2006

Algorithms & experiments for the protein chain lattice fitting problem

Thomas, Dallas

Lethbridge, Alta. : University of Lethbridge, Faculty of Arts and Science, 2006

ALGORITHMS & EXPERIMENTS FOR THE PROTEIN CHAIN LATTICE FITTING PROBLEM

DALLAS THOMAS
Bachelor of Science, University of Lethbridge, 2004
Bachelor of Science, Kings University College, 2002

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

Abstract

This study seeks to design algorithms that may be used to determine if a given lattice is a good approximation to a given rigid protein structure. Ideal lattice models discovered using our techniques may then be used in algorithms for protein folding and inverse protein folding.

In this study we develop methods based on dynamic programming and branch and bound in an effort to identify “ideal” lattice models. To further our understanding of the concepts behind the methods we have utilized a simple cubic lattice for our analysis. The algorithms may be adapted to work on any lattice. We describe two algorithms. One for aligning the protein backbone to the lattice as a walk. This algorithm runs in polynomial time. The second algorithm for aligning a protein backbone as a path to the lattice. Both the algorithms seek to minimize the CRMS deviation of the alignment. The second problem was recently shown to be NP-Complete, hence it is highly unlikely that an efficient algorithm exists. The first algorithm gives a lower bound on the optimal solution to the second problem, and can be used in a branch and bound procedure. Further, we perform an empirical evaluation of our algorithm on proteins from the Protein Data Bank (PDB).

Acknowledgments

There are many people, without whose support, patience and contributions this thesis would not be possible.

Firstly, I would like to thank my advisor, Dr. Daya Gaur, for encouraging me to pursue graduate studies, without him, this thesis would not exist. I would also like to thank him for his support, dedication and humility he has demonstrated throughout this process. I also wish to thank the rest of my thesis comittee, Dr. Shahadat Hossain and Dr. Igor Kovalchuk, and my external reader Dr. Binay Bhattacharya for their advice and support.

I would like to extend thanks to my friends and colleagues for providing their support, advice and encouragement as I completed this thesis; particularly Dr. Seamus O'Shea, Dr. Andrew Hakin, Dr. Stephen Wismath, Gerda VanderFluit, Barry Gergel, David Lenz, Trent Takeyasu and Nicole Wilson.

Special thanks are due to Ján Maňuch for his advice and support during this process and to Massih Khorvash for his assistance in the programming of the path algorithm.

Finally, I thank my parents David and Esther Thomas, my brothers: Wayne and his wife Christie, Jeffrey and his wife Ashlee, Loren, Gregory and Shaun for their continual love and support.

Contents

List of Figures

Notations and Definitions

List of Notations

(C_α)	first carbon atom in each peptide group	3,4
(ϕ)	rotation angle about the $C_\alpha - N$ *bond*	4,5
(ψ)	rotation angle about the $C_\alpha - C$ *bond*	4,5
(α)	secondary structure helix designation	4,5,6
(β)	secondary structure sheet, barrel and turn designation	4,5,6
(γ)	secondary structure turn designation	5
(χ)	order preserving bijection (1-1 mapping)	13

Definitions

Graph

An undirected graph is an ordered pair $G := (V, E)$ where V is a set of vertices, and E is a set of pairs (u, v) of distinct vertices.

Walk

$v_1, v_2, \ldots, v_k$ is a walk if for each $i = 1..k-1$, (v_i, v_{i+1}) is an edge in the graph.

Path

A path is a self-avoiding walk. A path is a walk in which no vertex is visited more than once.

CRMS

CRMS is a Euclidean distance measure of the deviation of a lattice model to the backbone of a three-dimensional protein structure. CRMS is calculated by:

$$CRMS = \sqrt{\frac{\sum_{p \in P} d_2(p, f(p))}{n}} \tag{1}$$

DRMS

DRMS measures how well the overall structure is preserved. DRMS is calculated by:

$$DRMS = \sqrt{\frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (d(p_i, p_j) - d(q_i, q_j))^2}{\frac{n(n-1)}{2}}} \tag{2}$$

Bipartite Graph

A graph, $G := (V, E)$, is bipartite if its vertices can be partitioned into two disjoint subsets U and V such that each edge connects a vertex from U to a vertex from V.

Contact Map

A contact map is a graph, $G := (V,E)$, with vertex set of integers $(V = 1,\ldots,N)$ and an edge set E. Vertices are linearly ordered 1 through N. Edges between vertices are defined when the distance between pairs of vertices is below a prescribed threshold.

Maximum Weight Matching

Given a graph, $G := (V,E)$, $M \subseteq E$ is a matching if not two edges in M are incident on the same vertex. Weight of a matching M is the sum weights of the edges in M. Maximum weight matching is the matching with the largest weight.

Neighbors

$N(u)$ is the set of all vertices $v \neq u$ such that (v,u) is and edge in the graph.

Chapter 1

Introduction

In 1928, Scottish scientist Alexander Fleming noticed a halo of growth inhibition on a *Staphylococcus* plate culture. Isolating this region and purifying a culture he discovered it to be *Penicillium notatum* which he called penicillin. More than a decade later, penicillin was first used on a human patient with successful results thereby solidifying its prominence in history.

The above is an example of the efforts related to drug discovery and design. Although not confined to the past century, there have been significant advancements during the past 100 years. Advancements in chemistry, biochemistry, pharmacology, genetics and microbiology have each played a role in this process.

Crucial to the process of drug design is an understanding of the underlying issues related to an illness or disease. Illnesses or diseases may be caused by either external factors such as infectious agents and microscopic organisms which may include bacteria, viruses and protozoans or by internal causes that may include hereditary abnormalities and congenital diseases. A large number of ailments are caused by a combination of both internal and external factors. Understanding how these factors interact and by what processes is of great importance.

Advances in molecular genetics have revealed that many diseases stem from specific protein defects. Included in the list of these diseases are muscular dystrophy, cystic fibrosis, malaria, anthrax, HIV, herpes simplex and herpes complex, Alzheimer's, Huntington's, Lou Gehrig's disease (ALS), Parkinson's, many cancers and cancer-related disorders. Also included are the prion diseases such as Alpers syndrome, Kuru, Fatal Familial Insomnia (FFI), Gerstmann-Straussler-Scheinker syndrome (GSS) and Creutzfeld-Jacob disease (CJD) which is the human counterpart of bovine spongiform encephalopathy (BSE) often referred to as Mad Cow disease. Therefore being able to understand the precise nature of the underpinning factors and design drugs specific to these factors is of great importance.

To date a substantial amount of work has focused on the study of proteins. This includes an analysis of the proteins structure from a linear arrangement of their base molecules to that of multiple domains of three-dimensional structure. It is known that a protein sequence will fold to a unique native conformation (minimal free energy state) very quickly and that the three-dimensional structure of the native conformation will dictate the function of the protein [1, 2]. Substantial advances have been made, however the exact mechanism by which a protein folds to a specific conformation is not clear.

1.1 Proteins - What are they?

Proteins are vital biological molecules that underpin every aspect of biological activity, through the expression of the genetic information contained in a cell's DNA [3, 4]. This becomes particularly important in areas such as medicine where protein structure and function have great effect. In fact, molecular research over the years has revealed that numerous diseases stem from specific protein defects, such as cystic fibrosis, influenza, sickle cell anemia, HIV and cancer [4].

Proteins are the most abundant macromolecule in a cell each performing their own specific functions directly related to each individual protein's native conformation. This conformation is the minimum energy state, also referred to as the ground state [5]. Subsequently an

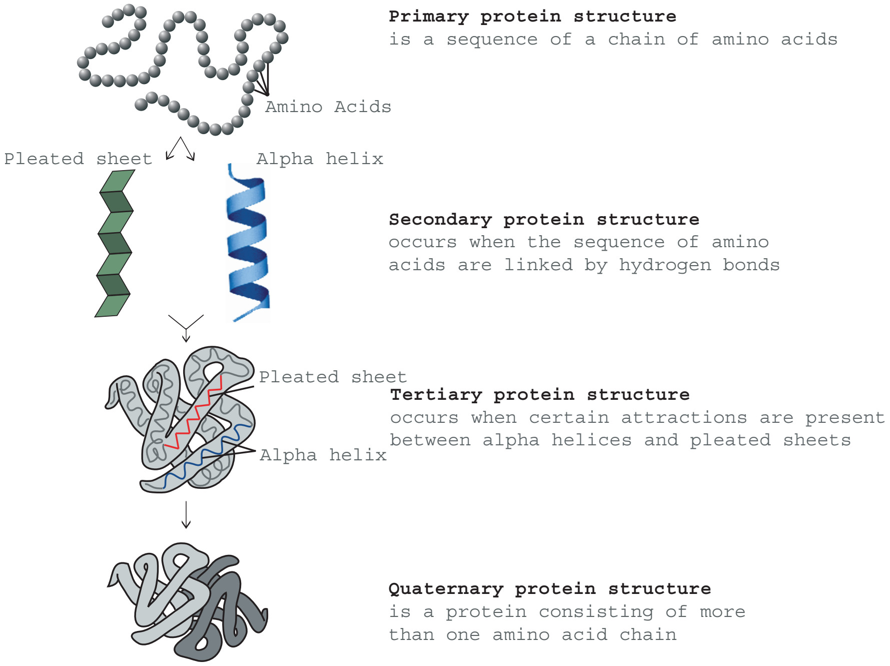
Primary protein structure
is a sequence of a chain of amino acids
Amino Acids
Pleated sheet
Alpha helix
Secondary protein structure
occurs when the sequence of amino
acids are linked by hydrogen bonds
Pleated sheet
Tertiary protein structure
occurs when certain attractions are present
between alpha helices and pleated sheets
Alpha helix
Quaternary protein structure
is a protein consisting of more
than one amino acid chain

Figure 1.1: Four level structural hierarchy of a protein.

understanding of the relationship between structure and function is crucial.

The structure of a protein is organized in a hierarchy of four levels [Figure 1.1]: primary, secondary, tertiary and quaternary structure. Each level is uniquely important and when combined yields an all-encompassing view of the protein.

