COMPARING AND MODELING PROTEIN STRUCTURE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Rachel Kolodny
August 2004

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael Levitt
(Principal Co-Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Leonidas J. Guibas
(Principal Co-Adviser)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jean–Claude Latombe

Approved for the University Committee on Graduate Studies.

Abstract

Proteins are remarkably versatile macromolecules involved in essentially all biological processes. The detailed three-dimensional structure of a protein encodes its function. A fundamental computational challenge in the study of proteins is the comparison and modeling of protein structure. Structural similarities of proteins can hint at distant evolutionary relationships that are impossible to discern from protein sequences alone. Consequently structural comparison, or alignment, is an important tool for classifying known structures and analyzing their relationships. Efficient models are crucial for structure prediction; in particular, for the generation of decoy sets (*ab initio* protein folding) and loop conformations (homology modeling).

The first part of this work focuses on protein structural alignment, namely, the comparison of two structures. We formalize this problem as the optimization of a geometric similarity score over the space of rigid body transformations. This leads to an approximate polynomial time alignment algorithm. Our result is theoretical, rather than practical: it proves that contrary to previous belief the problem is not NP-hard. We also present a large-scale comparison of six publicly available structural alignment heuristics and evaluate the quality of their solutions using several geometric measures. We find that our geometric measure can identify a good match, providing a method of analysis that augments the traditional use of ROC curves and their need for a classification gold standard.

In the second part, we present and use an efficient model of protein structure. Our model concatenates elements from libraries of commonly observed protein backbone fragments into structures that approximate protein well. There are no additional degrees of freedom so a string of fragment labels fully defines a three–dimensional

structure; the set of all strings defines the set of structures (of a given length). By varying the size of the library and the length of its fragments, we generate structure sets of different resolution. With larger libraries, the approximations are better, but we get good fits to real proteins with less than five states per residue. We also describe uses for these libraries in protein structure prediction and loop modeling.

Acknowledgements

I was blessed to have had the opportunity to come to Stanford. It is truly an exceptional place, with many exceptional people. Thesis acknowledgments present the opportunity to pause, and express appreciation to the people and things that make one's life better. Let me pause then.

My advisors, Michael Levitt and Leo Guibas, faithfully guided me along the rugged PhD path. They taught me a great deal, far more than a few sentences can summarize. Above all, I am indebted for the teachings that were hard to articulate and thus relied on a long series of examples — judging if a question is interesting and deciding if an answer is convincing. Michael also put extra care in improving my writing skills. With great foresight, he chose the lovely setting of Paris to ease my painful experience. I was also fortunate to work with Patrice Koehl, who was like an advisor to me, and watched closely over my studies. It was an honor and a pleasure to work with these scholars; I cannot thank them enough for this experience.

Many wise people crossed my way during my studies, and contributed to or commented on my work. Prof. Jean-Claude Latombe, Prof. Vijay Pande, and Prof. Sebastian Doniach served on my thesis committee, and I thank them for doing so. I enjoyed collaborating with Nati Linial, my masters advisor, and with Yonatan Bilu — both of the Hebrew University. I greatly appreciate the enlightening discussions I had with Herbert Edelsbrunner and Pankaj Agarwal of Duke University, Ileana Streinu of Smith College, Jack Snoeyink of UNC, and Chris Lee of UCLA.

Leo Guibas, Michael Levitt, and the American people funded my studies through the National Science Foundation grant CCR-0086013. The Sudarsky center for computational biology in the Hebrew University supported the travel needed for starting

the structural alignment project.

I spent most of my days at my Levitt lab desk. I am grateful to Golan Yona for suggesting that I join the lab, and for his generous welcome in my first months on campus. Tanya Raschke contributed to my work throughout: debugging my thoughts, teaching me biology, and sharing daily coffees. I spent many pleasurable hours chatting with Yu Xia, Michael Sykes, Erik Sandelin, Erik Lindahl, Chris Summa and Dahlia Weiss. I was also fortunate to hear the innovative perspective of Chris Lee, a lab alumnus. Bick Do and Jeannie Lukas, apart from fixing things that seemed un-fixable, always had a kind and encouraging word.

I was also a member of the Guibas group. I much enjoyed our weekly seminars, as well as my everyday interaction with the group members, especially Daniel Russel, Afra Zomorodian, Niloy Mitra, Qing Fang, Natasha Gelfand, and An Nguyen. I was fortunate to have had many insightful conversations with Ileana Streinu during her visit with the group, and in two of her workshops in Barbados. Pankaj Agarwal was my partner in crime for coffee sipping and question tackling, both during his visit to Stanford, and later when he kindly hosted me at Duke. Last but by no means least, Julien Basch, my Geometric Algorithms TA, has been providing computational geometry (and other) support, to this very day.

All in all, it was five years — long enough to make close friends. Adela Ben-Yakar, Awino Kuerth, Daniel Russakoff, Daniel Russel, Ian White, Issi Rosen-Zvi, Itay Lotan, Joyce Haas, Julien Basch, Luis Sentis, Mattan Erez, Mor Naaman, Ofer Levi-Tsabari-Ting, Rachel Garfield, Rina Levitt, Rosetta Pedotti, Sharon Kuperfish, Tanya Raschke, and Uli Barnhoefer shared my life here, making California feel like home. I cannot imagine my world without our joint ceremonies: morning and four o'clock coffees, Friday dinners, Sunday brunches, happy hours, and stupid-movie nights. Meanwhile, back in the homeland, friends (by now considered family) kept the fort. The hours on the phone with Miki Cohen, Miri Tsafrir, and Yonatan Bilu gave the illusion that they were with me all along. I am proud to call all these people my friends; they bring great joy into my life.

Two of my extra-curricular obsessions, 'Law and Order' and ceramics, were not the least shared by my usual crowd. While the former is a private endeavor, the latter

was shared with my many friends and teachers at the Palo Alto Art Center; I thank Rina Levitt, Norma Lyon, Glenn Matsumara, Susan Witebsky, and Frank Arroyo for joyful hours.

These years would have not been the same had I not spent them with Yuval, my love and my home in this faraway land. Even in the restricted context of this thesis, he made many contributions: Most significantly, his structured logic balances my fuzzy line of thought; I also thank him for editing almost everything I have written, and for patiently pointing out, time and again, that an example is not a proof.

I thank my parents, Amira and Yeshu, for bringing me *ad halom*, and raising me knowing girls can do anything; their love and support forms the foundation for everything I do. Years ago, they sowed the seeds of my graduate school dreams, and exactly as advertised, it gave me a fresh perspective and a few great years. I thank my beloved brothers, Noam and Uri, who have been enduring their kid sister so well and so long. With their spouses, Varda and Hilla, they are now attending to the next Kolodny generation; one day, this new generation will travel across seas to get their PhDs, thus securing my short link in a longer chain.

It was a great privilege. I would like to thank all those who were involved, suggested, taught, participated or shared a coffee: You have enriched my life tremendously. Thank you.

Stanford, August 2004

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Proteins are molecules that govern virtually all aspects of life. Creighton [18] describes very well their versatile roles:

> Proteins store and transport a variety of particles ranging from macromolecules to electrons. They guide the flow of electrons in the vital process of photosynthesis; as hormones, they transmit information between specific cells and organs in complex organisms; some proteins control the passage of molecules across the membranes that compartmentalize cells and organelles; proteins function in the immune systems of complex organisms to defend against intruders (the best known are the antibodies); and proteins control gene expression by binding to specific sequences of nucleic acids, thereby turning genes on and off. Proteins are the crucial components of muscles and other systems for converting chemical energy into mechanical energy. They also are necessary for sight, hearing, and other senses. And many proteins are simply structural, providing the filamentous architecture within the cells and the materials that are used in hair, nails, tendons, and bones of animals.

A protein has a unique three-dimensional structure, which carries out, or enables, its function. Consequently, evolution conserves the structure of proteins significantly more than it conserves their amino acid sequence. Thus, researchers can use structural

similarity as a powerful "telescope" for looking back to early evolutionary history [14, 51].

Proteins are linear macromolecules — chains of smaller molecules called amino acids; we refer to the molecules along the protein chain as 'residues'. There are twenty types of amino acids: all have the same backbone component and a type-specific side-chain component. The backbone components connect the amino acids to each other, forming a chain from which the side chains branch off. Each backbone component has a central carbon atom denoted C^{α}. The structural elements along the backbone between those C^{α} atoms are known as peptide groups; these are planar with almost fixed bond lengths and bond angles between their atoms [6]. Equivalently, the backbone has only two rotational degrees of freedom per amino acid: ϕ about the N–C^{α} bond, and ψ about the C^{α}–C bond (see Figure 1.1). A protein is described either by the sequence of its amino acids, or by its structure. The structure is the three-dimensional positions of the backbone and side chain atoms. A common, viable simplification, is to consider only the positions of the backbone atoms, or equivalently the values of the (ϕ, ψ) angles. We often further simplify the model by using only one atom per amino acid, most often the C^{α} atom; this makes the unit of structure (a C^{α} atom) correspond to a unit of sequence (an amino acid).

Measurements of protein atom positions are inherently noisy. Most importantly, proteins are generally flexible except for the peptide groups, and the coordinates of their atoms vibrate about their mean positions [58]. In addition, these coordinates are acquired via physical experiments, which are prone to measurement errors [12]. It follows that when dealing with functions defined over the coordinates of proteins, there is not much point in seeking exact solutions. Rather, approximate solutions are called for, and we do not assume that the approximation error tends to zero; instead, we assume it is bounded from below by a small positive value.

1.1 Protein structural similarity

A fundamental task in structural molecular biology is comparing and detecting similarity between protein structures. Perutz *et al.* [92] showed in 1960 that myoglobin

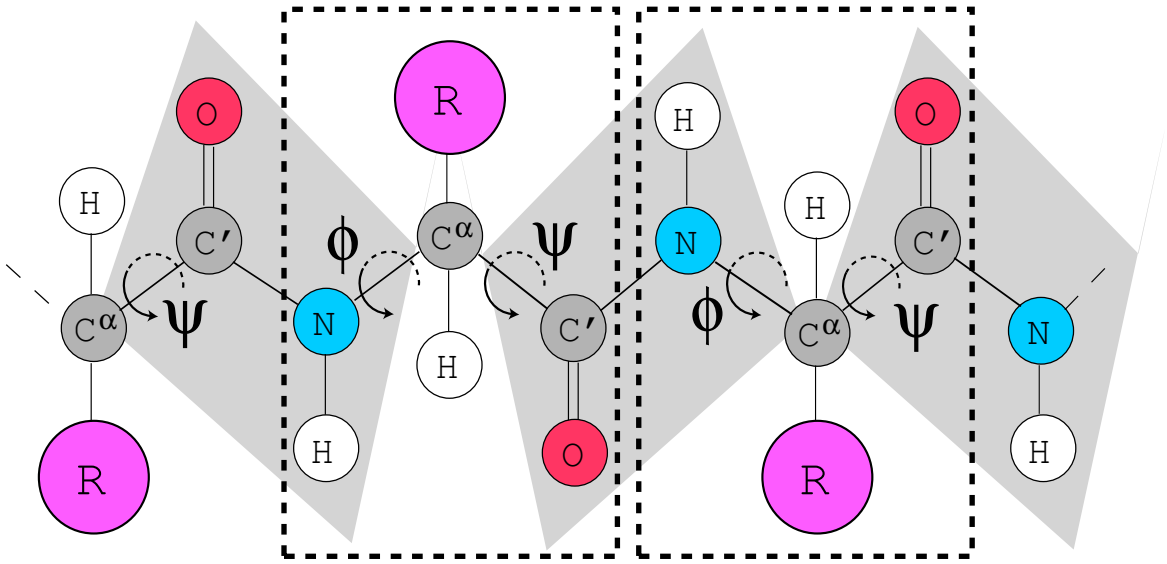

Figure 1.1: Schematic view of a segment of the protein backbone: the backbone chain is drawn in detail, and the side chains are marked by pink circles with the letter R. Two of the resides are marked with a dashed line. The planar peptide groups between the C^{α} atoms are shown in light gray, and the two rotational angles (ϕ, ψ) are marked.

and hemoglobin have similar structures, even though their sequences are different. Functionally, these two proteins are similar and are involved with the storage and transport of oxygen, respectively. Since then, researchers have continued to look for structural similarity in hope of detecting shared functionality or origin.

