conditional probability discriminatory function for protein structure prediction**. *J Mol Biol* 1998, **275**(5):895-916.

96. Faraggi E, Yang Y, Zhang S, Zhou Y: **Predicting continuous local structure and the effect of its substitution for secondary structure in fragment-free protein structure prediction**. *Structure* 2009, **17**(11):1515-1527.

97. Topf M, Baker ML, Marti-Renom MA, Chiu W, Sali A: **Refinement of protein structures by iterative comparative modeling and CryoEM density fitting**. *J Mol Biol* 2006, **357**(5):1655-1668.

98. Pal L, Basu G: **Neural network prediction of 3(10)-helices in proteins**. *Indian journal of biochemistry & biophysics* 2001, **38**(1-2):107-114.

99. Montgomerie S, Sundararaj S, Gallin WJ, Wishart DS: **Improving the accuracy of protein secondary structure prediction using structural alignment**. *Bmc Bioinformatics* 2006, **7**.

100. Pollastri G, Martin AJM, Mooney C, Vullo A: **Accurate prediction of protein secondary structure and solvent accessibility by consensus combiners of sequence and structure information**. *Bmc Bioinformatics* 2007, **8**.

101. Yaseen A, Li YH: **Dinosolve: a protein disulfide bonding prediction server using context-based features to enhance prediction accuracy**. *Bmc Bioinformatics* 2013, **14**.

102. Muskal SM, Holbrook SR, Kim SH: **Prediction of the Disulfide-Bonding State of Cysteine in Proteins**. *Protein Engineering* 1990, **3**(8):667-672.

103. Sevier CS, Kaiser CA: **Formation and transfer of disulphide bonds in living cells**. *Nat Rev Mol Cell Bio* 2002, **3**(11):836-847.

104. Washington AT, Singh G, Aiyar A: **Diametrically opposed effects of hypoxia and oxidative stress on two viral transactivators**. *Virol J* 2010, **7**.

105. Kim YW, Otterson GA, Kratzke RA, Coxon AB, Kaye FJ: **Differential Specificity for Binding of Retinoblastoma Binding-Protein-2 to Rb, P107, and Tata-Binding Protein**. *Mol Cell Biol* 1994, **14**(11):7256-7264.

106. Jung YS, Bonagura CA, Tilley GJ, Gao-Sheridan HS, Armstrong FA, Stout CD, Burgess BK: **Structure of C42D Azotobacter vinelandii FdI - A Cys-X-X-Asp-X-X-Cys motif ligates an air-stable [4Fe-4S](2+/+) cluster**. *J Biol Chem* 2000, **275**(47):36974-36983.

107. Ashraf Yaseen YL: **CASA: a protein solvent accessibility prediction server using context based features to enhance prediction accuracy**. *BMC Bioinformatics (Invited)* 2014.

108. Kinch L, Shi SY, Cong Q, Cheng H, Liao YX, Grishin NV: **CASP9 assessment of free modeling target predictions**. *Proteins* 2011, **79**:59-73.

109. NVIDIA: **CUDA programming guide version 3.1**. In.; 2010.

110. Zhang C, Liu S, Zhou YQ: **Accurate and efficient loop selections by the DFIRE-based all-atom statistical potential**. *Protein Sci* 2004, **13**(2):391-399.

111. Patterson D: **The top 10 innovations in the new NVIDIA fermi architecture, and the top 3 next challenges**. In.; 2009.

112. Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E: **Equation of State Calculations by Fast Computing Machines**. *Journal of Chemical Physics* 1953, **21**(6):1087-1092.

113. Lee MS, Olson MA: **Assessment of detection and refinement strategies for de novo protein structures using force field and statistical potentials**. *J Chem Theory Comput* 2007, **3**(1):312-324.

114. Piccolboni A, Mauri G: **Application of evolutionary algorithms to protein folding prediction**. *Artificial Evolution* 1998, **1363**:123-135.

115. **Compute Visual Profiler 3.2** [developer.download.nvidia.com]

116. Harris M: **Optimizing parallel reduction in CUDA**. In.: NVIDIA Developer Technology; 2008.

117. **Tinker package** [http://dasher.wustl.edu/tinker/]

118. Li ZQ, Scheraga HA: **Monte-Carlo-Minimization Approach to the Multiple-Minima Problem in Protein Folding.** *P Natl Acad Sci USA* 1987, **84**(19):6611-6615.

119. Axilrod BM, Teller E: **Interaction of the van der Waals type between three atoms.** *Journal of Chemical Physics* 1943, **11**(6):299-300.

120. Elrod MJ, Saykally RJ: **Many-body effects in intermolecular forces**. *Chemical reviews* 1994, **94**(7):1975-1997.