The primary structure consists of a two-dimensional linear arrangement of amino acid residues assembled together in a polypeptide chain [4]. Amino acids are the building blocks of proteins, each block a translation of a 3-codon segment of messenger RNA. A codon (3-codon) is a grouping of three RNA base pairs that encode one of the 20 possible amino acids. A polypeptide chain is a sequence of different amino acids held together by covalent peptide bonds. The peptide group (defined by the three atoms $C_\alpha - C - N$) of each amino acid forms the backbone of the peptide chain. C_α refers to the first carbon in each amino acids peptide group that attaches to the functional group (responsible for the characteristic

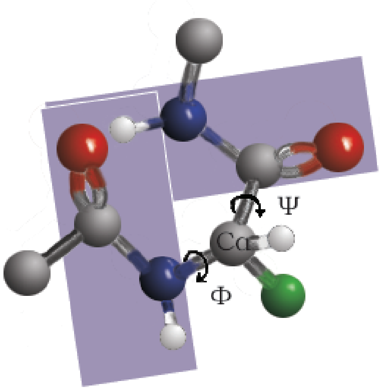

Figure 1.2: Polypeptide backbone demonstrating the location and rotational direction of a peptide groups torsion angles: ϕ and ψ

chemical reaction). A peptide groups structure is both rigid and planar as demonstrated by Linus Pauling in the 1930s and 1940s [6]. This finding lead to the identification of torsion angles that specify a polypeptides backbone conformation. Torsion angles [Figure 1.2] are angles of rotation about the $C_{\alpha} - N\ bond(\phi)$ and the $C_{\alpha} - C\ bond(\psi)$. These angles are sterically constrained thereby limiting the backbones conformational range [6]. The degree of the ϕ and ψ angles characterize the secondary structures in the three-dimensional protein structure.

Primary structure leads to the secondary structure. The secondary structure is characterized by the local conformation of the polypeptide chain or the spatial arrangement of amino acid residues [4]. There are three basic units of secondary structure: the α helix, the β strand and turns. Each of these units is held together with hydrogen bonds. The right-handed α helix [Figure 1.3] is probably the best known and most identifiable unit of secondary structure, accounting for over 30 percent of all residues in globular proteins [4]. The values of ϕ and ψ in the α helix allow the backbone atoms to pack close together with few unfavourable contacts in order to form hydrogen bonds. There is another more rare helical structure associated with the α helix known as the 3_{10} helix. This 3_{10} helix is a smaller more compact

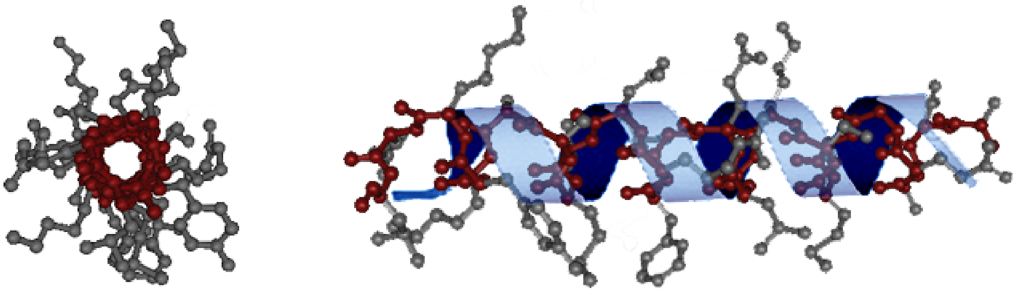

Figure 1.3: Left figure displays a ball and stick model of a α helix as viewed from the top. The right figure a side view with ribbon representation included.

structure found in proteins where the regular α helix is distorted.

The β strand like the α helix is a helical structure. However, its arrangement is extremely elongated compared to the α helix with two residues per turn compared to the 3.6 residues per turn of the α helix [4]. β strands [Figure 1.4] are harder to recognize and individually not stable primarily due to a lack of local stabilizing interactions. Stability increases when a number of strands connect using hydrogen bonds forming a β sheet. A sheet can be either parallel or anti-parallel depending on the arrangement of the β strands in the sheet. In a number of proteins β strands associate spectacularly into extensive curved sheets known as β barrels [4].

Turns have the universal role of enabling the polypeptide to change direction and in some cases to reverse on itself [4]. Studies have revealed as many as 10 different conformations on the basis of number of residues making the turn and the degrees of ϕ and ψ associated with the central residues. Common turns include the γ turn, characterized by the residue in the middle of the turn not participating in hydrogen bonding, and the β turn in which the two middle residues are never involved in hydrogen bonding [4].

These structural units, although common, are not found in all proteins together. In fact,

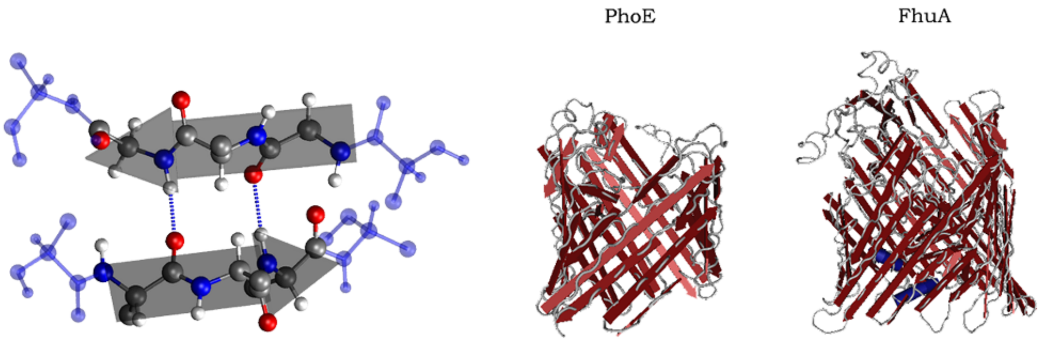

Figure 1.4: Left figure displays a β sheet composed of single β strands [7]. Sheets can create more complex structures such as β barrels as seen in the right figure.

proteins are divided into structural classes based on the composition of their secondary structures. There are four different structural classes of proteins characterized by the inclusion or exclusion of α helices and β strands. These four classes are: all-α, all-β, $(\alpha + \beta)$ and α/β [8, 9].

The tertiary structure represents the overall three-dimensional structure, or the fold, of the polypeptide chain. It is the spatial arrangement of the protein formed by specific stabilizing interactions. A stable tertiary fold relies on a number of different interactions characterized by their relative strength and frequency. These interactions include: disulfide bridges, ion pair interactions, hydrogen bonds, van der Waal forces and hydrophobic interactions.

In 1990, Dill [10] proposed that an understanding of the dominant forces involved in the structure of proteins would further our understanding of proteins. Basing his findings on the criteria that the dominant force must explain why a folded state is advantageous and adheres to reversibility, Dill concluded that hydrophobic interactions were the dominant force. Interactions such as van der Waals and hydrogen bonds were crucial to preventing the protein from denaturing (unfolding), however, they lacked an explanation for the folded states advantage [10]. Though the dominant force, hydrophobic interactions, cannot be solely responsible for determining a proteins native structure, it can ‘select’ a relatively small number of conformations from which the native structure is determined by the balance of all interactions [10].

For larger proteins the tertiary structure may be organized around more than one structural unit. Each unit is called a domain, which is defined as a region or regions of a polypeptide that fold independently and possess a hydrophobic core [11].

Quaternary structure explains the existence of more than one polypeptide chain found in many proteins. It is similar to tertiary structure in that it utilizes the same interaction forces as tertiary structure. It differs, however, in that the interactions occur between one or more polypeptide chain often referred to as subunits [4].

Experimentally we are able to denature proteins and study their primary structure. Using X-ray crystallography we can study the three-dimensional structure of the protein [12]. What we are not able to establish is exactly what it is that causes the protein to fold and why the protein folds into the conformation it does.

The protein folding problem seeks to determine the native conformation of a protein given its linear arrangement. The inverse protein folding problem is the problem of protein design. Given a specific three-dimensional conformation the question is to determine a linear sequence that will create this unique structure. The inverse protein folding problem has great potential as a tool in drug design as not only could it provide a first approximation of a primary amino acid sequence for a target protein [5]; it could also help in the engineering of known protein secondary structures with the potential for drug delivery [13].

1.2 Protein Folding

Protein folding is a rapid biochemical reaction. The three-dimensional structure of a protein dictates the function of the protein. The computational difficulty stems from the fact that there are an exponentially large number of possible structural conformations and numerous criteria which may determine correct folds [14]. The difficulty is compounded further as numerous proteins, each with differing amino acid sequences, have nearly identical structural conformations. The converse is true as well.

Numerous techniques have been used to study protein folding. These include experimental, theoretical, imaging and computational approaches. A recent experimental method [15] involves rapidly triggering the folding of a sample of unfolded proteins, and observing the resulting dynamics. Theoretical methods involve the calculation of protein energy landscapes. The energy landscape of a protein is the variation of its free energy as a function of its conformation, due to the peptide interactions. Imaging techniques involve the use of X-ray crystallography and NMR. These methods are both lengthy and complicated. Finally, there are computational methods that employ simulations of protein folding to determine the native conformation.

An exponentially large number of possible conformations create difficulties in simulations. However, the use of lattice models allow for a finite, yet exponentially large number of possible conformations [16] thereby reducing the search space of all possible conformations. The question is, which is the right lattice model to use, and how do you know? Suitability of lattice models for protein folding is often determined using distance measurements known as CRMS and DRMS and is the object of study in this thesis.

1.3 Inverse Protein Folding

Inverse protein folding seeks to determine a protein's amino acid sequence based on a desired native conformation. A 'good' target sequence is the desired outcome of the inverse protein folding model, and is the one whose native conformation has the lowest accessible free energy and the lowest accessible free energy fold is unique [17]. Determination of a good target sequence is of concern in modern drug design.

Known techniques and models used in the study of protein folding have been incorporated in the search for good target sequences [18, 19, 10, 20, 21, 22, 23, 24, 25, 26, 27, 5, 14, 28, 16, 29, 30, 13].

Current theoretical approaches focus on local and global interactions. Local interaction studies include the intrinsic propensities of residues to form helices and turns [31, 32, 5]. De novo studies of global forces include hydrophobic interactions, conformational freedoms of the sequence chain and steric restrictions imposed by excluded volumes on the sequence chain [5, 17]. Computational models include the use of 'on-lattice' and 'off-lattice' models.

1.4 Protein Analysis

Understanding the relationship between the amino acid sequence and three-dimensional native conformation of a protein is essential to understanding biological processes [25]. The protein folding, and inverse protein folding problems as mentioned above focus on this. Each problem can be formulated as a search for the native conformation, the conformation with the minimal free energy. However, finding the unique native conformation out of the enormous number of possible existing conformations presents huge combinatorial problem in structural and sequence space [25, 33]. This issue was first articulated in the 1960s by Levinthal and has become known as the 'Levinthal paradox' [26]. Amino acid sequences must also satisfy two requirements; a thermodynamic requirement in which the sequence must have a unique folded conformation, that is stable and corresponds to the native structure, and a kinetic requirement whereby the denatured polypeptide chain can fold into the native conformation under the appropriate solution conditions [33].

Even with the simplified models and restrictions imposed on protein folding models, it has been shown that the problem is NP-complete [3]. This implies that it is unlikely a polynomial time algorithm exists to solve the protein folding problem.

Despite the hardness, the search for practical algorithms remains the driving force behind numerous studies [16, 10, 25, 34, 35].