There are many applications for automatic methods of protein structural comparison, and the rapid growth of the Protein Databank (PDB) [5] underscores the need for fast and accurate comparison methods. Structural biologists have been making intensive efforts to classify systematically all known protein structures, based on their structural (and possibly sequence) similarity [36, 86, 87, 105]. These efforts led to structural databases such as SCOP [81], FSSP [50] and CATH [87]. Identifying similar known structures is also very useful when studying a newly determined structure [50]. Lastly, many studies (e.g., [119, 101]) use structural similarity as the 'gold standard' for sequence alignment.

1.1.1 Protein structural alignment

The structural alignment problem is the structural analog of the well-known sequence alignment problem; it formalizes the problem of detecting similar sub-structures. The input to the former consists of two protein structures, or chains, in the three-dimensional space $\mathbb{R}^3$. The desired output is a pair of *maximal* sub-chains, one from each protein, that exhibit the highest degree of similarity. The sequential alignment of the two sub-chains is a sequence of residue pairs, also called a correspondence. Figure 1.2 is a two-dimensional illustration of the structural alignment problem.

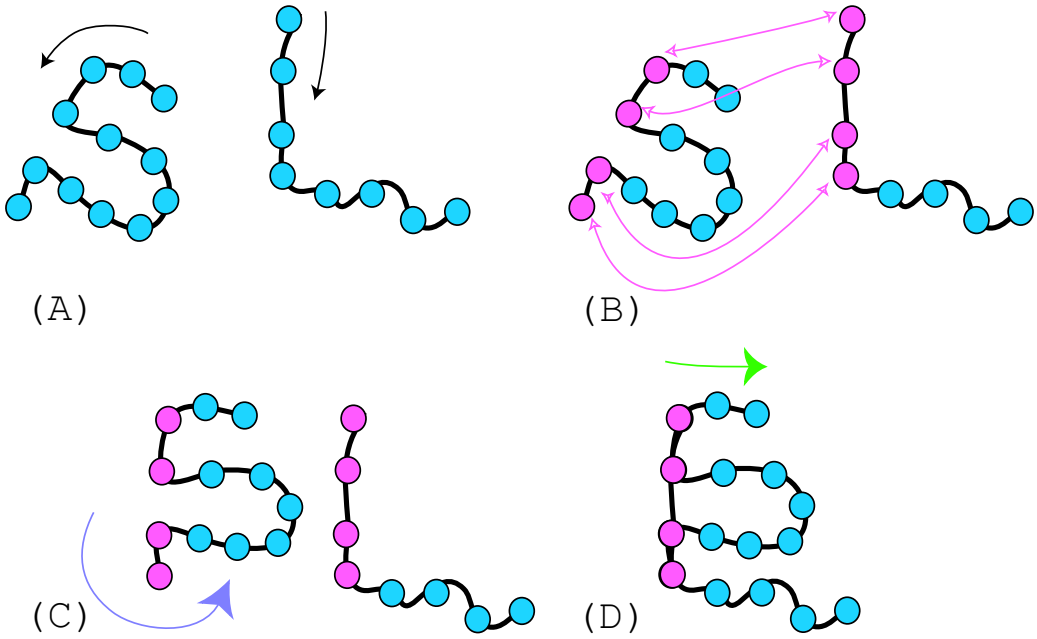

Figure 1.2: Two-dimensional illustration of the structural alignment problem. The input to the problem, shown in panel (A), is two chains; the black arrows give their N-to-C direction. An example correspondence of four residues is colored in pink in panel (B), with arrows associating the pairs of atoms from the two chains. We can compare these two sub-chains using dRMS, in which case we consider corresponding internal distances. Alternatively, if when using cRMS, we need to be rotate (panel (C)) and translate (panel (D))one of the chains, such that corresponding atoms are close in space.

There are two main methods to quantify similarity between sub-chains of two

proteins. The first method computes internal distances between corresponding pairs of atoms in the two sub-chains, and compares these distances in the two proteins under consideration; this distance measure is denoted dRMS. The second method uses the actual Euclidean distance between corresponding atoms in the two proteins under comparison; this distance measure is denoted cRMS. To calculate the optimal cRMS, the method must also determine the rigid transformation that optimally positions the two structures vis-a-vis each other. In both cases the similarity is geometric, or equivalently it is captured by the Euclidian distance between corresponding (generally C^{α}) atoms or atom pairs.

These two methods of quantifying similarity give rise to two approaches for solving the structural alignment problem. Investigators subscribing to the first approach have developed heuristic algorithms that compare the internal distance matrices in search of the optimal correspondence (e.g., [40, 48, 79, 104, 116, 117, 130]). An advantage of these algorithms is that they bypass the need to find an optimal rigid transformation. The most commonly used structural alignment server, DALI [48], belongs to this group. Along the second approach, heuristic algorithms have been developed to optimize the correspondence and the rigid transformation simultaneously (e.g., [1, 21, 59, 61, 73, 75, 84, 112, 125, 128]). In addition there are hybrid approaches, which use both representations (e.g., [129, 118]). Excellent reviews of these and other methods can be found in [86, 70, 26, 63].

A prevailing sentiment in both research communities is that structural alignment requires exponential computational resources, and thus investigations should concentrate on heuristic approaches [51, 26, 39, 104]. Indeed, none of the above mentioned heuristics guarantees finding an optimal alignment with respect to any scoring function.

As a side note, notice that there can be several distinct solutions to the structural alignment problem (see Figure 1.3 for a two-dimensional illustration). A priori, all solutions that are close to the optimum (depending on measurement errors) are equally interesting. Moreover, as Zu-Kang and Sippl [132] noted, the multiple correspondences that exist may be all equally viable from the biological perspective, and hence deserve equal computational attention.

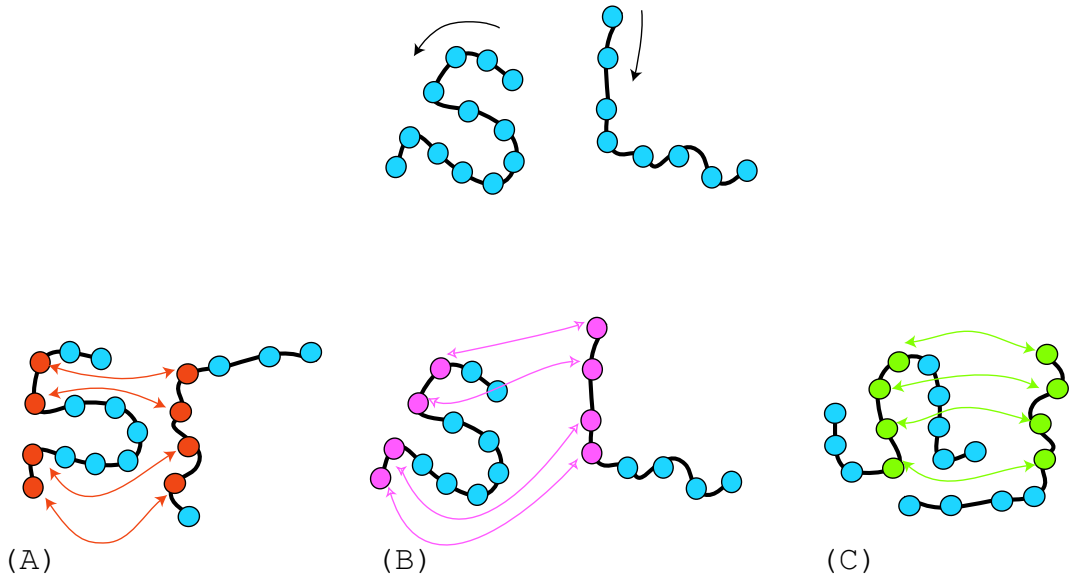

Figure 1.3: Two-dimensional illustration of multiple solutions for structural alignment. The input to the problem is shown on top. Three possible alignments are (A), (B), and (C). Especially when taking into account the noisy nature of the coordinates, all solutions are a priori equally interesting.

1.1.2 Comparisons of sequence alignment and structural alignment heuristics

In 1970 Needleman and Wunsch [82] presented an efficient dynamic programming algorithm that finds the optimal sequence alignment for a family of sequence alignment scoring functions. The difficulty in aligning sequences lies in finding the parameters of a scoring function that best captures the *evolutionary* similarity between amino acids. A scoring function is defined via a substitution matrix, and many studies try to specify matrices that produce biologically meaningful sequence alignments (e.g., [19, 46]). Another challenge is to develop fast heuristics that find sub-optimal yet meaningful alignments. Similarly to their structural counterpart, sequence alignment methods often scan databases of protein sequences in hope of detecting homologs, or highly similar sequences, to a newly found sequence. The rapid growth in size of the sequence databases has led to the development of fast heuristics such as BLAST [3] and FASTA [90].

It is hard to rely on sequence alignment scores for comparing sequence alignments found by different programs, because they quantify the elusive concept of evolutionary distance. Instead, researchers use structure similarity (often derived from a classification) as the gold standard. For example, Brenner *et al.* [7] use the hierarchical protein classification SCOP [81], which is based on structural and sequence similarity, as their gold standard for comparing sequence alignment programs. Others [23, 32, 101] also evaluate sequence alignment methods using structural alignment.

Previous evaluations of structural alignment heuristics use CATH [87] or SCOP [81] classifications as a gold standard, and verify that pairs of structures that are classified the same are in fact similar, whereas all other structure pairs are not. Novotny *et al.* [83] assess the performance of structural alignment methods, as part of their evaluation of structural alignment (or fold comparison) servers. Their study uses CATH as the gold standard, and queries the servers' databases with approximately seventy structures. Sierk and Pearson [107] compare ROC curves [41] to evaluate the success of different methods in detecting domains of the same homology or topology, as defined by CATH; they test one query in each of 86 families. Using SCOP as the gold standard, Leplae and Hubbard [71] built a web-server that evaluates structural alignment methods by comparing their ROC curves. Descriptions of structural alignment methods sometimes include evaluations of the methods. For example, Gerstein and Levitt [34] evaluate STRUCTAL using SCOP; Shindyalov and Bourne [104] compare CE to DALI; Shapiro and Brutlag [103] evaluate FoldMiner, VAST and CE by comparing their ROC curves, using SCOP.

However, since structural similarity is a geometric concept rather than an evolutionary one, judging which of several alignments for a pair of structures is best, is easy; researchers have devised geometric match measures for this purpose. The significant properties of a structural alignment, which are used by these measures, are the geometric similarity, the number of matched residues, and the number and length of alignment gaps. Clearly, better alignments match more residues, have fewer gaps and are more geometrically similar. These alignment properties are not independent (for example, shortening the alignment or introducing many gaps can decrease the cRMS deviation), and the alignment match measures attempt to balance these values. In

this work, we consider the SAS [112], SI [62], and MI [62] match measures. Note that deciding which alignment is the most geometrically similar is an easier question than evaluating whether an alignment is biologically meaningful [111, 107].

1.2 Approximation nets for protein structure

We mention some concepts from the computational geometry community that influenced our modeling in protein structure space [74]. Let X be a set; here, X is all chains (up to some maximal length) in three-dimensional space. Formally, a probability measure μ is defined on X, but since this is an intuitive discussion we sidestep this point. A range space is a pair (X, R) where X is a set and R is a set of (μ-measurable) subsets of X. A better way to describe a protein is as a subset of X, rather than a point in X. This subset is the particular measured structure and all structures within some cRMS deviation from it. Considering a set of subsets R rather than points in X, captures the flexibility of protein structure as well as the presence of noise in the measurements. The cRMS deviation that defines each subset in R varies depending on the application – one may be interested in fairly accurate approximations (i.e. within 1 Å cRMS), or less accurate approximations may suffice.