121. Wang L, Sadus RJ: **Influence of two-body and three-body interatomic forces on gas, liquid, and solid phases**. *Physical review E, Statistical, nonlinear, and soft matter physics* 2006, **74**(2 Pt 1):021202.

122. Marcelli G, Sadus RJ: **Molecular simulation of the phase behavior of noble gases using accurate two-body and three-body intermolecular potentials**. *Journal of Chemical Physics* 1999, **111**(4):1533-1540.

123. Marcelli G, Todd BD, Sadus RJ: **Beyond traditional effective intermolecular potentials and pairwise interactions in molecular simulation**. *Computational Science-Iccs 2002, Pt Iii, Proceedings* 2002, **2331**:932-941.

124. Wang L, Sadus RJ: **Three-body interactions and solid-liquid phase equilibria: application of a molecular dynamics algorithm**. *Physical review E, Statistical, nonlinear, and soft matter physics* 2006, **74**(3 Pt 1):031203.

125. Wang L, Sadus RJ: **Effect of three-body interactions on the vapor-liquid phase equilibria of binary fluid mixtures**. *J Chem Phys* 2006, **125**(7):074503.

126. Marcelli G, Sadus RJ: **A link between the two-body and three-body interaction energies of fluids from molecular simulation**. *Journal of Chemical Physics* 2000, **112**(14):6382-6385.

127. Anta JA, Lomba E, Lombardero M: **Exploring the influence of three-body classical dispersion forces on phase equilibria of simple fluids: An integral-equation approach**. *Physical review E, Statistical physics, plasmas, fluids, and related interdisciplinary topics* 1994, **49**(1):402-409.

128. Zhao G, Carson MB, Lu H: **Prediction of specific protein-DNA recognition by knowledge-based two-body and three-body interaction potentials**. *Conf Proc IEEE Eng Med Biol Soc* 2007, **2007**:5017-5020.

129. Feng Y, Kloczkowski A, Jernigan RL: **Four-body contact potentials derived from two protein datasets to discriminate native structures from decoys**. *Proteins* 2007, **68**(1):57-66.

130. Best RB, Mittal J, Feig M, MacKerell AD, Jr.: **Inclusion of many-body effects in the additive CHARMM protein CMAP potential results in enhanced cooperativity of alpha-helix and beta-hairpin formation.** *Biophys J* 2012, **103**(5):1045-1051.

131. Barnes J, Hut P: **A Hierarchical O(N-Log-N) Force-Calculation Algorithm.** *Nature* 1986, **324**(6096):446-449.

132. Darden T, York D, Pedersen L: **Particle Mesh Ewald - an N.Log(N) Method for Ewald Sums in Large Systems.** *Journal of Chemical Physics* 1993, **98**(12):10089-10092.

133. Greengard L: **The rapid evaluation of potential fields in particle systems.** *Thesis (doctoral).* Cambridge, Mass.: MIT Press, Yale University.; 1988.

134. Hockney RW, Eastwood JW: **Computer simulation using particles**, Special student edn. Bristol England ; Philadelphia: A. Hilger; 1988.

135. Yaseen A, Li YH: **Accelerating knowledge-based energy evaluation in protein structure modeling with Graphics Processing Units.** *J Parallel Distr Com* 2012, **72**(2):297-307.

136. Ward JJ, McGuffin LJ, Bryson K, Buxton BF, Jones DT: **The DISOPRED server for the prediction of protein disorder.** *Bioinformatics* 2004, **20**(13):2138-2139.

137. Zhang T, Faraggi E, Xue B, Dunker AK, Uversky VN, Zhou Y: **SPINE-D: accurate prediction of short and long disordered regions by a single neural-network based method.** *Journal of biomolecular structure & dynamics* 2012, **29**(4):799-813.

138. Song J, Lee MS, Carlberg I, Vener AV, Markley JL: **Micelle-induced folding of spinach thylakoid soluble phosphoprotein of 9 kDa and its functional implications.** *Biochemistry-Us* 2006, **45**(51):15633-15643.

VITA

The Author of this dissertation was born in Jordan on August 18, 1980. He got his B.S. in Computer Science and Information Technology from Jordan University of Science and Technology in 2002, and his M.S. in Computer Science from New York Institute of Technology in 2003. Then, he worked as an instructor at Jordan University of Science and Technology. After that, in 2007, he decided to continue his graduate study and pursue his Ph.D. in Computer Science at Old Dominion University, Norfolk, VA.

Currently, the author works as an Assistant Professor of Computer Science at Central State University, Wilberforce, Ohio. His research interests are Computational Biology, Bioinformatics, and High Performance Computing.