Three methods have been introduced to overcome Levinthal's paradox, and for the demonstration of mechanisms employed in folding [25, 33]. The first method is a computation procedure that utilizes empirical information on protein structures [25]. Protein structures are compared through structural similarity and homology. The second method is based on a global search for the minimum energy state. The energy function can be simplified either by performing a search of an approximate solution or by replacing and modifying the potential energy function itself [25]. Incorporation of mean field theory has been utilized as well. The last method drastically reduces the dimension of the problem through a reduction in structure space [25].

Restricted state models as well as the introduction of lattice models reduce the number of possible conformations from infinite possibilities to one of finite [16]. Studies combining methods two and three have been appearing with greater frequency over the years with positive results.

1.4.1 Structure and Sequence

A protein's biological function is characterized by its three-dimensional structure. Therefore through the comparison of known to unknown proteins we may be able determine possible structural characteristics of the unknown proteins. There are a number of different approaches including sequence alignment, structure alignment and contact map overlap [36, 3, 37, 38].

In 1994, Thompson *et al* [38] introduced CLUSTAL-W an algorithm for progressive multiple sequence alignment. Multiple sequence alignment is a technique used to: (i) identify highly conserved patterns in protein families; (ii) identify the homology between new and existing sequences, and; (iii) identify the prediction of protein secondary and tertiary structure. It is a modified version of pairwise alignment that detects weak similarities' common to biologically related sequences [39].

Structural alignment compares the three-dimensional structure of a known protein to that of an unknown protein. To date there are a large number of known protein structures listed in the Protein Data Bank [40]. When a new protein is crystallized, structural comparison of the unknown structure to the known structures is undertaken in the hopes of determining a class for the protein, or rather a plausible function for the protein.

In 2004, Kolodny, Koehl and Levitt [37] reported on a comprehensive evaluation they performed on six publicly available structural alignment methods. Methods evaluated represented two general strategies for finding good alignments. Group one represented by

STRUCTAL, LSQMAN and SSM, search for transformations that optimally position the two structures with respect to one another. Group two represented by SSAP, DALI and CE directly search for a good alignment [37]. Scores for good alignments vary from one method to another, therefore, the perceived performance of each method is subjective. Comparison was made using analysis of geometric match measures and through comparison of receiver operating cost (ROC) curves based on a CATH gold standard.

They developed their own method for finding the best alignment, however, they did conclude that the evaluation of structural alignment methods should be done through alignment comparison rather than the ranking of methods and ROC curves. ROC curves are of limited value because the disadvantages outweigh the advantages. These curves may seem reasonable due to the use of a gold standard however this standard is based on a classification rather than a direct reflection of the similarity measure. In this case, the classification is converted to binary, thereby allowing only two levels of similarity. This introduces noise whereas a scale of similarity is far richer. ROC curves also do not detect if one method does better than another.

Building on the methods of structural and sequence alignment is a method known as a contact map overlap. A contact map is a graph whose vertices are protein residues and whose vertex set is linearly ordered to represent the backbone of a polypeptide chain. Edges between vertices are defined when the distance between two residues in the native state is below a prescribed threshold.

The contact map overlap problem (CMO), seeks to evaluate the similarity of the native state of two proteins based on the similarity of their contact maps. Given two folded proteins, an alignment is sought which specifies residues considered to be equivalent in the two proteins. The goal of CMO is to find the alignment that highlights the largest set of common contacts.

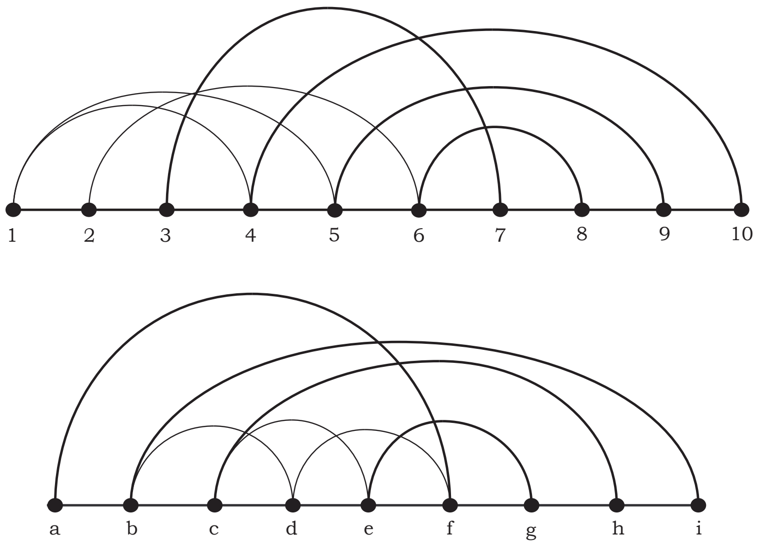

Figure 1.5: Given two subsets, $S = \{3,4,5,6,7,8,9,10\}$ and $S\prime = \{a,b,c,d,e,f,g,h,i\}$, an alignment which preserves the greatest number of similar contacts (edges) is (3,a), (4,b), (5,c), (6,e), (7,f), (8,g), (9,h), (10,i); the maximum overlap between the two contact maps is 4 [3]

The value of an alignment is determined based on the number of residue pairs in contact in one protein and aligned with residue pairs also in contact in the second.

CMO is the following optimization problem: Given two contact maps $([n],E)$ *and* $([m],E\prime)$, find an order-preserving bijection (1-1 mapping), χ, between two subsets $S \subseteq \{1,\ldots,n\}$ and $S\prime \subseteq \{1,\ldots,m\}$ with $\mid S \mid = \mid S\prime \mid$, such that the number of edges is preserved by the bijection, that is $\mid \{(u,v) \in E : u,v \in S, [\chi(u),\chi(v)] \in E\prime\} \mid$, is as large as possible [3]. An example can be found in Figure 1.5.

As stated in [3], it is intuitively clear and widely accepted that contact maps of proteins are far from arbitrary collections of edges, since they have a specialized structure reflecting the geometry of proteins. Therefore, to capture a proteins structure a special class of contact maps is required.

In 2002, Goldman [3] presented polynomial-time algorithms for computing the contact overlap of special classes of two contact maps. Contact maps are derived from self-avoiding walks on a 2-dimensional lattice, and the special class of contact maps are stack, queue or staircase, as defined next. Define a stack to be a contact map ([*n*], *E*) such that if (*i*,*j*), $(k,l) \in E$ then either: (a) one of the intervals (*i*,*j*) or (*k*,*l*) contains the other, (b) they are disjoint, or (c) they overlap at an endpoint. A queue is defined as a contact map ([*n*],*E*) such that if (*i*,*j*), $(k,l) \in E$ then neither of the intervals $[i, j]$ and $[k,l]$ contains the other unless they share an endpoint. A staircase is a queue consisting of sets of intervals such that all intervals in a set have more than one point intersecting. They proposed a 3-approximation algorithm with $O(n^6)$ running time to compute the maximum contact overlap of two contact maps, one of which is a staircase. A recursive dynamic program is used to compute the contact map overlap. The contact map overlap problem is NP-hard in general as established by Goldman [3]. For other NP results see the paper by Hart and Newman [41] and Berger and Leighton [42].

1.4.2 Energetics

The energy based approach to protein structure prediction is the study of the kinetics of the system in which the goal is to determine the structure with minimum free-energy [43]. Refolding of denatured protein into a biologically active form takes an average of 1 msec to 100 seconds [43]. This time-frame is too short for an exhaustive random search for the minimum free-energy structure to be conducted [43]. In 1992, Leopold *et al* [44] introduced a model of a convergent kinetic pathway, or ‘folding funnel’ which explained the use of a broader landscape whereby large numbers of pathways and intermediates are utilized during a fold, rather than the use of a single mechanism. Therefore to understand protein folding, and in particular the pathway involved, it is necessary to determine the structure and energetics of not only the initial unfolded state but all intermediate states, the final folded states and the transition states occurring along this pathway [4].

A selection of the methods, used in the past decade, incorporate folding based on internal interactions, molecular dynamics, Monte Carlo, self consistent field theory and self consistent mean field theory [19, 10, 15, 25, 45, 28, 46, 33]. Monte Carlo simulations were used by Gutin *et al* [15] in 1995 in their fast fold method and by Šali *et al* [34] in 1994 in their study of the kinetics of protein folding.

Self-consistent-field-optimization (SCF) [47] was utilized by Reva *et al* [28, 46] first in 1996 and later in 1998. In the first study SCF-optimization was used to minimize the protein chain molecules 'energy'. Later in 1998 SCF-optimization was used on a combined pseudoenergy energy function which includes both the potential energy and geometric constraint terms. From this Reva *et al* [46] were able to reproduce off-lattice structures with minimal errors in geometry and energetics.

The Hydrophobic-Polar (HP) model was introduced by Dill [19] as a method of reducing the complexity of the protein folding problem. Amino acid residues of the primary sequence are characterized as either hydrophobic (H) or polar (P) monomers. A fold relies only on the interactions of these classes, thereby decreasing the complexity of the problem. Since contact between any two non-consecutive hydrophobic monomers reduces the free energy of the resulting molecule, a protein is assumed to reach the native state when the number of contacts between non-consecutive hydrophobic monomers is maximized.

The last of the three methods deals with the reduction in conformational space. Models in this category focus on reduction through restriction of possible conformations. This can be done through the incorporation of lattice models, off-lattice models [14] and by restricting [48] the proteins secondary structures from the native conformation.

1.4.3 Lattice Models

For our purpose a lattice is a two- or three-dimensional array of regularly spaced points. Lattice models have been used extensively in protein folding for a number of years now, however it may be conjectured that lattice underpinnings in folding came as the result of the pioneering work of Levitt and Warshel who in the 1970s marked the beginning of physically based models in the studies of protein folding [49]. Advantages associated with lattice models are clear. They allow for an efficient sampling of conformational space, yet restrict the overall size of conformation space.

There are two types of lattice model simulations, with two distinct objectives. Simulations geared towards understanding the physics of protein folding, and those geared towards realistic folding of proteins, and are parameterized using real proteins as templates [49].

A key feature of the type one lattice model is its simplicity [49]. These models are not designed for real proteins, thus tests on these models have been limited to the studies of general features of protein folding [49]. Nevertheless, a good deal can be learned from the studies. These include the work of Dill *et al* [20] and Duan *et al* [49] who emphasized the importance of hydrophobic interactions, or those of Bryngelson *et al* [43] who postulated that proteins have a funnel-like energy landscape with minimal “frustration”. Other studies include the works of Gutin *et al* [15], Koehl *et al* [25], Park and Levitt [14], Sanchez *et al* [33] and Yue *et al* [17, 50].

Lattice models of type two are more complex. Because the residue-level representations are applied to real proteins, that have a large number of energy minima as compared to the simplified type one models, their energy surface can no longer be described exactly [49]. They can however be used for exhaustive search of protein sequences, and do impose a restriction on the number of possible conformations due to pair-wise interactions between nearest neighbors [49]. Given the simplicity and success of these models, a number of studies have been conducted using these models, including studies by Park and Levitt [14], Rykunov and Reva [16] and Mead *et al* [5] who not only use these models, but also test to see which lattice is the best lattice for protein folding studies.