An epsilon net is a set that intersects all sufficiently large subsets in R. Formally, a subset $Y \subset X$ is called an epsilon-net for (X, R) if for all $S \in R$ $\mu(S) \geq \epsilon$ implies that $Y \cap S \neq 0$. Often when modeling protein structure, we seek an approximation to every protein structure, or a representative within some cRMS deviation. Thus, we can think of a subset that describes a protein as large when it includes all approximations within a high cRMS value, and we seek an analog of an epsilon net for the subsets of chains describing proteins.

Thinking of epsilon nets is useful because it maintains our desire to deal with finite (and small) sets while exploiting the freedom offered by finding only approximate solutions (see Figure 1.4). There are several requirements that we would like to satisfy when constructing approximation nets. First, we would like a model that will work for different applications or for different bounds on the accuracy. Second, we would like the nets to be as small as possible (most economical). Lastly, we prefer

that the description of the net is implicit — that is there is no need to list all its elements.

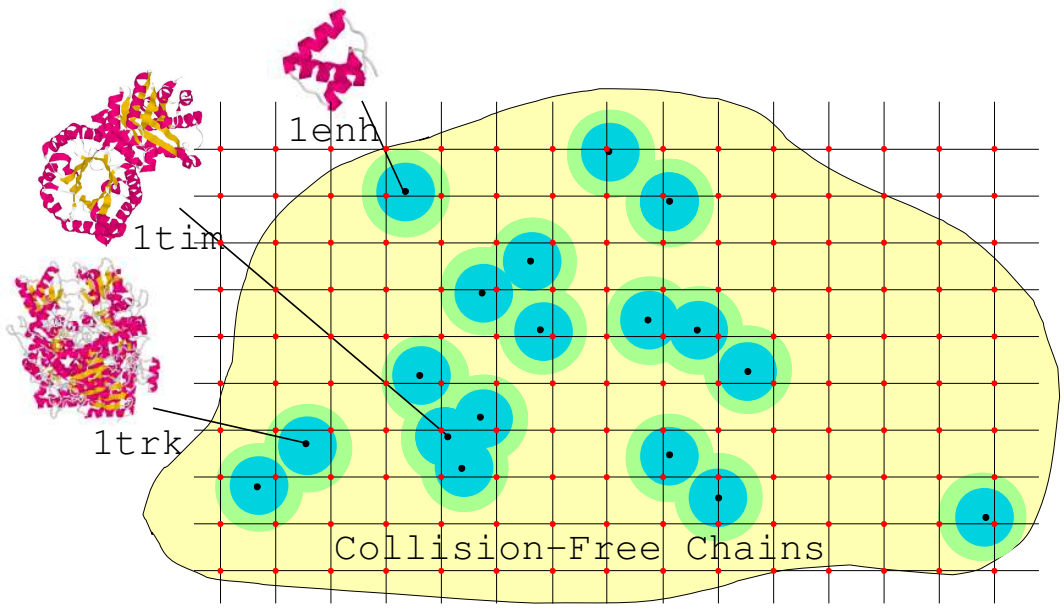

Figure 1.4: Illustration of approximation nets for protein structure. The high-dimensional space of self avoiding chains is drawn in yellow, and observed protein structures are marked as black points; we specify three proteins - 1enh, 1tim, and 1trk. A better description of proteins is a set of structures that approximate the measured structure. We illustrate the sets describing proteins with light blue and green balls: the light blue approximations are more accurate than the green ones. An approximation net is illustrated by all structures on the intersection points of a grid, implicitly defined by the horizontal and vertical lines (marked with red points). In this cartoon, the net approximates proteins well – every protein has a red net point in the set of its green approximations.

1.3 Models of protein structure

Understanding the nature of protein structure has been a subject of intense study for several decades. More than fifty years ago, Corey and Pauling [15] described the two common types of local secondary structure, the α-helix and the β-sheet. Ten years later, Ramachandran [93] ascribed the limited (ϕ, ψ) torsion angles of each residue

due to the interactions of the side chain with its backbone. In 1986, Jones and Thirup [55] discovered that almost all regions of the protein backbone are comprised of repeating canonical structures. These regions, up to ten residues long, provided an efficient method for interpreting electron density maps. Unger *et al.* [121] followed by classifying peptide backbone units 4 to 10 residues long, into a collection of fragments. These building block units constitute an intermediate level of protein structure representation between single residues and secondary structure. Since then, many studies have investigated the classification of protein fragments in general, and loop structures in particular [20, 127, 85, 78].

By restricting the local conformations of individual residues to a handful of states, one can discretize protein conformation space, such that a fixed-length chain has only finitely many spatial arrangements. Alternatively, when using continuous values for the (ϕ, ψ) torsion angles, the protein chain has infinitely many conformations, even if these angles are restricted to allowed regions. Intuitively, the discretization defines sets of structures that serve as epsilon-nets of protein structure space. The utility of a discrete model depends on the accuracy with which it models real protein conformations as well as on its size. Here, the size of a model is a function of its complexity or its average number of allowed states per residue. Rooman *et al.* [97] and Park and Levitt [89] show that discrete models that take into account the uneven (ϕ, ψ) distribution of single residue conformations in proteins are more accurate (for a fixed complexity). Others (e.g., [47, 22, 131]) use lattice models to discretize conformation space.

1.3.1 Applications of protein structure models to protein structure prediction

Efficient models of protein structure are useful for the hard task of developing methods for protein folding, or predicting structure from sequence. We denote by 'target', the protein whose structure we want to predict. A determining factor of the prediction difficulty is whether there is a homolog structure (i.e. the structure of another protein with similar sequence) in the PDB. Cases where the target protein has no homologs

of known structure are harder; this problem is called *ab initio* structure prediction. When a homolog with known structure exists the prediction is easier because we can modify that structure [9]; this is called homology or comparative modeling. The portions in the structure that need to be removed and rebuilt are the misaligned ones, and often correspond to exposed loop regions that connect secondary structure elements. The missing loops are short chains of up to 14 residues; finding these chains is also called 'the loop closure problem'. It is, in effect, a mini *ab initio* problem.

Protein structure prediction is often decomposed into two sub-problems: (1) generating candidate structures (such candidates for *ab initio* folding are called 'decoys'), and (2) devising a scoring function that discriminates between near native structures and non-native structures. Clearly, the scoring function should separate the native structure from non-native decoys in our candidate set. Many studies (e.g., [115, 110, 91, 31] of decoys, and [80, 30, 10, 11] of loop structures) consider the parallel question of searching for conformations that have low energy; they use minimization-based methods and search the landscape of the scoring functions. However, as Huang *et al.* [53] pointed out, a clean separation of the two sub-problems is advantageous as it allows an easier comparison between different energy potentials, without penalizing scoring functions with a rugged landscape.

It is easier to generate candidate loop structures than to generate candidate folds (full protein structures) because loops are shorter and constrained at their ends. As expected, methods that were designed for generating short chains cannot generate longer chains. Jones and Thirup [55] noticed that the PDB has sufficient sampling of short loops, suggesting that a simple search for segments that satisfy the end geometric constraints suffices. Researchers estimated differently the maximal loop length for such a PDB search: Fidelis *et al.* [29] estimated four residues; later studies estimated greater values: Vlijmen and Karplus [122] nine and Du *et al.* [24] fifteen. An alternative approach that works only for short loops is placing a chain analytically by solving the constraints implied by the fixed length segments and the end points (e.g., [38, 126]). As this approach is similar to the robotics problem of inverse kinematics, advanced robotics techniques were also applied to this problem (e.g., [11]). A recent review by Honig and co workers [2] surveys many loop building methods.

1.4 Outline of research

This dissertation is organized as follows: We begin with defining some terms in Chapter 2. Then, we describe three research projects. In the first one, detailed in Chapter 3 and summarized in Subsection 1.4.1, we address the problem of protein structural alignment; this was published in [67]. In the second, described in Chapter 4 and summarized in Subsection 1.4.2 we present a new methodology for comparing structural alignment heuristics and compare six publicly available programs (this has been submitted for publication). The third, summarized in Subsection 1.4.3, is an economical model of protein structure. In Chapter 5 we present this economical model; this was published in [64]. We also employ it in Chapter 6 to generate decoys and build protein loops; this was published in [66, 65]. Lastly, Appendix A documents a novel clustering heuristic that we use.

1.4.1 Protein structural alignment in polynomial time

First, we present a polynomial time algorithm to the structural alignment problem, and prove that contrary to common belief [26, 39, 51, 104], finding ϵ-approximations of the optimal solutions is computationally feasible, albeit too slow to be a useful everyday tool. Our algorithm optimizes both the correspondences and rigid transformations, and it is not heuristic: it guarantees finding ϵ-approximations to all solutions of the protein structural alignment problem. To bound the size of the solution space, we first consider the complexity of searching rotation and translation space, and show that it depends polynomially on the length of the longer protein, n, and on $1/\epsilon$ for an approximation parameter ϵ. On the other hand, the number of possible correspondences grows exponentially with the length. Based on these observations we suggest the following algorithm for structural alignment: place an epsilon net in the relatively small space of all rigid transformations, and exhaustively search all its members for an optimal alignment. Since the algorithm is exhaustive, when it fails to find a good alignment, it is certain that none exists. We emphasize that our contribution is mostly theoretical, rather than practical.

Our solution applies to a broad class of relevant scoring functions, including, most

importantly, STRUCTAL [112] — a commonly used structural alignment score. The above algorithm finds the optimal transformations and correspondences, as defined by STRUCTAL score, in time complexity $O(n^{10}/\epsilon^6)$ for two globular proteins whose length is at most n. We also point out some of the difficulties involved with the (numerous) attempts to solve the structural alignment problem based on the internal distance matrices of the two proteins.

This approach also offers a way to visualize the structural alignment score as a function of the epsilon net of all rigid transformations. The visualization helps developing intuitions for designing better optimization algorithms and heuristics for protein structural alignment. We show scores of three examples: (a) two structures with a unique good alignment, (b) two structures with several good alignments, and (c) two structures that cannot be aligned. This work has been published in [67].

1.4.2 Comparison of structural alignment methods

The second research project is a large-scale computer experiment for comparing six publicly available protein structural alignment (heuristic) methods. We consider a set of 2,930 sequence diverse domains from CATH v.2.4 [87] and align all pairs of structures. The methods we test are (listed chronologically): (1) SSAP [118] (2) STRUCTAL [112, 34] (3) DALI [48, 49] (4) LSQMAN [61] (5) CE [104] and (6) SSM [69]. Each alignment has a significance (or native) score assigned by the program that found it and four geometric match measures (SAS, SI, MI, and GSAS – a variant of SAS that penalizes for gap openings). We first evaluate the above methods by comparing the ROC curves [41] based on their native score, and the four geometric match measures. We take the view that structural alignment methods are, in effect, optimizers of the geometric match measures, and create a better optimizer, the 'Best-of-All' method, which takes the best alignment found by all methods. Thus, we evaluate the performance of the programs by directly comparing the geometric match measures of their alignments. We also elaborate on problems in assuming the lack of structural similarity between structures that are classified differently by the gold standard. The total computer time used in this experiment is over 20,000 hours on a

2.8 GHz processor.

We argue that structural alignment methods should be evaluated by comparing the alignments they find. In our opinion, ROC curves are of limited value since that their ranking of the methods is not consistent with the ranking implied by the quality of the alignments the methods find. Also, there are problems inherent to comparing similarity of pairs of objects based on a hierarchical classification (namely a tree) of those objects: a classification attempts to group objects that share some property but does not guarantee that pairs of objects in different classes are indeed different. We find that the objective geometric match measures provide more relevant information than the native scores given by each particular method. Of the different measures we use, GSAS and SAS perform best in separating good alignments from less good ones. We consider the set of the pairs that are in the same CATH fold class as well as the set of all pairs and find that certain structural alignment methods consistently outperform other methods. More specifically we find that the 'Best-of-All' method (i.e., a combination of all six methods under study) finds the best results, followed by STRUCTAL and SSM. Finally, we identify a set of structurally similar pairs that are found only by a single method, providing a useful test set that will allow structural alignment methods to be improved. This work has been submitted for publication.