In 1995, Park and Levitt [14] developed an algorithm that could be applied to any lattice model in hopes of determining the relationship of a models "complexity" (number of conformations) to its accuracy (preservation of protein geometry). This algorithm allows fitting of both on- and off-lattice models to the X-ray coordinates of a large database of proteins, analyzing the relationship in both a global and local sense. They discovered that less complex models that take into account the uneven distribution of residue conformations can be as accurate as models of higher complexity. Therefore, they argued that low complexity off-lattice models would be more appropriate for protein structure prediction than higher complexity lattice models [14]. With this in mind they presented a set of optimized 4-state models that could reproduce the backbone structure accurately in both a global and local sense.

Experimental runs were performed on a set of lattice models of varying complexity and deviations as determined by a coordinate root mean square (CRMS) were recorded. Analysis demonstrated that increasing model complexity above a certain point would yield little improvement in the accuracy. They found the accuracy of fit to follow a simple law [14]:

$$(CRMS) \propto (Complexity)^{-1/2}$$

Accuracy measures observed include: (1) the preservation of native contacts between residues, and (2) the preservation of a large proportion of the native secondary structure.

In general, higher complexity (more conformations allowed) equates to greater preservation. However, in actuality preservation of the secondary structures varied substantially because of the composition of structures present. In some low-complexity models one of the secondary structures was preserved more than the others and vice versa with more complex models [14]. Low-complexity models were able to reproduce the important characteristics of tertiary structure, native contacts, secondary structure and CRMS deviation, giving Park and Levitt a basis for their optimized 4-state models.

That same year, Rykunov *et al* [16] proposed a novel algorithm based on dynamic programming that could be applied to any three-dimensional lattice producing lattice models having minimal RMS deviations from the actual folds. Their model sought to minimize the error function (difference between the coordinates of the protein chain and the lattice model) while satisfying the conditions of chain connectivity and self-avoidance [16]. The algorithm builds a self-avoiding walk by testing a large number of lattice-protein orientations and selecting the best one based on that orientations error function. The model with the minimum error function is best.

Since then, Reva *et al* [51] have revisited the problem twice. First, with a modification to lattice and algorithm for the inclusion of sidechains and then with the incorporation of self-consistent mean field theory which guaranteed the absence of self-crossings in the model chain thereby producing self-avoiding walks [28]. However, due to the complexity of the models, the largest protein that the algorithms could handle was limited to approximately 247 residues.

In 2005, building on the work of Park and Levitt [14], Mead *et al* [5] proposed a novel algorithm to test for an ideal lattice that could be used in the study of inverse protein folding. In this thesis we study the problem of determining ideal lattices [14], those that best approximate the rigid protein backbone when residues are placed on the vertices. This problem is of interest because it arises in the context of protein folding and inverse protein folding algorithms that use lattice models. We focus on the simple cubic lattice and on a coordinate root mean square (CRMS) distance measure for determining suitability.

Two algorithms are presented: the first algorithm computes an optimal non-self-avoiding walk (walk henceforth) to embed a protein to a lattice. Dynamic programming is used to determine the best walk. Branch and bound techniques are utilized to help prune a large number of possible invalid walks and considerably speed up the algorithm. The algorithm is fast and as yet does not suffer from any size restrictions. Tests have been performed on approximately 3704 protein structures (obtained from PDB), some containing greater than 1000 residues.

The second algorithm computes an approximate self-avoiding walk of a protein. The problem of computing the self-avoiding walk that minimizes CRMS deviation was recently shown to be NP-complete [35]. Once again the protein is mapped to the lattice and an approximate path is determined using dynamic programming. Branch and bound techniques are investigated so as to prune the vast number of possible non optimal conformations. The condition of self-avoidance makes the process considerably more difficult. Results are compared to those of the walk algorithm.

The remainder of the thesis is organized as follows: it contains three chapters, repectively on the algorithms designed, experimentation and discussion, and open problems discovered through our research. In the first chapter we present algorithms for the determination of ideal lattices [14] (those that best approximate the rigid protein backbone when residues are placed on the vertices). The second chapter addresses our findings and the last chapter, new questions that are open for further study.

Chapter 2

Algorithms

It is a fact that the quality of the protein structure predicted depends on the lattice model used. What constitutes a good lattice model is still being debated. With this in mind, Park and Levitt [14] have indicated that there are three basic criteria that hold merit in determination of a good model: (1) the preservation of a protein's native contacts; (2) a measure such as CRMS to judge a model's accuracy, and; (3) a large proportion of a protein's native secondary structure should be preserved.

The build-up algorithm developed in 1995 by Park and Levitt [14] can be applied to any lattice model. As mentioned previously, the algorithm allows for the fitting of both on- and off-lattice models to the X-ray coordinates of the C_α of a large database of proteins. The algorithm is brute force and does not guarantee that a globally optimal fit will be found. Because of the prohibitive runtime of the algorithm on larger structures, shorter peptides were used to test the quality of fit. One hundred segments, 12-residues long, were randomly selected and all the possible conformations were enumerated using an optimized 4-state model that found an optimal fit. These segments were then used for the usual build up procedure with each of the models and compared to the optimal. Structures much larger than 12-residues long were found to have too many conformations therefore there was no way of knowing how close they were to the exact optimum [14].

The algorithm proposed in 1995 by Rykunov *et al* [16] mentioned previously is based on dynamic programming, can be applied to any lattice model and is able to find self-avoiding walks with minimum error function. Unfortunately, the algorithm does not guarantee self-avoidance in all the runs. This is because the basic design of their algorithm is that of a walk. To address this they incorporated a couple of methods which help avoid self-crossings. Method one employed the addition of an approximated 'steric repulsion' of links into their error function. Repulsion only minimally improved avoidance, therefore Rykunov *et al* [16] proposed a second method. In this method, they searched through multiple ($9^3 = 729$) lattice protein orientations in an effort to find the best self-avoiding model. Orientations searched corresponded to rotations by $0°, \ldots, 80°$ over each of three Euler angles. Self-avoiding walks were found on proteins of length no greater than 107 residues.

The two algorithms in this thesis are also based on dynamic programming and like Rykunov *et al* [16], can in theory be applied to any lattice to determine lattice models having minimal CRMS deviations from the actual folds. Unlike Rykunov *et al* [16] these algorithms incorporate branch and bound techniques to prune invalid conformations thereby increasing the efficiency. Further more our objective function is considerably simpler, and relies on the study of Mead *et al* [5]. The first algorithm, does not guarantee self-avoidance and determines the optimal fit (one with minimum CRMS deviation). In the second algorithm, the path guarantees self-avoidance, a problem shown to be NP-complete [35] (it is highly unlikely that a polynomial time algorithm for the problem exists). We use dynamic programming with branch and bound, but due to combinatorial explosion, we are unable to compute optimal solutions to many of the protein samples. Given the hardness of the problem, we evaluate the performance ratio of a greedy algorithm as there are no known approximation algorithms. Empirically, we establish that the greedy algorithm computes paths that have CRMS value 1.56 times that of the optimal CRMS on average. A theoretical analysis of the greedy remains an open and interesting problem.

2.1 Dynamic Programming

Dynamic programming is a recursive technique, similar to the divide and conquer method that solves problems by combining the solutions with subproblems [52]. Dynamic programming, however, differs from the divide and conquer method in that it solves every subproblem just once and saves its answer in a table, thereby avoiding repetitive calculations every time a subproblem arises [52].

Dynamic programming has a rich history. It has been successfully applied to problems such as assembly line scheduling, matrix chain multiplication, contact map overlap and global sequence alignment to name a few [52].

Dynamic programming can be summarized as follows:

1. Establish a recursive property based on the 'principle of optimality', that describes the optimal solution to the problem in terms of the optimal solution to the subproblems.
2. Solve the problems in a bottom up fashion, i.e., compute optimal solutions to the smaller sized problems first, and use these solutions to compute the solution to the larger problem.

2.2 Branch and Bound

Many algorithms involve exhaustive procedures in that they choose between a number of alternatives during each iteration. Depending on the branching factor of the search tree the total number of possible alternatives can be huge thereby consuming large amounts of time and memory. Being able to prune the branches of the search tree that do not lead to the solution would be advantageous. This is precisely what branch and bound attempts to accomplish.

Branch and bound is implemented via the introduction of a test at each alternative. If the process passes the test the alternative is included for subsequent testing in following iterations. If the process does not pass the test then this alternative is skipped. A common type of test involves the use of upper and lower bounds. Consider a minimization problem. If the value of the current partial solution plus 'some' lower bound on the value of the remaining subproblem is greater than an upper bound on the value of the optimal solution to the problem, then the current partial solution need not be explored further. Hence the node associated with the current partial solution in the search tree can be fathomed.

2.3 Walk

Our study examines the 3-D structure of protein backbones, adhering to the characteristics verified by Mead *et al*, [5], (i.e., the distance between two consecutive amino acid residues is 1 unit, and the distance between any two non-consecutive amino acid residues is at least 1 unit). The protein chain we denote P and the number of amino acid residues in P by n. Also given is a simple cubic lattice L.

Let $f : P \to L$ be a mapping such that consecutive residues in P are mapped to consecutive vertices in L. Our objective is to find a mapping f such that the CRMS-deviation of the model from the actual 3-D structure is minimized. CRMS deviation is given by: $\sqrt{\frac{\sum_{p \in P} d_2(p,f(p))}{n}}$.[1] The minimum CRMS value referred to in this section we obtain by min-

[1] d_2 is the square of the distance.

imizing the term $\sum_{p \in P} d_2(p, f(p))$, is denoted c^*. Note, any two residues p_i, p_j may map to the same vertex $v \in L$. Therefore, f does not guarantee a self-avoiding walk.

We denote the total number of vertices in the lattice by $|L|$ and neighbors of a given vertex $v \in L$ we denote $N(v)$.

2.3.1 Dynamic Programming Algorithm

Let C be a $|L| \times n$ matrix, whose entry $C[v, j]$ is the cost of the optimal solution that maps the first j residues and ends in v. The walk fixes protein residues $p_1, \ldots, p_j$ on L, subject to the constraint that protein p_1 is mapped to a fixed vertex $v_1 \in L$, and p_j is mapped to vertex v. By definition, c^*=$min_{v \in L}\{C[v, n]\}$. Entries in C can be computed using dynamic programming in $O(|L|n)$ time, as shown below.

All entries in C are initialized to $+\infty$, followed by assigning: $C[v, 1]$=$d_2(v_1, p_1)$. Subsequent entries are computed column by column according to the following rule:

$$C[v, j] = min_{u \in N(v)}\{C[u, j-1] + d_2(v, p_j)\} \tag{2.1}$$

Efficiency can be improved by incorporation of a candidate list. If for all vertices $u \in N(v)$, $C[u, j-1]$ is $+\infty$, the evaluation of equation 2.1 for entry $C[v, j]$ is ignored. A candidate list maintains elements $C[v, j-1]$ from iteration $j-1$ with a finite value. We update $C[v, j]$ only if $v \in N(u)$, and u is in the candidate list, and $C[u, j-1] + d(v, p_j) < C[v, j]$.

Theorem 1 *The dynamic programming algorithm correctly computes the walk, starting at a fixed vertex v_1, with minimum CRMS value in O($|L|n$) time and O($|L|$) space.*

Proof Correctness proof is by induction.

- *Base Case, n=1:* The algorithm maps p_1 to v_1 with minimal CRMS. Therefore the base case holds.
- *Induction Hypothesis, n=k-1:* The algorithm correctly computes entries $C[u, k-1]$, i.e. the cheapest cost walk starting at v_1 and ending in u that maps residues $p_1, \dots, p_{k-1}$.
- *Induction Step, n=k:* By definition a walk that fixes protein residues $p_1, \dots, p_{k-1}$ to L will map p_{k-1} to $u \in N(v)$. If we assume an optimal solution where p_k maps to v, then p_{k-1} will map to $u \in N(v)$. If the CRMS value of $C[v, k-1]$ is not minimal for the mapping of $p_1, \dots, p_{k-1}$ beginning at v_1 then there is a contradiction. Therefore, by induction we have shown equation 2.1 computes correctly $C[v, p_k]$.

Running time of the walk algorithm is proportional to the entries in C, $O(|L|n)$. Values in C are computed by column via information stored in the $j-1$ column, therefore, only two columns of C need to be stored. Thus $O(|L|)$ space is sufficient. ∎

Next we show how we can further speed up the algorithm using a branch and bound procedure. Utilizing branch and bound techniques we were able to greatly reduce the running time from hours to under 20 minutes on protein structures with approximately 1000 residues on a Celeron 1.3Ghz machine with 256MB of RAM.

Let LB be an array of size n, whose i^{th} entry $LB[i]$ is a lower bound on the optimal solution to the subproblem determined by residues $p_{i+1}, \dots, p_n$ of f, $LB[1] \leq c^*$. Two different runs of the algorithm are performed. In the first run, an upperbound constructed using a greedy algorithm is used. For the second run, UB equals the CRMS value computed in the first run, and CL is initialized with a list of possible vertices where the first protein residue can reside in the optimal solution. This will be expounded on later in this section. A few methods for computing the LB will be illustrated in the next subsection.

If

$$min_{u \in N(v)}\{C[u, j-1] + d_2(v, p_j)\} + LB[j+1] \geq UB \tag{2.2}$$

then in any optimal solution residue p_j is ignored and will not be mapped to vertex v. Subsequently $C[v, p_j]$ is set to $+\infty$, i.e., $C[v, p_j]$ is not added to the candidate list in iteration j.

We also note that in the optimal solution, residue p_1 is assigned to one of the vertices in a circle of size atmost R^2, where $R^2 = UB - LB[1]$, centered around p_1.

Assume that p_1 is mapped to v outside of the circle (radius R^2), then

$$CRMS \geq R^2 + LB[1] \geq UB - LB[1] + LB[1] \geq UB \tag{2.3}$$

Thus choosing some vertex v outside of R^2 is ill-advised.

Remark: Note that this dynamic programming algorithm will not minimize the DRMS value, as the objective function is not “separable”. For example, let us look at a protein of four residues. Let residues be p_1, p_2, p_3, p_4. The DRMS for the subproblem with first three residues is $x = (d(p_1, p_2) - d(v_1, v_2))^2 + (d(p_1, p_3) - d(v_1, v_3))^2 + (d(p_3, p_2) - d(v_3, v_2))^2$, where v_i is the lattice point where residue p_i is placed in the lattice. Note that the DRMS for the four residues cannot be written as $x + y$, where y does not share any term with x. This separation is necessary for the optimality of the solution constructed by the dynamic program.

2.3.2 Upper Bound

For our upper bound value we used the CRMS value of a walk determined by means of a greedy algorithm. Greedy algorithms solve problems by selecting the locally optimal choice at each stage.

In each iteration, the six possible lattice points are mapped to the protein point under consideration. The square of the Euclidean distance for each mapping is calculated and summed with the previous CRMS value. The mapping with the minimal total CRMS value is saved and the algorithm proceeds to the next iteration. This algorithm is very fast and through our experiments has proven to be quite accurate at times.

2.3.3 Lower Bound

The lower bound value is added to the CRMS value of the current iteration and the sum is compared to the upper bound. If the sum of the CRMS value and lower bound to the subproblem remaining is greater than the upper bound the lattice point is pruned. Determination of a good lower bound is crucial. If the lower bound is too small, there may be too many possibilities tested and the algorithm will suffer. Therefore, finding a tight lower bound (as close to the optimal solution) as possible is important.

Recall that the i^{th} entry in the lower bound array is a lower bound on the CRMS value of the optimal solution to the subproblem given by residues $p_{i+1}, \ldots, p_n$. We calculate a lower bound array based on different distance matrices. These ideas are from Mead *et al* [5]. The first distance array is created by calculating the minimum distance of the protein point to its closest lattice point. The second distance array stores in positions i and $i+1$ (for odd i)

the smallest sum of distances that map residues p_i, p_{i+1} to an edge in the lattice.

The lower bound is computed based on the parity of the subproblem from the two previous arrays. If the subproblem has even number of residues then the lower bound is obtained by summing up the (respective) values in the second array. If the number of residues is odd then for the first residue we use the value from the first array, and for the remaining residues we sum up the (respective) values in the second array.

2.3.4 SC Lattice

We have tested our algorithm on simple cubic (SC) lattice. Other promising lattices include the face centered cubic (FCC) lattice and the extended face centered cubic (eFCC) lattice [5]. The walk algorithm can be modified to compute optimal solutions for different lattices as well, albeit with additional programming effort. Our choice of SC lattice was based on the previous studies of Park *et al* [14], Mead *et al* [5] and Rykunov *et al* [16] who determined that lattices having uniform edges of length 3.8 Å, periodic regular structure, and predominant angles of 90° and 120° degrees would represent protein structure relatively closely.

2.4 Path

The algorithm for computing paths with minimum CRMS value, as with the walk algorithm, utilizes dynamic programming and branch and bound techniques. The algorithm is in many respects, near identical to the walk algorithm, however, it does include some important functional changes. Recall that a path differs from a walk, in that the mapping must now be self-avoiding. No vertex is used more than once in each path. This simple restriction makes the resulting problem NP-complete [35].

The path algorithm, like the walk algorithm, seeks a minimal CRMS deviation mapping of protein structure. We initialize the path by mapping the first protein residue p_1 to a fixed vertex v_1. Later, we consider different starting vertices $v \in L$ for v_1.

Let T_k be the set of all paths of length k, stored in a candidate list array, denoted CL. We obtain paths of length $k+1$ from T_k in iteration $k+1$, as well as the associated CRMS values. Note that the total number of possible paths of length k is exponential in k.

Branch and bound can be applied to the algorithm to improve the efficiency. Upper bound (UB) is determined using a greedy path algorithm. Lower bound (LB) may be determined via three methods described below.

2.4.1 Upper Bound

Taking a page from the walk algorithm, a greedy algorithm for the path has been used to determine the upper bound. The greedy algorithm for the path is very similar to that of the greedy algorithm for the walk except the check for self-avoidance. No lattice vertex can be visited more than once, therefore, a test must be performed during each iteration. A hash table is used to store each of the vertices added for quick access later.

During each iteration, there are six possible vertices that can be used to extend the path. Each vertex is tested to see if it has been previously visited. Vertices visited previously are ignored. All remaining vertices are tested for extension of the path. The extension with the minimum CRMS value is added to the path and the vertex is added to the hash table, the algorithm proceeds to the next iteration. Vertex coordinates make up the key for the hash table. This algorithm is very fast and is somewhat accurate as observed empirically. Note that this algorithm need not necessarily find a path that maps all the residues in the protein.

2.4.2 Lower Bound

Let B_i be a bipartite graph whose vertices are the vertices in L, and the residues $p_i, \ldots, p_n$ in protein P. All the edges are of type (v, p_j) where $v \in L$ and $p_{j \geq i} \in P$. The weight on edge (v, p_j) is $d_2(v, p_j)$. The number of edges in B are $|L|(n-i+1)$. A minimum weight perfect matching in B_i is a lower bound on the subproblem determined by residues $p_i, \ldots, p_n$.

A good lower bound, LB may also be found by computing the minimum cost (optimal solution) walk, W_i, to a subproblem determined by residues $p_i, \ldots, p_n$.

Let D be a distance array, such that the i^{th} entry in D is the distance of the closest lattice vertex v to protein residue p_i. Let DD be a double distance array, which stores the smallest distance that maps residues p_i and p_{i+1} to an edge in the lattice. Subject to the parity of i a lower bound, LB may be determined, such that $LB[i]$ is the cost of mapping $p_{i+1}, \ldots, p_n$. $LB[i]$ is determined via one of two equations dependent on (n) *mod* 2. For values equal to 0:

$$LB[i] = LB[i+1] + D[i+1] \tag{2.4}$$

else

$$LB[i] = LB[i+2] + DD[i+1] \tag{2.5}$$

At present, the lower bound method, described in the previous paragraph, is in use in both the walk and the path algorithms. The second method is a good choice for lower bound calculations, however, as currently coded it is inefficient to use, as we require $n-1$ calls. Determination of $n-1$ walks especially if n is large takes a substantial amount of time, making this method infeasible. Method one has not been evaluated for the path algorithm.

2.4.3 Path Algorithm

Each iteration of the path algorithm modifies a hash table and creates a new candidate list. The hash table stores the CRMS value, the current path, the nodes visited and the length of the path. The candidate list is created for each next possible vertex, that could be used to extend the current path, without violating the condition of self-avoidance. These vertices are tested and pruned using branch and bound limiting the size of the candidate list in the next iteration. Unfortunately, due to the nature of the problem even with pruning there are far too many possibilities to be searched efficiently.

During each iteration, the path algorithm can add at most six vertices to the candidate list for each vertex in the candidate list. To reduce the combinatorial explosion a new vertex is added to the candidate list with some predefined probability, if the size of the candidate list is above a certain threshold.

Use of randomization allows for the total number of paths to be reduced. In iteration k, viable paths of length k determined from T_{k-1} are added to the candidate list if a random number generated is greater than some threshold. The threshold value is h/x, where h is some constant and x is the number of viable paths that may be added to the candidate list. Random selection is only performed if the candidate list is greater than some multiple of 2.

Unlike the walk algorithm, the path stored in the hash table is composed of the three-dimensional coordinates of each node. Memory requirements are increased. By storing all of the nodes in the path we are able to test for self-avoidance. There is some duplication of data, as the path is stored both in an array and a hash table. The array is used in lieu of a traceback procedure and the hash table for a quick reference for self-avoidance testing.