1.4.3 Efficient modeling of protein structure

In the third research project we construct economical approximation nets for protein structure. We begin by following Unger *et al.* [121] and Micheletti *et al.* [78], who use the unsupervised learning technique of clustering to identify small sets of protein fragments. We use our own clustering heuristic (documented in Appendix A) and find better libraries of fragments that approximate all proteins fragments well. These libraries construct more economical discrete approximations, or approximation nets, for protein structure. Indeed, as observed by Simon *et al.* [108], considering only protein models constructed from valid protein fragments yields smaller nets for structure space.

We study many different sized libraries of fragments of length 4, 5, 6, and 7. The

accuracy with which our discrete representations capture native structure depends on average number of states per residue (complexity) and varies from 1.9 Å for a 4-state model based on fragments of length 7 to 0.76 Å for a 15-state model based on fragments of length 5. With discrete representations a protein conformation is reduced to a string of symbols that define the local states (alphabets of 4 and 15 letters, respectively, in the above examples). These strings completely specify the conformation, and all conformations can be generated by considering all strings. Thus, discretization associates a three-dimensional structure with one-dimensional strings akin to the amino acid sequence. We find that longer fragments are more accurate as they include more correlation than shorter fragments. However, the complexity that can be explored with longer fragments is severely limited by the relatively small number of known protein structures.

Using our efficient approximation nets of protein structure, we address the first sub-problem mentioned above (Section 1.3.1) and generate candidate structures for *ab initio* structure prediction. When considering full (albeit small) proteins, the approximation net is too large to exhaustively enumerate; thus we resort to sampling, focusing on structures that admit to the secondary structure of the target protein. Despite the extreme simplicity of this method, the sets of decoys include many structures that have a cRMS deviation smaller than 6 Å from the native conformation. This method works well for small all-α proteins, and reasonably well for an α/β protein.

We also use our approximation nets for building loops for homology modeling. Since the loops are at most 14 residues long, we can exhaustively enumerate all candidates. For longer loops, we mitigate combinatorial explosion by simultaneously building from both ends of the gaps in the framework, and testing if the two parts join. The approximation nets have structures that have local (along the backbone) conformations that are similar to that observed in other proteins. Thus, our loops and decoys have protein-like local conformations. Part of this work has been published [64, 66] and the remainder is in press [65].

Chapter 2

Preliminaries

For the present work, a *protein* is a chain of atoms in three-dimensional space. Consider a protein A of n atoms, $A = (a_1, \ldots, a_n)$, with $a_i \in \mathbb{R}^3$. When needed, we assume without loss of generality that A is positioned with its center of mass at the origin, and is bounded by a box of dimensions $X^A \times Y^A \times Z^A$. It is known [43] that the volume of a protein is linear in the number of its residues, that is, $X^A \cdot Y^A \cdot Z^A = O(n)$. We also let R^A denote the radius of the bounding sphere of the protein A. In the special case of *globular proteins*, the size of the protein along all axis X^A, Y^A, Z^A is $O(n^{1/3})$ and $R^A = O(n^{1/3})$.

A *sub-chain* of the protein A is a subset of its atoms, arranged by their order of appearance in A. Denote the k-long sub-chain defined by $P = (p_1, p_2, \ldots, p_k)$, where $1 \leq p_1 < p_2 < \ldots < p_k \leq n$, by $A(P) = (a_{p_1}, a_{p_2}, \ldots, a_{p_k})$. A *gap* in the sub-chain is two consecutive indices p_i, p_{i+1} such that $p_i + 1 < p_{i+1}$.

Consider two proteins, A of n atoms and B of m atoms, and two sub-chains, P of protein A and Q of protein B; we assume, without loss of generality, that $n \geq m$. We call two sub-chains P and Q of equal length, $|P| = |Q|$, a *correspondence*, or a *match*. Thus, a correspondence associates pairs of atoms from two proteins that appear in the same position in their respective sub-chains. Note that while the two complete proteins can differ in length, the sub-chains cannot. In the world of protein sequences, the analogous term is *alignment*; at times, we use it here too, interchanged with correspondence. The number of gaps in a correspondence, denoted $G_{P,Q}$, is the

sum of the number of gaps in P and Q.

Next, we describe the structural alignment problem, or detecting structural similarities between two proteins. We begin with an imprecise description of the problem, and follow with the details that allow a precise definition.

Structural Alignment Problem: Given two proteins A and B, find two sub-chains P and Q of equal length such that

1. $A(P)$ and $B(Q)$ are similar
2. The correspondence length $|P| = |Q|$ is maximal under condition 1.

A protein can be rigidly transformed (i.e. rotated and translated) without affecting its inherent structure. Rotations and translations are each specified by three parameters [17]. Since we are interested in the relative position and orientation of the two proteins, we can hold A fixed, and only transform B; the rigidly transformed B is denoted by $\hat{B} = (\hat{b}_1, \ldots, \hat{b}_m)$. The relative position and orientation of the proteins are useful for solving the protein alignment problem.

There are various measures of similarity, or deviation, between two sub-chains. Among the more commonly used are dRMS (distance root mean squared) deviation and cRMS (coordinate root mean squared) deviation.

For sub-chains P and Q of length k, dRMS is defined as

$$\text{dRMS} = \left(\frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} (\|a_{p_i} - a_{p_j}\| - \|b_{q_i} - b_{q_j)}\|)^2 \right)^{1/2} \tag{2.1}$$

and cRMS is defined as

$$\text{cRMS} = \min_{\hat{B}} \left(\frac{1}{k} \sum_{i=1}^{k} \|a_{p_i} - \hat{b}_{q_i}\|^2 \right)^{1/2} \tag{2.2}$$

where $\hat{B}$ is the image of protein B under a rigid transformation. Given the correspondence P and Q, the transformation that achieves the minimum cRMS can be found in closed form. The closed form solution to this problem is known in different communities under different names: in structural biology this is Kabsch's procedure [57];

in computer science this is attributed to Horn [52].

The general structural alignment problem gives rise to a family of concrete optimization problems. In these, we optimize a match measure that is based on the geometric properties of the alignment. Intuitively, it should favor alignments with many matched residues, that have small deviation of the two sub-chains, and few gaps. Note that cRMS and dRMS measure only deviation and must, therefore, be transformed to favor longer correspondences. These sought properties are not independent. For example, a lower cRMS deviation can always be achieved by selecting a shorter match; given the fixed inter-C^{α} distance there is the extreme case of many alignments of just two residues that have cRMS deviation of 0 Å. Also, by allowing additional gaps, the alignment can be lengthened without necessarily increasing the cRMS deviation.

An important score, which we study in detail, is the STRUCTAL [112] score. It is closer in spirit to cRMS in that it compares matching pairs of the correspondence, and considers the rotated and translated position of the structures. It also penalizes for gaps in the correspondence.

$$\text{STRUCTAL-score}_{P,Q} = \max_{\hat{B}} \sum_{i=1}^{k} \frac{20}{1 + \|a_{p_i} - \hat{b}_{q_i}\|^2/5} - 10 \cdot G_{P,Q} \tag{2.3}$$

It is an instance of a family of STRUCTAL-type scores:

$$\text{STRUCTAL-type-score}_{P,Q} = \max_{\hat{B}} \sum_{i=1}^{k} \frac{C_1}{C_2 + \|a_{p_i} - \hat{b}_{q_i}\|^2} - C_3 \cdot G_{P,Q} \tag{2.4}$$

where C_1,C_2, C_3 are positive constants (in STRUCTAL these are $C_1 = 100$, $C_2 = 5$ and $C_3 = 10$). When using this score, we seek a rigid transformation and a correspondence that achieves a *maximal* (rather than minimal) value.

We rely on four additional scores, or match measures: Similarity Index (SI) [62], Match Index (MI) [62], Structural Alignment Score (SAS) [112] and a measure which we define denoted Gapped SAS, or GSAS. These match measures are not necessarily easy to optimize. However, given several alignments they are easy to calculate and

can be used to evaluate alignments. The original Match Index (OMI) [62], has values between 0 and 1, with better alignments having higher values; instead we take MI = 1 − OMI, so that lower values always imply better alignments. We also introduce GSAS — a variant of SAS that penalizes gap openings. MI includes a normalizing factor $w_0 = 1.5$ (following Krissinel's suggestion, see http://www.ebi.ac.uk/msd-srv/ssm/comparisons/cmp_index.html).

The explicit definition of these measures in terms of cRMS, k, n, m and $G_{P,Q}$ ($n \geq m$) is:

$$\text{SI} = \frac{\text{cRMS} \cdot m}{k} \tag{2.5}$$

$$\text{MI} = 1 - \frac{1+k}{(1+\text{cRMS}/w_0) \cdot (1+m)} \tag{2.6}$$

$$\text{SAS} = \frac{\text{cRMS} \cdot 100}{k} \tag{2.7}$$

$$\text{GSAS} = \begin{cases} \dfrac{\text{cRMS} \cdot 100}{k - G_{P,Q}} & \text{if } k > G_{P,Q} \\ 99.9 & \text{else} \end{cases} \tag{2.8}$$

A GSAS value of 99.9 denotes worst possible value. The number "100" used above is in units of number of aligned residues. Consequently, the units of SI, SAS and GSAS are Å.

Chapter 3

Protein structural alignment

Alignment of protein structures is a fundamental task in computational molecular biology. Good structural alignments can help detect distant evolutionary relationships that are hard or impossible to discern from protein sequences alone. We study the structural alignment problem as a family of optimization problems and develop an approximate polynomial time algorithm to solve them. For a commonly used scoring function - STRUCTAL score [112], the algorithm runs in $O(n^{10}/\epsilon^6)$ time, for globular proteins of length n, and detects alignments that score within an additive error of ϵ from all optima. Thus we prove that this task is computationally feasible, though the method we introduce is too slow to be a useful everyday tool. The measurement of similarity between a pair of protein structures used by our algorithm uses the Euclidean distance between the structures (appropriately rigidly transformed). We also show that an alternative approach, which relies on internal distance matrices, must incorporate sophisticated geometric ingredients if it is to guarantee optimality and run in polynomial time. We use these observations to visualize the scoring function for several real instances of the problem. Our investigations yield insights on the computational complexity of protein alignment under various scoring functions; these insights can be used in the design of new scoring functions for which the optimum can be approximated efficiently, and perhaps in the development of efficient algorithms for the multiple structural alignment problem.

In line with our perspective that this is mostly a theoretical study, we use the O

notation freely. Recall that the notation $g(n) = O(f(n))$ means at most $cf(n)$ for a constant c independent of n.

We investigate scoring functions in Section 3.1. In Section 3.2 we consider the space of alignments for three specific pairs of proteins and draw our conclusions in Section 3.3.

3.1 Approximate structural alignment

We focus on scores that evaluate the similarity of two structures by explicitly applying a rigid transformation to one, and then comparing the transformed structure with the other. For such scores, the optimization problem is to find transformations and correspondences of (near) optimal score.

The polynomial time algorithm we present, calculates the optimal score for a substantial number of rotations and translations. It then sifts through these scores to find the best ones, i.e., the pairs (transformation, correspondence) with near globally maximum scores. For the algorithm to run in polynomial time, two conditions must hold:

- Given a fixed transformation, it should be possible to find in polynomial time an optimal correspondence. We elaborate on this in Subsection 3.1.1.
- The number of rigid transformations under consideration must be bounded by a polynomial. This issue is addressed in Subsection 3.1.2.

Next, we define guidelines that can be used in designing novel scoring functions for structural alignment; scores that follow these guidelines come hand in hand with a polynomial time algorithm that finds all near optimal alignments. The following guidelines are useful as researchers are still far from a thorough understanding of the desirable characteristics of scoring functions for this problem. Examples of obvious interesting options that are yet to be investigated include: variable gap penalties depending on the gap's location within the structure (e.g., higher penalty inside a helix), and scores that take into account sequence information.

3.1.1 Separability of scoring function

As mentioned above, we are assuming that for a fixed rotation and translation, the optimal correspondence can be determined in polynomial time. This requirement strongly points to scores that can be optimized using dynamic programming. When applicable, a dynamic programming algorithm finds in polynomial time an optimal solution among an exponential number of potential correspondences. Score functions that are amenable to dynamic programming must satisfy two requirements: (1) Optimal substructure: the restriction of an optimal correspondence to any substructure is itself an optimal correspondence of the substructure (2) The space of relevant subproblems is small (polynomial). For more details see [16]. Scores that satisfy these conditions are called *separable*. STRUCTAL-type scores (defined in Equation 2.4) and STRUCTAL in particular, are separable and the optimal correspondence can be determined using dynamic programming in $O(n^2)$ time and space [112, 34].

3.1.2 Lipschitz condition on scoring functions

Here, we provide conditions on a scoring function under which its overall behavior can be approximated by evaluating it only polynomially many times. Recall that a scoring function assigns a value to every correspondence *and* rotation and translation. We present a scheme for approximating all rotations and translations whose (local) optimal correspondence is near the global optimum.

A rigid transformation in $\mathbb{R}^3$ consists of a translation and a rotation. A translation is parameterized by a vector (t_x, t_y, t_z) in $\mathbb{R}^3$. Here, it suffices to have t_x, t_y, t_z range over $[-(X^A + X^B)/2, (X^A + X^B)/2]$, $[-(Y^A + Y^B)/2, (Y^A + Y^B)/2]$ and $[-(Z^A + Z^B)/2, (Z^A + Z^B)/2]$ respectively (recall that X^A, Y^A, Z^A and X^B, Y^B, Z^B are the sides of the bounding boxes of protein A and protein B, respectively). There are many ways to parameterize the group of rotations SO(3). For the purpose of our proofs we represent each rotation by three angles (r_1, r_2, r_3) in the range $[0, 2\pi]$ [17]; this constitutes a 4-fold cover of SO(3). For the purpose of sampling (see section 3.2), we use the parametrization of rotations by means of unit norm quaternions (unit vectors in $\mathbb{R}^4$) [106]: a rotation by an angle θ about the normalized vector (n_x, n_y, n_z)

is described by $(n_x \cos\theta/2,\ n_y \cos\theta/2,\ n_z \cos\theta/2,\ \sin\theta/2)$. This representation has the advantage that angular separation of quaternions captures the natural metric of SO(3). In this representation, opposite points in S^3 are identified, reflecting the projective topology of SO(3). Equivalently it suffices to consider only half of S^3.

Consider a specific correspondence between sub-chains P of protein A and Q of protein B. Fix protein A in space. For every rigid transformation of protein B, one can compute the correspondence-dependent scoring function, denoted *CDS*, using the distances between corresponding atom pairs in space. We index the real-valued CDS function by P and Q and denote it $\mathrm{Sc}_{P,Q}$.

The CDS function is defined over the space of all rigid transformations. Note that there are exponentially many correspondences and thus exponentially many CDS functions defined in this manner. Specifically, there are $\sum_{i=1}^{\min(n,m)} \binom{n}{i}\binom{m}{i}$ functions. In this paper, we are concerned with a scoring function of the form:

$$F(r_1, r_2, r_3, t_x, t_y, t_z) = \max_{|P|=|Q|} \mathrm{Sc}_{P,Q}(r_1, r_2, r_3, t_x, t_y, t_z) \tag{3.1}$$

This is the *upper envelope* of all CDS functions (see Figure 3.1).

The following lemma gives conditions on the scoring function. When these hold, it is possible to derive good approximations of all of the function's near-optimal maxima from only polynomially many evaluations of the function.

Definition 1. *A CDS function* $\mathrm{Sc}_{P,Q}$ *satisfies a* coordinate-wise Lipschitz conditions *with values* c_r *and* c_t *if for all rigid transformations* $\vec{p} = (r_1, r_2, r_3, t_x, t_y, t_z)$ *and for all* $\delta > 0$,

$$|\mathrm{Sc}_{P,Q}(\vec{p} + \delta \cdot \vec{e_r}) - \mathrm{Sc}_{P,Q}(\vec{p})| \leq c_r \delta$$
$$|\mathrm{Sc}_{P,Q}(\vec{p} + \delta \cdot \vec{e_t}) - \mathrm{Sc}_{P,Q}(\vec{p})| \leq c_t \delta$$

where $\vec{e_1}, \ldots, \vec{e_6}$ *are the standard basis vectors in* $\mathbb{R}^6$, $\vec{e_r} \in \{\vec{e_1}, \vec{e_2}, \vec{e_3}\}$, *and* $\vec{e_t} \in \{\vec{e_4}, \vec{e_5}, \vec{e_6}\}$.

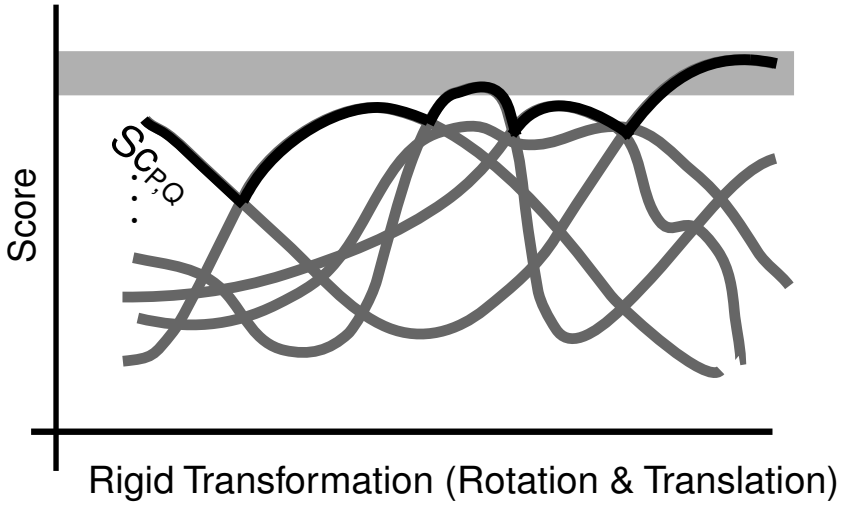

Figure 3.1: A schematic view of the scoring functions, parameterized by the rigid transformation. The (exponentially many) correspondence dependent scoring (CDS) functions are depicted in dark gray, and their upper envelope (defined by Equation 3.1) is marked in black. We are concerned with all solutions in the light gray region: top values in the upper envelope.

Lemma 3.1.1. *Let the CDS functions satisfy coordinate-wise Lipschitz conditions with values c_r, c_t. For every $\epsilon > 0$, there exists a finite set $G = G(\epsilon)$ of rotations and translations such that:*

1. $|G| = O\left(\frac{nc_r^3 c_t^3}{\epsilon^6}\right)$
2. *For every choice of a translation and a rotation $\vec{p}$, there is a point $\vec{p}_G \in G$ with $|F(\vec{p}) - F(\vec{p}_G)| \leq \epsilon$*

We refer to the set G as an ϵ-net for the scoring function.

Proof. The set $G(\epsilon)$ is the product of 6 sets of equally spaced points in each of its six dimensions. In the 3 dimensions of rotations, the spacing is $\delta_r = \epsilon/3c_r$; the size of the set in each dimension of rotation is $O(c_r/\epsilon)$. Similarly, in the 3 dimensions of translation the spacing is $\delta_t = \epsilon/3c_t$ and the size of the set is $O((W^A + W^B)c_t/\epsilon)$, where $W = X, Y, Z$ respectively. Taking into account that a protein of n residues satisfies $X \cdot Y \cdot Z = O(n)$ (see Chapter 2), the total size of G follows.

The coordinate-wise Lipschitz condition for each CDS function $\mathrm{Sc}_{P,Q}$, implies that the same condition holds for their upper envelope F. Given a point $\vec{p}$ in rotation and translation space, the nearest point $\vec{p}_G \in G$ can be reached by moving at most $\delta_r/2$ along each of the 3 dimensions of rotation and $\delta_t/2$ along each of the 3 dimensions of translation. Because the upper envelope F satisfies Lipschitz, the change in value of F induced by each such step is at most $\delta c_r/2$ in the first case and $\delta c_t/2$ in the latter. Thus, the overall change is at most $3c_r\delta_r/2 + 3c_t\delta_t/2 = \epsilon/2 + \epsilon/2 = \epsilon$. □

The above lemma suggests the following algorithm to find all points with near-maximal values of the scoring function F. Let M be the global maximum of F, and call a point $\vec{p}$ ϵ-maximal if $F(\vec{p}) \geq M - \epsilon$. For every point $\vec{p}$ that is ϵ-maximal, there is a nearby point $\vec{p}_G \in G$ with $F(\vec{p}_G) \geq F(\vec{p}) - \epsilon$. We would like every ϵ-maximal point to be accounted for by a nearby point in G. Given ϵ, evaluate F on all points of $G(\epsilon)$ defined above. Next, select the subset of these points within 2ϵ from the maximal value found. This will guarantee finding approximations to all ϵ-maximal points, satisfying our requirement.

The ϵ-net here is a slightly different from the one defined in the introduction. Let M be the maximal value of F. We consider the connected components of the set of rigid transformations with F values in the range $[M - \epsilon, M]$ (often referred to as $F^{-1}([M - \epsilon, M])$). The set G intersects, or is a net for, all these sets.

STRUCTAL type scores

Lemma 3.1.2 guarantees that all STRUCTAL-type scoring functions (for definition see Equation 2.4) are well behaved and can thus be approximated by a polynomial sized net. Namely, STRUCTAL-type scoring functions can be approximated to ϵ-accuracy with a net of size $O(\frac{n^8}{\epsilon^6})$ for globular proteins and $O(\frac{n^{10}}{\epsilon^6})$ for non-globular ones. This, of course, also holds for the specific case of STRUCTAL score (defined in Equation 2.3). In particular, we show

Lemma 3.1.2. *Any CDS function of the form*

$$\sum_{i \in \text{correspondence}} \frac{C_1}{C_2 + \|a_i - b_i\|^2}$$

satisfies the Lipschitz condition of lemma 3.1.1 with $c_t = O(n)$ *and* $c_r = O(n^{4/3})$ *in case of globular proteins. For non-globular proteins* $c_r = O(n^2)$.

Proof. First consider a single pair of atoms a and b, and their contribution to the scoring function. A translation of b by δ along any axis can change $\|a - b\|$ by at most δ. A rotation of b by δ around any axis can change $\|a - b\|$ by at most $R\delta$. The function $\phi(x) := C_1/(C_2 + x^2)$ has a bounded derivative $|\phi'(x)| \leq M(C_1, C_2) = M$. By the mean value theorem, it follows that a change of δ in any of the six coordinates will result in a change of at most $MR\delta + M\delta$ in the scoring function, the first term being due to rotations and the other to translations. There are at most n contributions to a CDS-function (n is the number of atoms in the longer protein), so the total change is bounded by $nMR\delta$ from rotations and $nM\delta$ from translations. For globular structures, $R = O(n^{1/3})$, and in general $R = O(n)$. We have shown that the coordinate-wise Lipschitz condition is satisfied with $c_r = O(n^{4/3})$ for globular proteins and $c_r = O(n^2)$ for non-globular ones. □

Altogether, finding approximations to all ϵ-near optimal points for STRUCTAL-type scoring functions takes $O(n^{10}/\epsilon^6)$ time for globular proteins. This is due to $O(n^8/\epsilon^6)$ evaluations of the scoring function at the points of $G(\epsilon)$, each evaluation taking $O(n^2)$ time. More generally, our scheme is an approximate polynomial algorithm for every separable scoring function that requires only polynomially many evaluation points (see Lemma 3.1.1). Notice that this scheme gives all near optimal function values, rather than all near optimal alignments. It would be interesting to determine whether the task of finding all near optimal *correspondences* can be efficiently solved.

3.1.3 A closely related NP-hard problem

Associated with every protein chain A of n atoms is an $n \times n$ real symmetric matrix D, where $D(i, j)$ is the Euclidean distance in $\mathbb{R}^3$ between the ith and jth atoms of A. This matrix is called "the (internal) distances matrix" and is invariant under rigid and mirror transformations of the protein. The internal distances matrix that corresponds

to a sub-chain S of the protein A is the sub-matrix (minor) of D consisting of the rows and columns indexed by the elements of S.