The simple cubic (SC) lattice is used again in testing of the path algorithm. Like the walk it is theoretically plausible for any lattice to be used in the algorithm, as shown in [14, 5].

Chapter 3

Results and Discussion

In this study, we developed new algorithms to improve on the work of Park and Levitt [14]; Rykunov [16], and Reva [28]. Our approach allows for improvement in the speed and size of proteins, that the algorithm can handle. We have been able to work with proteins consisting of over 1000 residues as compared to the 753 residues of Park and Levitt [14] and 445 of Rykunov and Reva [51] when it comes to computing optimal walks. Like Rykunov and Reva [51] we used dynamic programming in an algorithm not much different than their own. However, we used branch and bound methods to improve the running time and space of the algorithm. Like Rykunov and Reva [16] we have two algorithm, a walk algorithm that does not guarantee self-avoidance and a path algorithm that does.

The walk algorithm is polynomial in the worst case and produces an optimal solution, however, as mentioned above it does not guarantee self-avoidance. Due to the recent NP-completeness result [35], it seems highly improbable that a polynomial time algorithm exists for computing paths that minimize the CRMS value. In addition we compared the quality of a greedy solution, using the CRMS value of the walk as a lower bound on the optimal solution to the path problem. The empirical evidence suggests that the greedy algorithm is quite good.

Even though we evaluated our algorithms only on a simple cubic (SC) lattice, with modifications we found that the algorithms can be used for any other lattice.

To represent naturally occurring protein structures, we selected the same subset of 3704 PDB protein structures used by Mead *et al* [5]. Of this subset, a large portion have multiple subchains. For our analysis a subchain was defined as any polypeptide chain. We treat each subchain separately in our experiments. The separation of the subchains of 3704 protein structures yields a final set of approximately 8236 protein structures. The basis for the selection and utilization of this set follows from the selection criteria of Mead *et al* [5].

3.1 Walk

The walk algorithm described in Chapter 3 is polynomial and ensures that a globally optimal fit is found. Incorporation of branch and bound techniques significantly reduces the running time. Runs on the dataset were performed without any difficulties.

3.1.1 Results and Analysis

Results were based on a simple cubic lattice with a spacing of 3.8 Å. Validity was assured through a comparison of analysis data to the findings of Park *et al* [14] and Mead *et al* [5].

Experimental results for the set of 8236 PDB protein structures were recorded in output files. The output files contained the PDB identifier of the protein structure, the CRMS value and size of the protein sample. All the output files for the set were collated for analysis.

Analysis of the output data includes a histogram comparing the frequency of the CRMS values, a plot of protein size vs CRMS value distribution and calculation of the mean and standard deviation for the CRMS value. Ideally, the CRMS value should not depend on the number of residues in the protein.

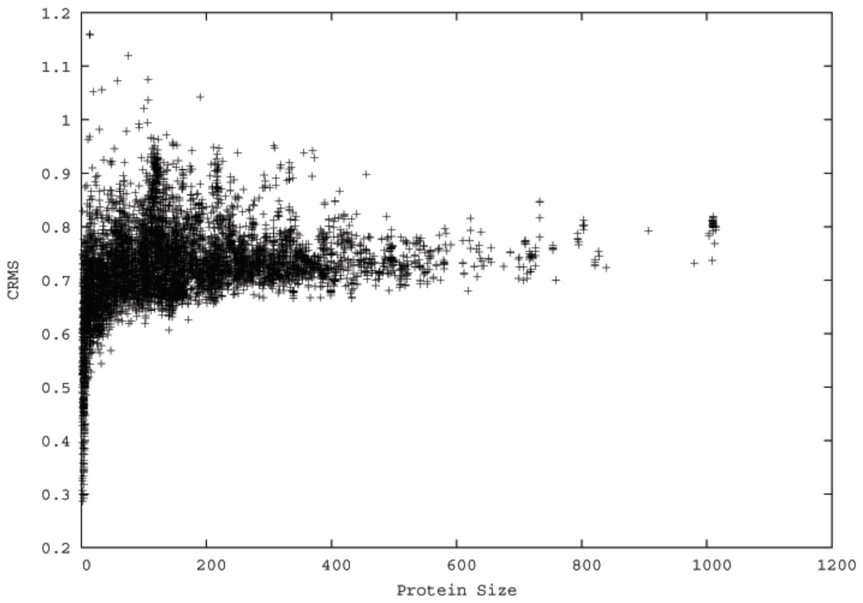

Figure 3.1: Shows a plot of the number of protein residues (size of protein) verses the CRMS value, sorted by increasing order of the number of residues.[2]

Figure 3.1 indicates that the proteins length and CRMS values are not dependent on one another. Had the values been dependent on one another, we would expect to see larger CRMS values associated with larger protein structures. However, as seen in Figure 3.1 this is not the case. CRMS values are closely related regardless of the size of the protein structure, and seem to get smaller with an increase in the number of residues.

Further analysis of the data through a histogram of CRMS frequency further substantiated the walk algorithm's performance. Results of the histogram are displayed in Figure 3.2. Once again results are based on a simple cubic lattice.

As seen in Figure 3.2 the greatest frequency of the CRMS data is found close to the 0.75 mark. With a second large selection of frequencies around the 0.90 mark.

[2] A comment of Dr. Bhattacharya revealed that two outliers were not corrected for inclusion of subchains resulting in an error, values have been recalculated and the figure changed to reflect this.

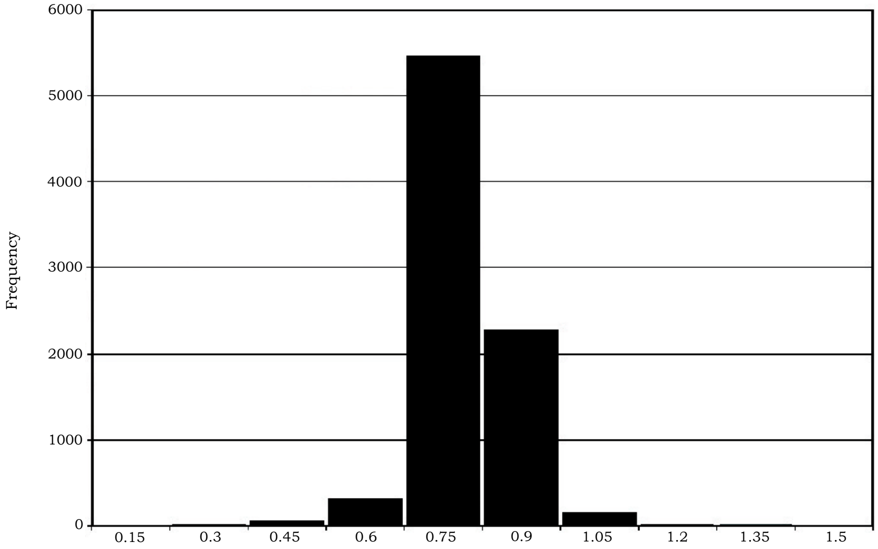

Figure 3.2: Histogram of the CRMS values from the 8236 subset of PDB protein structures

The results not only demonstrate the validity of the performance of the walk algorithm we also confirm previous findings that substantiate the use of cubic lattice's to approximate known protein structures. Previous findings include those obtained in 1995 by Park and Levitt [14] and those of Mead *et al* [5] in 2005.

Park and Levitt's [14] build-up algorithm developed in 1995 was designed to fit arbitrary models to X-ray structures. In order to test the accuracy with which the models could represent each member of their database of proteins. Models of varying complexity were considered and tested. As we discussed previously, analysis of their tests revealed that increasing model complexity above a certain point, yields little improvement in accuracy. Therefore lower complexity models with good accuracy, like the simple cubic lattice, are acceptable for representing protein structures.

Later in 2005, Mead *et al* [5] building on the work of Park and Levitt [14] developed their own novel algorithm to test for an ideal lattice that could be used in the study of inverse protein folding. They performed a three-step analysis, on a number of different lattice models, testing alignments based on CRMS, a relative all-to-all coordinate root mean square (DRMS), and an angle root mean square deviation (ARMS). Their investigation coincided with Park and Levitt [14] demonstrating that cubic lattices work well in approximating known protein structures.

Our findings show that the results (CRMS values computed) of Mead *et al* [5] are quite close to the optimal, as they are close to the lower bound given by our algorithm.

Mead *et al's* [5] data is based on an edge length 3.8 Å, whereas ours is on length 1. Therefore, prior to comparison with our mean of 0.721 their CRMS mean must first be divided by 3.8 Å. Mead *et al* [5] reported a mean CRMS value of 2.7579 for the simple cubic lattice. Modifying this value for length 1 produces a CRMS mean of 0.726, whereas the mean CRMS value for the walk is 0.721. The mean CRMS values for our algorithms differs by 3 decimal places. Such a small difference in CRMS mean shows that the optimal walk is an excellent lower bound for the path. Using this evidence, we hypothesize that if the path algorithm uses a lower bound (in branch and bound) based on walks then the running time should be 'reasonable' in practice due to search space reduction. However, to compute the lower bound array for proteins of size 1000, we would require 1000 invocations of the walk algorithm. This translates into a total time of approximately $20 \times 1000 = \frac{20000}{60} = 333.\bar{3}$ hours or close to 14 days on a 1.3 GHz machine. It was for this reason that we decided to use a different lower bound (one that could be computed a lot faster) in our experiments for the path algorithm.

The preceding discussion also illustrates the need for an incremental algorithm for computing lower bounds based on walk. An interesting theoretical question that remains open.

3.2 Path

The path problem was recently shown to be NP-complete [35]. Therefore, a polynomial algorithm to calculate the optimal path is highly unlikely. Chapter 3 describes a dynamic programming algorithm that may be used to find an approximate solution to the path. However, due to the complexity of the problem, restrictions have been imposed on the dataset and results on a greedy path algorithm have been included.

3.2.1 Results and Analysis

To test the efficiency and accuracy of the path algorithm we used the same subset of 8236 protein structures as discussed in the walk. Experimental results obtained from the samples are based on a simple cubic lattice satisfying chain connectivity and the constraint that no vertex may be used more than once in each path. Validity of our results is compared against results for the walk algorithm and the data of Park et al [14] and Mead et al [5].

Our analysis of the path algorithm on the protein data subset has shown that it is very difficult to build self-avoiding walks. Due to combinatorial explosion, memory requirements are too great to complete experimental runs. Candidate list sizes quickly expanded into the millions as compared to candidate list sizes of approximately 14125 and 24410 as recorded by the walk on the largest of the protein samples. To reduce the search space protein structures were split into sections approximately 80-120 residues long and further modifications were made to the algorithm.