The two representations of a protein, by atomic coordinates and by the internal distances matrix, are of course closely related. Calculating the distances matrix from the atomic coordinates is easy (and takes quadratic time). It is also known that the protein's coordinates can be recovered in polynomial time from the distances matrix, using distance geometry [45]. This calculation is possible because proteins lie in a 3-dimensional Euclidean space. The recovered atomic coordinates are the original ones, modulo a rigid (and possibly a mirror) transformation.

It follows that any algorithm that uses either of these two representations can be converted into one that uses the other. The conversion is straightforward: add a pre-processing and a post-processing step that translate, in polynomial time, between representations.

The internal distances matrix representation of proteins may seem attractive because it limits the search to the correspondences, without need to optimize on the rigid transformations. Methods that use the internal distance matrix representation directly compare pairs of sub-matrices and optimize a measure that is derived from dRMS deviation (for definition of dRMS see Equation 2.1). Once the correspondence is found, the rigid transformation that optimally superimposes the two substructures can be recovered with Kabsch's procedure [57]. It is generally considered a minor problem that the final positioning and orienting of the structures optimizes the cRMS deviation, while the correspondence optimized a different measure (dRMS).

We point out that a correct and efficient solution to the approximate structural alignment problem must exploit the fact that proteins lie in 3-dimensional Euclidean space. In particular we show that a slightly generalized problem, where the internal distances come from a general metric space (not necessarily Euclidean) is NP-hard. We define in Lemma 3.1.3 a particular scoring function to focus the discussion; a similar argument applies to variants of this scoring function.

Intuitively, the problem is hard because all (exponentially many) pairs of subchains are potential solutions. Notice, that if we restrict the number of gaps by a constant, there are only polynomially many potential solutions, and this substantially

reduces the computational complexity of the problem. The CLIQUE problem is well-known to be NP-hard [88]: the input is a graph and an integer k, the output should be either a k-clique, or if there is no such clique, the answer 'no'. We reduce the CLIQUE problem to the problem at hand to demonstrate its hardness, that is we show how an algorithm that finds an ϵ-approximation to the optimal solution, can be efficiently used to solve CLIQUE.

Lemma 3.1.3. *Let D^A, D^B be distance matrices of two metric spaces (think of them as the internal distance matrices of chains A and B).*

Let the score of two sub-chains P and Q, of equal length $|P| = |Q| = k$, be

$$\mathrm{Sc}_{P,Q} = \sum_{i=1}^{k} \sum_{j=1, j\neq i}^{k} 2/(1 + (D^A(p_i, p_j) - D^B(q_i, q_j))^2).$$

For every $0 < \epsilon < 1$, it is NP-hard to find sub chains that are within ϵ from the optimal score.

Proof. Given a graph $G = (V, E), |V| = n$ we "construct" two chain structures and use the algorithm for finding correspondences of near-optimal scores. The first structure, denoted S^A, has n "atoms" and encodes the graph G: each vertex is associated with an atom (using some ordering) and the distance between two atoms is the length of a shortest path in G between the two corresponding vertices. The internal distances matrix associated with this structure is an $n \times n$ matrix D^A where $D^A(i, j)$ is the length of the shortest path from v_i to v_j. The second structure, denoted S^B, has k "atoms"; it encodes a clique of size k. The internal distances matrix in this case, denoted D^B has zeros on the diagonal and ones elsewhere. If the score is strictly greater than $2(k)(k-1) - 1$ return the subset of S^A (a k-clique); otherwise return 'no'. Since S^B has k atoms, the score is a sum of at most $k(k-1)$ terms. Also, the distances are integers, and this restricts the possible values of the terms in the sum; 2 and 1 are the two largest values. Thus, if a good score is found it is greater or equal to $2(k)(k-1)$. This optimal value is achieved when a k-clique in S^A is found; otherwise, there is clearly no k-clique. □

An algorithm that solves the approximate structural alignment problem using only

the metric properties (and not the fact that is a three-dimensional Euclidean metric) also solves the above generalized problem. This implies that it either fails to find optimal approximations, or it is inefficient (or that P=NP which is generally viewed as unlikely).

To summarize, the problem in finding good correspondences is that there are (exponentially) many potential candidates. The number of possibilities can be greatly reduced because the structures lie in three-dimensional Euclidean space and the scores are separable. However, if these restrictions are removed, an exponential blow-up in computational complexity seems unavoidable.

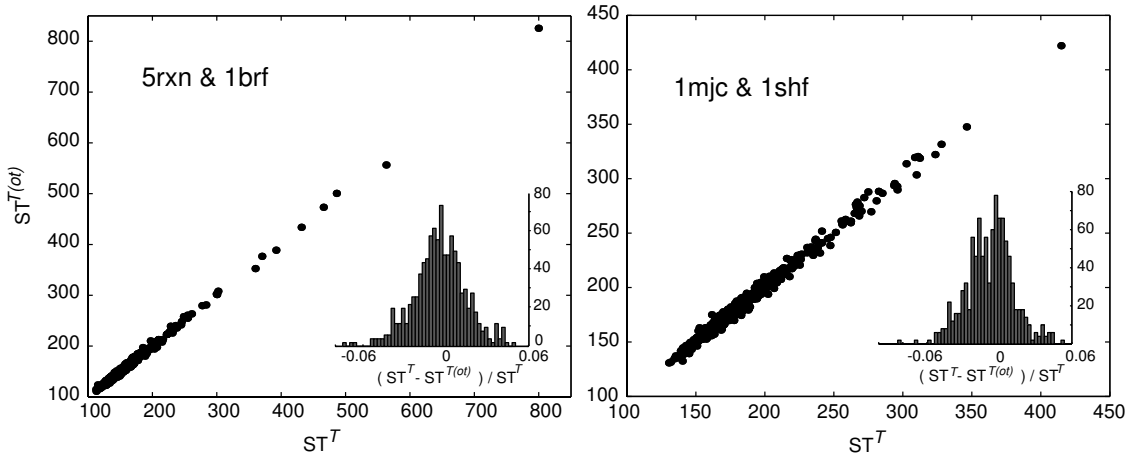

Figure 3.2: Reducing the net of translations to $T(\mathrm{ot})$ (the faster heuristic we use), for 1000 random rotations. The x-axis is the best score with respect to all translations over a $50 \times 50 \times 50$ net. The y-axis gives the best score when an atom of one protein must be placed on top of an atom of the other. The inserted histogram shows the distribution of $(ST^T - ST^{T(\mathrm{ot})})/ST^T$. The proteins in this example are 5rxn & 1brf and 1mjc & 1shf. This heuristic finds only very slightly worse scores, and behaves well for practical purposes

3.2 Visualizing the STRUCTAL score

We examine the properties of the STRUCTAL score for structural alignment of various pairs of proteins; we focus on the domain of rotations, while optimizing the translation parameters. By the above observations, it suffices to calculate the STRUCTAL score on a net in the space of transformations. Let R be a net for the space of rotations, T a net for translations and $R \times T$ the net for all rigid transformations. Ideally, we would like to visualize the STRUCTAL score over $R \times T$ to determine all (near) maxima. Visualizing a function of six parameters is of course very hard. We thus focus our attention on the three-dimensional space of rotations and define

$$\mathrm{ST}^T(r_1, r_2, r_3) = \max_{(t_x,t_y,t_z)\in T}(\mathrm{STRUCTAL}(r_1, r_2, r_3, t_x, t_y, t_z)).$$

The advantage of focusing on ST^T is that it can be visualized; the disadvantage is that multiple maxima due to translation changes alone, are hidden. Choosing r_1, r_2, r_3 as the three function parameters is somewhat arbitrary; we could have alternatively considered functions over different parameters, e.g.,

$$\mathrm{ST}^R(t_x, t_y, t_z) = \max_{(r_1,r_2,r_3)\in R} \mathrm{STRUCTAL}(r_1, r_2, r_3, t_x, t_y, t_z).$$

In order to reduce the time for exploring the scoring function over the space of rotations we heuristically calculate the maximum over a smaller set of translations, denoted $T(\mathrm{ot})$. $T(\mathrm{ot})$ is the set of translations that position an atom from protein A exactly *on top* of an atom from protein B, $|T(\mathrm{ot})| = O(nm)$. This speeds up the calculation by a factor of $O(n^2)$. The sets T and $T(\mathrm{ot})$ are different; the maximum over T can be higher or lower than the maximum over $T(\mathrm{ot})$. However, we assume that the *best* translation in T positions at least one atom from A on top (or close to) an atom from B, implying that $\mathrm{ST}^{T(\mathrm{ot})}$ and ST^T reasonably approximate each other. In Figure 3.2 we compare of the values of ST^T for a set of size $50 \times 50 \times 50$ and $\mathrm{ST}^{T(\mathrm{ot})}$ for 1000 random rotations, and demonstrate that the two scores indeed approximate each other well.

We parameterize the group of rotations using quaternions [106]. Notice that a

Cartesian product of longitude and latitude nets is a net for a sphere S^2 in R^3. Here, we use a longitude net that is uniformly spaced and a latitude net that uniformly spaces its cosine values. This net can be visualized in the plane by placing the longitude values along the x axis and the latitude values along the y axis. Note that unlike the sphere, this display does not show the wrapping around of the longitude values; it also has a distortion around the poles. Figure 3.3 shows examples of three-dimensional spheres, overlayed with their net points and a planar layout of the nets. The two-dimensional spheres can be sampled more efficiently (e.g., [100]); our sampling scheme allows easy planar visualization of the score function values on the spheres.

We visualize the function $\mathrm{ST}^{T(\mathrm{ot})}$, for specific pairs of proteins, using short videos. The position in the frame sequence serves as an additional dimension (aside of the two planar coordinates). The data figures show an (ordered) subset of the video frames. The full videos are available online[1]. Denote a unit quaternion in $\mathbb{R}^4$ by $\vec{q} = (x, y, z, w)$ ($\|q\| = 1$). The set of unit quaternions of a fixed w is a sphere S^2 in R^3 of radius $\sqrt{1 - w^2}$. Each frame in the video has a fixed w value, which determines the width of the frame and its position in the sequence. Since we are concerned only with half of S^3, w is equally spaced in the interval $[-1, 0]$. Varying w, we place a net on the corresponding sphere and evaluate the scoring function at all net points. Then, the score values are visualized in the plane as described above, using a color scale ranging from blue (low) to red (high). The number of points on a net varies with the area it covers, totaling in approximately 10^6 points. Lastly, the width of the displayed planar net depends on the radius of the sphere, as shown in Figure 3.3.

We examine three types of behaviors of the STRUCTAL scoring function when aligning pairs of proteins. Figure 3.4 depicts the score when considering two proteins with a single meaningful maximum: 5rxn and 1brf (SCOP fold classification Rubredoxin-like). This figure shows a clear, single, high-scored maximum yielding a good alignment. Figure 3.5 shows the scoring function for two proteins with several meaningful maxima: 1mjc and 1shf-a (SCOP fold OB and SH3-like barrel respectively). This example is listed in the work of Zu-Kang and Sippl [132]. Lastly,

[1] `http://csb.stanford.edu/rachel/structal_alignment`

Figure 3.6 shows the scores when aligning two structurally different proteins: 1jjd and 1dme (both SCOP fold Metallotheionein). In this case, there are only rotations with low scoring alignments, proving there are no good alignments. A closer examination of the different maxima in Figure 3.5 shows that the multiple high-scoring orientations are due to an internal symmetry in the structures 1mjc and 1shf-a; these alignments (sequences and superpositioning of the structures) are shown in Figure 3.7. This symmetry, coupled with the two structures being fairly similar to each other, accounts for the multiple orientations that position many atoms from one structure near corresponding atoms from the other.