We incorporated a probability screening procedure to reduce the number of possible paths. We randomly selected a subset of 363 protein structures from the original 8236 protein

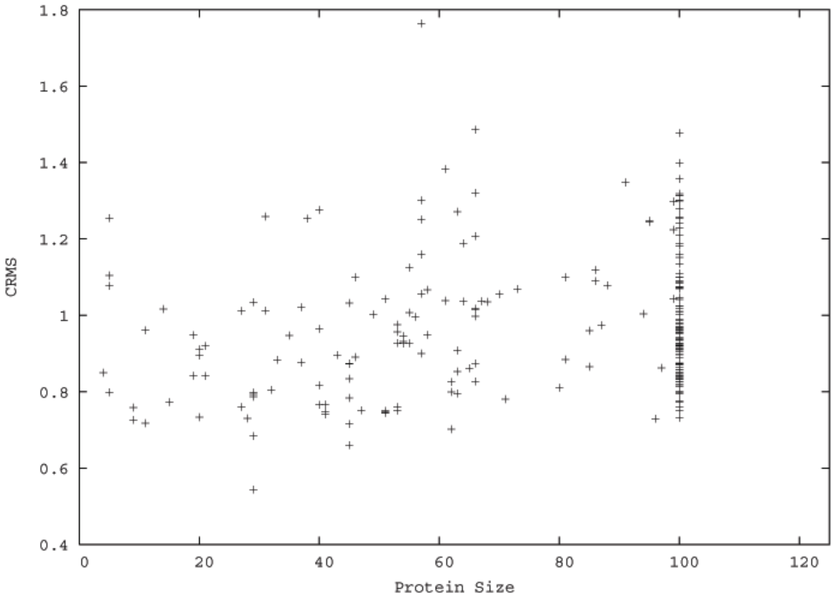

Figure 3.3: Minimum CRMS value over all subchains in a protein, obtained from path algorithm runs on a subset of 363 protein structures. Proteins are split into subchains of length no greater than 100 residues and the minimum CRMS value over all subchains is reported.

structures. Incremental adjustments to the probability function were performed and tested against the subset. We were able to produce results for a small portion of the subset, however we were not able to produce consistent results across the entire subset.

Based on the performance of the set of runs we questioned the effectiveness of the lower bounds used in the branch and bound procedure.

Using the path algorithm with a modified lower bound function we tested each protein structure in the subset of 363 structures. Runs finished with a 60% success rate and an average CRMS value of 0.99.

Results of these runs are represented in Figures 3.3, 3.4 and 3.5. Note that the sample protein structure is split into smaller sizes as reflected in Figures 3.3 and 3.4. The split protein structure is then run as separate subchains and their individual values are recorded.

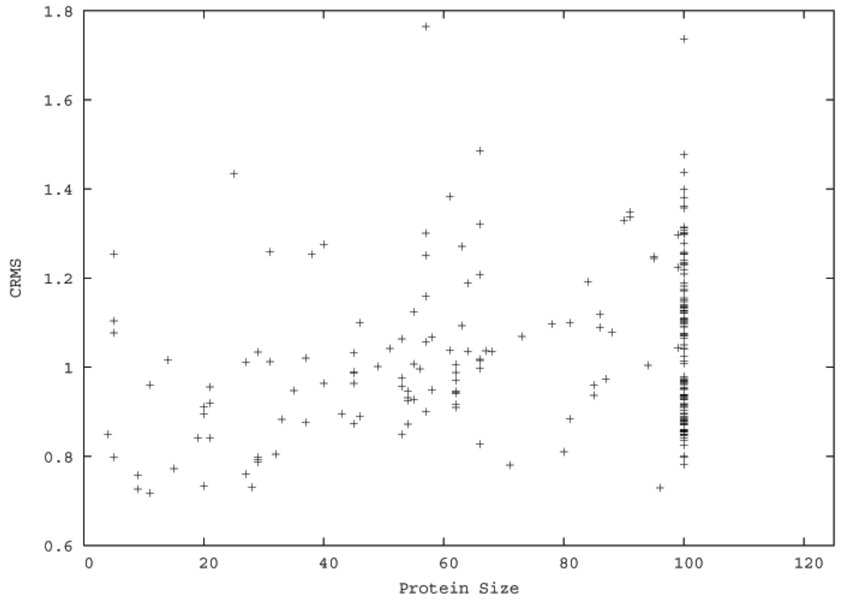

Figure 3.4: Maximum CRMS value over all subchains in a protein, obtained from path algorithm runs on a subset of 363 protein structures. Proteins are split into subchains of length no greater than 100 residues and the maximum CRMS value over all subchains is reported.

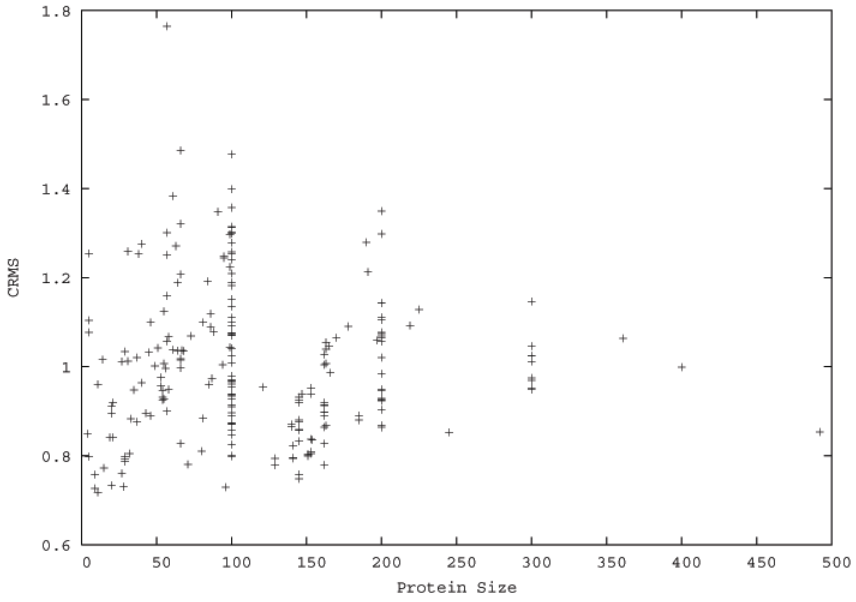

Figure 3.5: Mean CRMS value over all subchains in a protein, obtained from path algorithm runs on a subset of 363 protein structures. Proteins are split into subchains of length no greater than 100 residues and the average CRMS value over all subchains is reported.

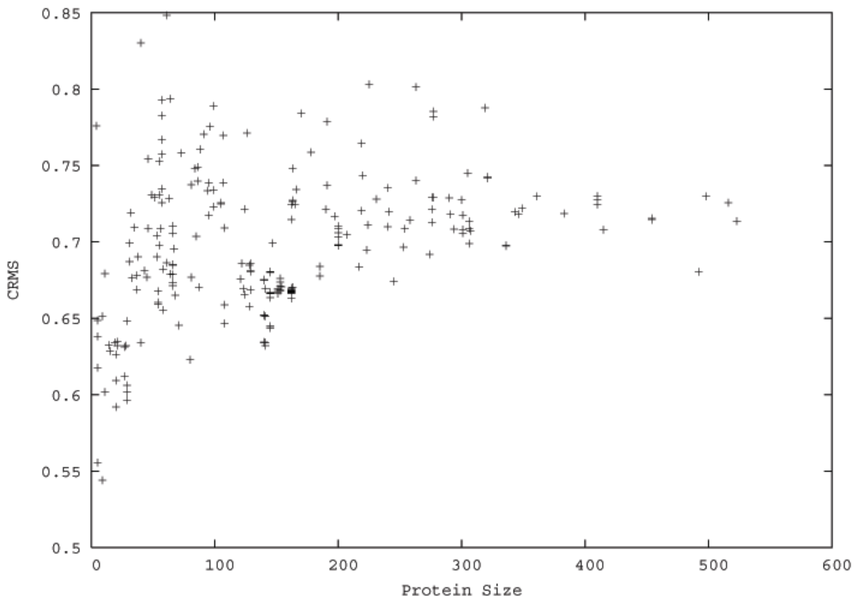

Figure 3.6: Shows a plot of the number of protein residues versus the CRMS value, sorted in the increasing order of the number of residues. [2]

Figure 3.3 shows the minimum values of the collection of subchains. Figure 3.4 shows the maximum values.

The mean CRMS values for the runs is shown in Figure 3.5. This data represents an averaging of a split protein structure therefore the CRMS average will deviate from runs based on original protein structures. Figure 3.6 displays the results of the walk algorithm on the subset of 363 protein structures.

From these results, I am able to demonstrate that the algorithm is able to produce viable results. However, it should be noted that the success rate for the runs is low, and it is not able to handle large proteins.

Recent findings of Maňuch *et al* [35] account for the difficulties encountered. For now, it remains an interesting problem to design approximation algorithms with guaranteed worst case performance ratio.

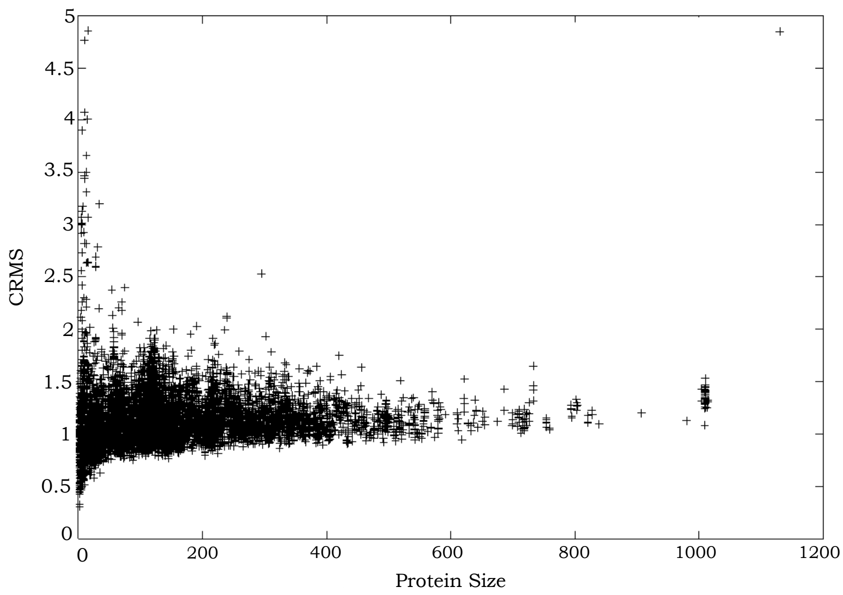

Figure 3.7: A plot of the number of residues versus the CRMS value, sorted in the increasing order of number of residues.

As one can see we were not able to produce any substantial results for our dynamic programming algorithm. We did, however, have a working greedy algorithm for the path problem that was producing results. As such we proceeded with a full experimental run on the original 8236 PDB protein structures.

We observed a 99% success rate for the greedy path algorithm and a mean CRMS value of 1.13 (Figure 3.7), that is 1.56 times that of the optimal CRMS on average. Based on these findings I surmise that for the majority of runs the greedy path algorithm makes a good upper bound.