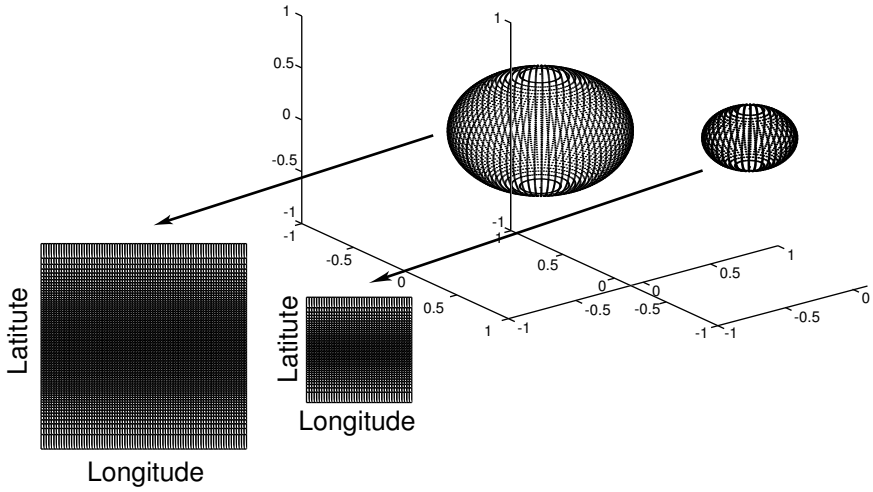

Figure 3.3: Visualization of score values for a net of discrete rotations. We model rotations using quaternions: each rotation is a 4-dimensional vector of unit length. We consider a net that covers the space of quaternions, or the unit sphere S^3 in $\mathbb{R}^4$. We use a net that is the union of nets on a discrete set of 3D spheres. For each 3D sphere we use a Cartesian product of longitude and latitude values and plot it in 2D. The width of the 2D plot varies with the radius of its corresponding sphere. The scores are described using color (specific values and colors are omitted from this figure, but used in Figures 3.4, 3.5 and 3.6). Notice that there is a distortion associated with this display, especially around the poles. Two spheres, overlayed with their net points and their corresponding 2D plots are shown above.

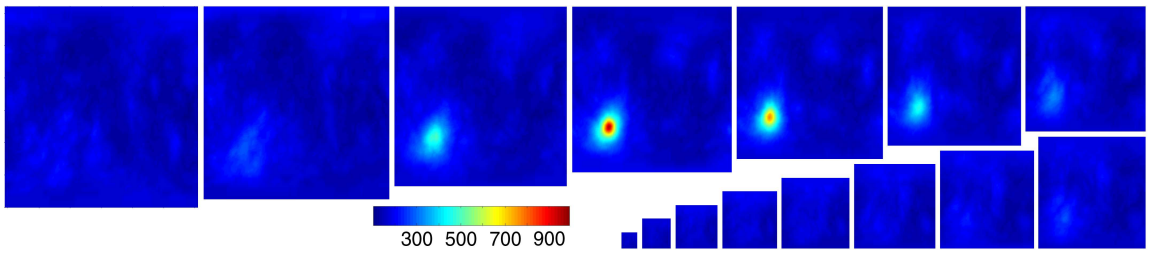

Figure 3.4: Example of a pair of structures with a single meaningful alignment. We plot the $ST^{T(ot)}$ score for aligning 5rxn (54 residues) and 1brf (53 residues) over the space of rotations. These proteins have the same SCOP fold classification - Rubredoxin-like, and each has 3 beta-strands and 3 helices. The $ST^{T(ot)}$ score function has a single maximum, implying one meaningful way of aligning the pair. The maximal score found is 993, aligning 53 residues to 0.797Å cRMS.

3.3 Discussion

We have presented a polynomial scheme for protein structural alignment. Exploring the space of rigid transformations solves this problem efficiently because it exploits the fact that proteins reside in a 3-dimensional Euclidean space. It seems unclear how to incorporate this crucial information if one phrases the problem via internal distances matrices. Unless 3-dimensionality is taken into account the problem becomes significantly harder (NP-hard). We found sufficient conditions for a scoring function so that all optima can be found in polynomial time. Devising novel scoring functions that detect biologically significant sub-structures is still an open area of research.

Experiments with the STRUCTAL scoring function on several pairs of proteins suggest that this scoring function is "well-behaved" on the domain of rotations. Studying the landscape of various scoring functions can prove valuable for the purpose of developing robust and efficient tools for structural alignment.

Note that an immediate extension of this algorithm solves multiple structural alignment. Namely, sampling the space of rigid transformations and finding the maximum using dynamic programming can find all approximate global maxima of the upper envelope of the CDS functions. For a fixed, small number of globular proteins, it is a polynomial algorithm (e.g., for three globular proteins it takes $O(n^{19}/\epsilon^{12})$ time).

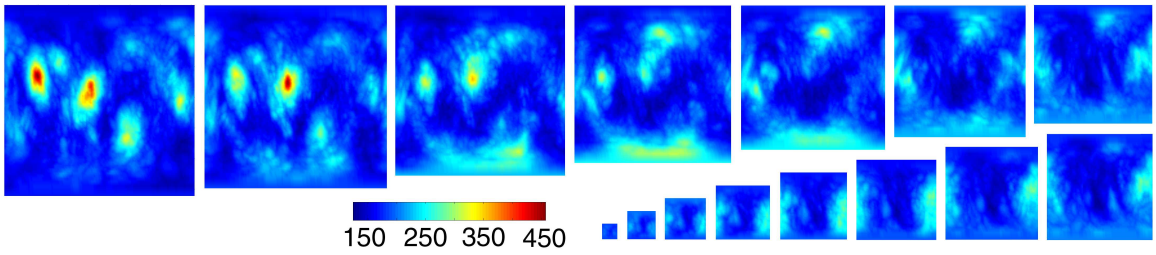

Figure 3.5: Example of a pair of structures with two meaningful alignments (this example was noted by Zu-Kang and Sippl [132]). We plot the $ST^{T(ot)}$ score for aligning 1mjc (69 residues) and 1shf (59 residues). The two maxima can be clearly seen, as well as additional less significant maxima. One of the two best alignments scores 458, aligning 41 residues to 2.89Å cRMS; the other scores 454, aligning 38 residues to 2.52Å cRMS. These proteins are of the same SCOP class - all-beta, and a different SCOP fold (OB and SH3-like barrel respectively).

Multiple structural alignment is a wide open problem and, although the direct extension has prohibitive running time, the analysis described here offers means of tackling it.

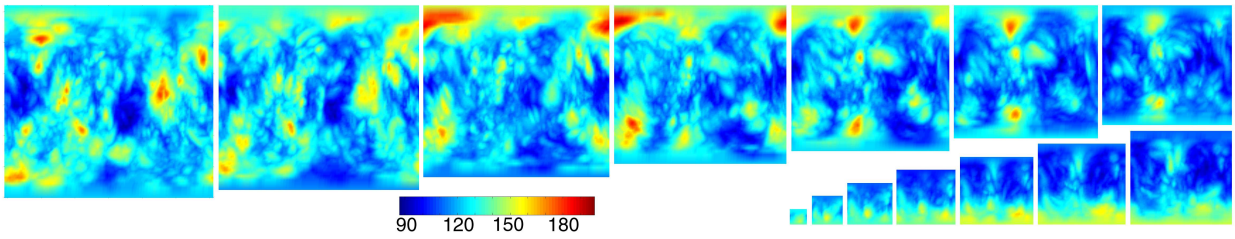

Figure 3.6: Example of a pair of structures without a clear alignment. We plot the $ST^{T(on)}$ score for aligning 1jjd (52 residues) and 1dme (28 residues) over the space of rotations. The scoring function landscape has two notable properties: (1) All score values are low (less than 200) even when taking into account the length of the proteins (2) There are no clear maxima, or equivalently, there are many bad solutions. Although these proteins have the same SCOP fold classification - Metallotheionein (SCOP class small proteins) it seems they cannot be aligned in a meaningful way.

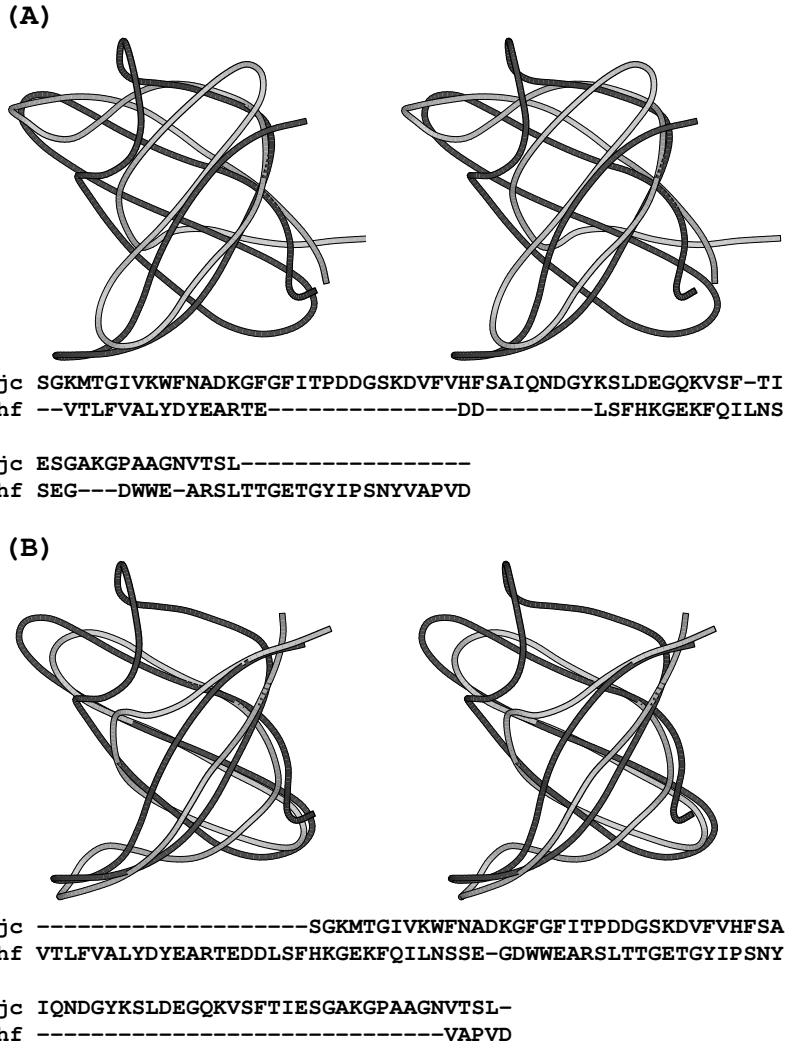

Figure 3.7: The optimal structural alignments of 1mjc and 1shf (superimposed structures and sequence); these are the two significant peaks seen in figure 6. 1shf is drawn in dark gray and 1mjc in light gray. Alignment (A) has STRUCTAL score 458, aligning 41 residues to 2.89 Å cRMS; alignment (B) has STRUCTAL score 454 aligning 38 residues to 2.52 Å cRMS. These alignments are equally significant, yet align different parts of the proteins. For clarity, both alignments are drawn in stereo and the proteins backbone is depicted as coil. The figure was created by MOLSCRIPT [68]

Chapter 4

Comparison of structural alignment methods

We report the largest and most comprehensive comparison of protein structural alignment heuristic methods. Specifically, we evaluate six publicly available structure alignment heuristic programs: SSAP [118], STRUCTAL [112, 34], DALI [48, 49], LSQMAN [61], CE [104] and SSM [69] by aligning all 8,581,970 protein structure pairs in a test set of 2,930 protein domains selected from CATH v.2.4 [87] to ensure sequence diversity. Evaluating structural alignment methods is not straightforward. First, we compare the rates of true and false positives using Receiver Operating Characteristic (ROC) [41] curves with the CATH classification taken as a gold standard. This proves unsatisfactory in that the quality of the alignments is not taken into account: sometimes a method that finds less good alignments scores better than a method that finds better alignments. We correct this intrinsic limitation by using four different geometric match measures (SI, MI, SAS, and GSAS) to evaluate the quality of each structural alignment. With this improved analysis we show that there is a wide variation in the performance of different methods; the main reason for this (as discussed in Chapter 3) is that it can be difficult to find a good structural alignment between two proteins even when such an alignment exists. Methods that do best in our study are neither the most popular nor those that are generally accepted to work well. We find that STRUCTAL and SSM perform best, followed by LSQMAN and

CE. Our focus on the intrinsic quality of each alignment allows us to propose a new method, called 'Best-of-All', that combines the best results of all methods.