There is room for improvement in determining a better upper bound. Determination of the efficacy of the branch and bound procedure with the lower bounds based on bipartite matchings and walks remains an interesting problem. Given that walks give a 'good' lower bound on paths, we hypothesize that the branch and bound will be effective.

Chapter 4

Open Problems

In particular we describe four interesting research questions.

Results showed that an optimal walk is an excellent lower bound for the path. Using this evidence, we can hypothesize that if a path algorithm uses a lower bound based on walks, the running time should be 'reasonable' in practice. However due to the time required to compute the lower bound array for proteins of significant size this model becomes infeasible. Development of an incremental algorithm for computing the lower bounds based on the walk algorithm would be of great value.

We were not able to produce any substantial results for our dynamic programming algorithm (for the path problem). Viablity of the results depended on the upper and lower bounds used in branch and bound. Based on the success rate of our experiments there is room for improvement in determining better upper and lower bounds. Determining other efficiently computable 'good' lower bounds remains an interesting open problem.

We established that the greedy path algorithm computes paths that have a CRMS value 1.56 times that of the optimal CRMS on average with a 99% success rate. A theoretical analysis of the greedy remains an open and interesting problem.

Recent findings of Maňuch *et al* [35] have shown the path problem to be NP-complete, therefore it is highly unlikely that a polynomial time algorithm for the problem exists. An interesting problem involves the design of approximation algorithms with guaranteed worst case performance ratio.

Bibliography

[1] Creighton, T. E. *Proteins: Structures and Molecular Properties*. Freeman, New York, NY, (1993).

[2] Hegyi, H. and Gerstein, M. The relationship between protein structure and function: A comprehensive survey with application to the yeast genome. *J Mol Biol* **288**(1), 147–164 (1999).

[3] Goldman, D. G. *Algorithmic Aspects of Protein Folding and Protein Structure Similarity*. Phd thesis, University of California Berkeley, (2000).

[4] Whitford, D. *Proteins: Structure and Function*. John Wiley & Sons, Chichester, West Sussex, England, (1995).

[5] Mead, C. R., Maňuch, J., Huang, X., Bhattacharyya, B., Stacho, L., and Gupta, A. *Investigating Lattice Structure for Inverse Protein Folding*. 30th FEBS Congrese & 9th IUBM Conference, Budapest, Hungary, (2005). Full Version in Preparation.

[6] Voet, D. and Voet, J. G. *Biochemistry, 2nd Edition*. John Wiley & Sons, (1995).

[7] Lenz, O. Picture of a β-sheet, an important motive of the secondary structure of proteins. http://en.wikipedia.org/wiki/Image:Betasheet.png, (2005). Licensed under GNU Free Documentation License.

[8] Ofran, Y. and Margalit, H. Proteins of the same fold and unrelated sequences have similar amino acid composition. *Proteins: Structure, Function and Genetics* **64**, 275–279 (2006).

[9] Taylor, W. R. Protein structure comparison using sap. In Protein Structure Prediction: Methods and Protocols, Webster, D., editor, 19–32. Humana Press (2000).

[10] Dill, K. A. Dominant forces in protein folding. *Biochemistry* **29**(31), 7133–7155 (1990).

[11] Birney, E. and Ponting, C. P. Identification of domains from protein sequences. In Protein Structure Prediction: Methods and Protocols, Webster, D., editor, 53–70. Humana Press (2000).

[12] Jones, D. T. A practical guide to protein structure prediction. In Protein Structure Prediction: Methods and Protocols, Webster, D., editor, 131–154. Humana Press (2000).

[13] Yu, Y. B. Coiled-coils: Stability, specificity and drug delivery potential. *Advanced Drug Delivery Reviews* **54**, 1113–1129 (2002).

[14] Park, B. H. and Levitt, M. The complexity and accuracy of discrete state models of protein structure. *J Mol Biol* **249**, 493–507 (1995).

[15] Gutin, A. M., Abkevich, V. I., and Shakhnovich, E. I. Evolution-like selection of fast-folding model proteins. *Proc. Natl. Acad. Sci.* **92**, 1282–1286 (1995).

[16] Rykunov, D. S., Reva, B. A., and Finkelstein, A. V. Accurate general method for lattice approximation of three-dimensional structure of a chain molecule. *Proteins: Structure, Function and Genetics* **22**, 100–109 (1995).

[17] Yue, K. and Dill, K. A. Inverse protein folding problem: Designer polymer sequences. *Proc. Natl. Acad. Sci.* **89**, 4163–4167 (1992).

[18] Covell, D. E. and Jernigan, R. L. Conformations of folded proteins in restricted spaces. *Biochemistry* **29**, 3287–3294 (1990).

[19] Dill, K. A. Theory for the folding and stability of globular proteins. *Biochemistry* **24**, 1501–1509 (1985).

[20] Dill, K. A., Bromberg, S., Yue, K., Fiebig, K. M., Yee, D. P., Thomas, P. D., and Chan, H. S. Principles of protein folding - a perspective from simple exact models. *Protein Sci* **4**, 561–602 (1995).

[21] Go, N. Theoretical studies of protein folding. *Ann Rev Biophys Bioeng* **12**, 183–210 (1983).

[22] Gupta, A., Maňuch, J., and Stacho, L. Structure-approximating inverse protein folding problem in 2d hp model. *J Comput Biol* **12**, 1328–1345 (2005).

[23] Hinds, D. A. and Levitt, M. A lattice model for protein structure prediction at low resolution. *Proc. Natl. Acad. Sci.* **89**(7), 2536–2540 (1992).

[24] Hinds, D. A. and Levitt, M. Exploring conformation space with a simple lattice model for protein structure. *J Mol Biol* **243**(4), 668–682 (1994).

[25] Koehl, P. and Delarue, M. Building protein lattice models using self-consistent mean field theory. *J Chem Phys* **108**(22), 9540–9549 (1998).

[26] Levinthal, C. Are there pathways for protein folding? *J Chem Phys* **65**, 44–45 (1968).

[27] Levitt, M. A simplified representation of protein conformations for rapid simulation of protein folding. *J Mol Biol* **104**(1), 59–107 (1976).

[28] Reva, B. A., Finkelstein, A. V., Rykunov, D. S., and Olson, A. J. Building self-avoiding lattice models of proteins using a self-consistent field optimization. *Proteins: Structure, Function and Genetics* **26**, 1–8 (1996).

[29] Shakhnovich, E. and Gutin, A. M. A new approach to the design of stable proteins. *Protein Eng* **6**(8), 793–800 (1993).

[30] Shakhnovich, E. Proteins with selected sequences fold into unique native conformation. *Phys Rev Letters* **72**(24), 3907–3910 (1994).

[31] Andrew, C. D., Penel, S., Jones, G. R., and Doig, A. J. Stabilizing nonpolar/polar side-chain interactions in the α-helix. *Proteins: Structure, Function and Genetics* **45**, 449–455 (2001).

[32] Lyu, P. C., Liff, M. I., Marky, L. A., and Kallenbach, N. R. Side chain contributions to the stability of α-helical structure in peptides. *Science* **250**(4981), 669–673 (1990).

[33] Śanchez, R. and Šali, A. Comparative protein structure modeling: Introduction and practical examples with modeller. In Protein Structure Prediction: Methods and Protocols, Webster, D., editor, 97–129. Humana Press (2000).

[34] Šali, A., Shakhnovich, E., and Karplus, M. Kinetics of protein folding a lattice model study of the requirements for folding to the native state. *J Mol Biol* **235**, 1614–1636 (1994).

[35] Maňuch, J. and Gaur, D. Fitting protein chains to cubic lattice is NP-complete. *To appear in the Proceedings of 2007 APBC, Full Text*, (2007).

[36] Godzick, A., Skolnick, J., and Kolinski, A. Regularities in interaction patterns of globular proteins. *Protein Eng* **6**(8), 801–810 (1993).

[37] Kolodny, R., Koehl, P., and Levitt, M. Comprehensive evaluation of protein structure alignment methods: Scoring by geometric measures. *J Mol Biol* **346**, 1173–1188 (2005).

[38] Thompson, J. D., Higgins, D. G., and Gibson, T. J. Clustal W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position - specific gap penalties and weight matrix choice. *Nucleic Acids Research* **22**(22), 4673–4680 (1994).

[39] Jones, N. C. and Pevzner, P. A. *An Introduction to Bioinformatics Algorithms*. The MIT Press, Cambridge, Massachusetts, (2004).

[40] Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. The protein data bank. *Nucleic Acids Res* **28**(1), 235–242 (2000).

[41] Aluru, S., editor. *Handbook of Computational Molecular Biology*, volume 9 of *Computer and Information Science Series.*, chapter The computational complexity of protein structure prediction in simple lattice models. Chapman and Hall/CRC (2005).

[42] Berger, B. and Leighton, T. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *J Comp Biol* **5**(1), 27–40 (1998).

[43] Bryngelson, J. D. and Wolynes, P. G. Spin glasses and the statistical mechanics of protein folding. *Proc. Natl. Acad. Sci.* **84**, 7524–7528 (1987).

[44] Leopold, P. E., Montal, M., and Onuchic, J. N. Protein folding funnels: A kinetic approach to the sequence-structure relationship. *Proc. Natl. Acad. Sci.* **89**, 8721–8725 (1992).

[45] Park, B. H. and Levitt, M. Energy functions that discriminate x-ray and near-native folds from well-constructed decoys. *J Mol Biol* **258**, 367–392 (1996).

[46] Reva, B. A., Finkelstein, A. V., and Skolnick, J. A self-consistent field optimization approach to build energetically and geometrically correct lattice models of proteins. In *RECOMB '98: Proceedings of the Second Annual International Conference on Computational Molecular Biology*. ACM Press, (1998).

[47] Lifshits, I. M., Grosberg, A. Y., and Khokhlov, A. R. Some problems of the statistical physics of polymer chains with volume interactions. *Rev Mod Phys* **50**, 653–713 (1979).

[48] Sun, S., Thomas, P. D., and Dill, K. A. A simple protein folding algorithm using a binary code and secondary structure constraints. *Protein Eng* **8**(8), 769–778 (1995).

[49] Duan, Y. and Kollman, P. A. Computational protein folding: From lattice to all-atom. *IBM Systems Journal* **40**(2), 297–309 (2001).

[50] Yue, K., Fiebig, K. M., Thomas, P. D., Chan, H. S., Shakhnovich, E., and Dill, K. A. A test of lattice protein folding algorithms. *Proc. Natl. Acad. Sci.* **92**(1), 325–329 (1995).

[51] Reva, B. A., Rykunov, D. S., Olson, A. J., and Finkelstein, A. V. Constructing lattice models of protein chains with side groups. *J Comp Biol* **2**(4), 527–535 (1995).

[52] Corman, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, (2001).