4.1 Structural alignments data

We align all 4,290,985 pairs in a set of 2,930 sequence-diverse CATH v.2.4 domains (as pairs (i, j) and (j, i) are treated once) using all six methods. The programs output the alignment native score, the number of residues matched, and the cRMS deviation (Equation 2.2). We count the number of gaps in both structures by inspecting the alignment, which is available for all programs except for the standalone version of SSM. Based on this data, we calculate the geometric match measures SI (Equation 2.5), MI (Equation 2.6), and SAS (Equation 2.7) for all methods, and calculate GSAS (Equation 2.8) for all methods except SSM. Using the geometric match measures we create a seventh method denoted 'Best-of-All', which returns the best alignment found by all the other methods.

There are two types of comparisons: (1) Alignments found by different programs for the same pair(s) of structures. Here, we can only rely on the match measures, which use the geometric and other properties of the particular alignments (cRMS, number of matched residues, number of gaps, and length of the proteins). (2) Alignments found by the same program. Here, we compare different alignments for the same pair of protein structures, as well as different alignments for different pairs of structures. In the second case, the geometric match measures can be used in addition to the native score provided by the particular protein structure alignment program. Clearly, native scores cannot be used when comparing alignments found by different programs as the scores may be in different units and on different scales.

4.1.1 Set of structures to be aligned

The set of structures aligned in this study consists of 2,930 sequence-diverse CATH v.2.4 domains, each with a CATH classification. As we focus on three-dimensional structures, we consider only the top three levels of CATH, Class, Architecture, and

Topology to give a CAT classification. We refer to a set of structures with the same CAT classification as a *fold class*. There are 769 fold classes in the set that fit the Class categories as follows: 218 'Mainly-α' class, 132 'Mainly-β' class, 345 'Mixed α-β' class and 74 'Few Secondary Structures' class. A list of the names of the CATH domain of all these structures is available online[1].

Using a set of structures with sufficient sequence diversity ensures that the set is duplicate-free and that the problem of structural alignment is non-trivial for all pairs. Here, we select the structures as follows: (1) Sort all 34,287 CATH v.2.4 domains by their SPACI score [8]. (2) Select the domains from the list, starting with the one with the best SPACI score (most accurately determined). (3) Remove from the list the selected one and all domains that share significant sequence similarity with it (FASTA [90] E-value $< 10^{-4}$). (4) Continue until there are no domains left for selection. This is the same method used by Brenner *et al.* [7] to produce a sequence-diverse set of SCOP structures.

4.1.2 Running the alignments

Each of the programs compared in this study aligned all 8,581,970 pairs of structures from the above set (i.e. 2,930 $\times$ 2,929 = 8,581,970 pairs). As our focus is evaluating the performance of the structural alignment methods, and due to the scope of the test, we ran the standalone Linux versions of the programs that implement the methods in-house using i386 Intel processors. Unless otherwise noted, each structural alignment program took as input the coordinates for all atoms (all lines starting with ATOM, TER or END in the pdb file). Each of the programs tested output the alignments and their cRMS values; from the alignment, we then calculated the number of matched residues and the total number of gaps (for SSM we could not derive the number of gaps). We stored the values of the geometric properties of the alignment together with the native score for further analysis. Each pair of structures has two alignments and we consider the one with the better native score (i.e. we use only 4,290,985 alignments). In cases where a particular method finds more than one alignment for a

[1] `http://csb.stanford.edu/rachel/comparison/subset_list_web`

given pair of proteins, we consider the alignment with the best native score. Lastly, we recorded the computing times.

The structural alignment methods that we evaluate are (1) SSAP [118], (2) STRUCTAL [112], (3) DALI [48], more specifically DaliLite (v.1.8) [49], (4) LSQMAN [61], (5) CE [104], and (6) SSM [69]. In CE, we change the source code so that the z-threshold value is 0 rather than 3.5, to ensure that the optimization block is always invoked. For CE, we also add the SEQRES fields to the input files to insure proper parsing of multi fragment chains. Since the ROC curve of CE based on SAS was significantly better than the one based on the native score, we select the best alignment for a pair of structures based on the one with highest SAS score. In LSQMAN, we use 'Brute_force' mode, following the O macro procedure published in align2.omac[2].

4.2 ROC curves Analysis

We first evaluate the methods (i.e. the six individual programs mentioned above, and the 'Best-of-All' method) by comparing their ROC (Receiver Operating Characteristic) curves [41]; this is the traditional method of evaluating the performance of structural alignment programs. ROC curves quantify how much a scoring scheme agrees with a gold standard; the gold standard indicates for every pair if it is similar or not. Here, we derive a binary (similar or not similar) gold standard from the CATH hierarchy: we consider a structure pair to be a *positive* (similar) if both structures have the same CAT classification, and to be a *negative* (not similar) otherwise. The alignments found by each program are sorted according to a quality measure, either one of the four geometric measures or the native score of the program. For an ordering that is based on a particular measure, we consider varying thresholds: for each threshold, all pairs below the it are assumed positive, and all above it negative. Among those pairs, the ones that agree with the standard are called *true positives (TP)* while those that do not are *false positives (FP)*. The ROC curve plots the fraction of FPs (1-specificity) against the fraction of TPs (sensitivity). These fractions are calculated at every increasing threshold of the measure, starting from the most

[2]downloaded from `ftp://xray.mc.uu.se/pub/gerard/omac/align2.omac`

significant in the pair list sorted by the measure. A perfect predictor will have a ROC curve that climbs up the y-axis turning right at the left-topmost corner, with area 1 below it; a completely random predictor will have a curve that follows the diagonal, with area 0.5 below it. A method that best agrees with the gold standard will have the uppermost curve, or equivalently, the one with the largest area under it. It can be argued that we are more concerned with comparing the agreement of the methods with CATH when the percentage of FPs is low. We do this by plotting the x-axis of the ROC curve in log scale, that is, $\log_{10}$ of fraction of FPs against the fraction of TPs.

Panel A of Figure 4.1 shows the ROC curves of all the methods; we sort the alignments either by their native score (dashed lines) or by the SAS geometric match measure (solid lines). Panel B shows the same ROC curves with $\log_{10}$ (fraction of FP) vs. fraction of TPs. Table 4.1 lists values quantifying these ROC curves: the area under the curve and the number of TPs when the fraction of FPs is 1% (numbering 42,910).

When sorting the alignments by their SAS measure, the ROC curve analysis suggests that DALI, CE, STRUCTAL, and SSM are the strongest methods. When sorting the alignments by their native scores, the methods DALI and STRUCTAL are strongest. At low false positives rates (Panel B), DALI, CE, SSAP and STRUCTAL do well. It is somewhat surprising that the programs CE, SSM, LSQMAN and SSAP do much better when using the geometric measure than when using their own native score. DALI performs similarly when using the two measures, but the geometric score does better in the lower FP rates. STRUCTAL also performs similarly using these two measures, although the STRUCTAL native score, which was specifically designed to be better than the SAS measure [35], does increase the area under the ROC curve from 0.93 to 0.94. Clearly, a program can be successful in finding good alignments and less successful in evaluating them. Another surprising result is that the 'Best-of-All' method does not perform better, in terms of its ROC curve, than the individual methods.

4.2.1 Comparing ROC curves and geometric match measures

In Panels C and D of Figure 4.1 we plot the average SAS measure of the TPs and the FPs below increasing thresholds against the fraction of TPs. This compares directly the quality of the alignments found by different methods in equally sized sets of pairs. Here, successful methods find alignments with lower SAS values, represented by curves that are shifted to the left. The average SAS of the TPs is lower than the average SAS of the FPs. As expected from a list sorted by increasing SAS values, this difference is small and subtle in the parts of the curve corresponding to a low TP fraction. The 'Best-of-All' method finds the best alignments, followed by STRUCTAL and SSM. Surprisingly, the ranking of the structural alignment methods based on their average SAS values differs from that implied by their ROC curves. For example, DALI and CE have almost identical ROC curves ranking them similarly, yet their SAS curves show that CE finds consistently better alignments. Another example is SSAP that performs well by its ROC curve, while its average SAS curves suggests otherwise. This means that the best alignments found by SSAP, although generally not as good as those found by other methods, often correspond to proteins in the same CATH fold class. Clearly, a particular method can seem as, or more, successful than another method based on its ROC curve, while the actual alignments it finds are inferior.

We have also plotted the same figures when sorting the alignments by their SI, MI and GSAS measures (data not shown but available online[3]). In general the relative performance of the methods is similar when judged by the average values of the four geometric match measures. When examining the curves of average values of SI, MI, and GSAS, we see that similarly to Figure 4.1 (Panels C and D) the implied ranking of the methods is different from that suggested by the methods' ROC curves.

[3] `http://csb.stanford.edu/rachel/comparison/`

Measure	SSAP	STRU-CTAL	DALI	LSQ-MAN	CE	SSM	Best of ALL
ROC Area (native score)	0.83	**0.94**	**0.94**	0.70	0.89	0.80	-
ROC Area (SAS score)	0.91	0.93	**0.94**	0.87	**0.94**	0.93	0.93
ROC Area (GSAS score)	0.91	**0.94**	**0.94**	0.84	**0.94**	-	0.93
ROC Area (SI score)	0.80	0.79	**0.84**	0.72	0.81	0.79	0.76
ROC Area (MI score)	0.80	0.78	**0.84**	0.72	0.79	0.82	0.78
TP at 1% FP (native score)	0.09	0.42	**0.50**	0.004	0.29	0.03	-
TP at 1% FP (SAS score)	**0.48**	0.32	0.47	0.29	0.46	0.25	0.22
TP at 1% FP (GSAS score)	**0.48**	0.30	0.47	0.20	0.46	-	0.15
TP at 1% FP (SI score)	**0.16**	0.02	**0.16**	0.04	0.09	0.008	0.01
TP at 1% FP (MI score)	**0.16**	0.03	**0.16**	0.07	0.10	0.04	0.03
% CAT -CAT at GSAS = 5 Å	19	**68**	37	29	50	-	**74.0**
% CAT -CAT at SAS = 5 Å	25	**80**	62	54	61	73	**85.7**
% CAT -CAT at SI = 5 Å	16	56	36	36	37	45	60.4
% CAT -CAT at MI=0.8	26	**62**	44	28	50	53	**65.5**
% All pairs at GSAS = 5 Å	0.6	**5.6**	1.5	2.6	2.4	-	**8.5**
% All pairs at SAS = 5 Å	0.8	**9.4**	3.5	5.6	3.5	7.1	**13.5**
% All pairs at SI = 5 Å	1.4	**17.2**	4.7	11.1	6.1	13.8	**24.2**
% All pairs at MI = 0.8	2.6	**21.7**	7.0	6.9	12.2	13.6	**25.2**

Table 4.1: Quantifiers of the ROC curves and the cumulative distributions that distinguish structural alignment methods. A high value is always better: the best scores for each measure are shown in bold, as are the scores for 'Best-of-All' when they are the best.

4.3 Direct comparison of the methods using geometric match measures

We calculate the four geometric match measures (SI, MI, SAS, and GSAS) for all alignments found by all the methods. In all four measures, better alignments correspond to lower values. Using this data, we calculate the cumulative distributions of the geometric measures, over the set of 104,309 structure pairs, that have the same CAT classification (the CAT-CAT set) and over the set of all structure pairs (the full set). In both cases, we normalize to 100% the total number of pairs in the set.

Figure 4.2 shows the cumulative distributions of the geometric match measures.

Here, better performance corresponds to finding more alignments (greater values along the y-axis) with better geometric match measures (lower values along the x-axis). In the case of GSAS, SAS, and SI, we focus on good alignments: the cutoff value is 5 Å in all three cases. Even though all programs were given the same set of pairs, the maximum value on each curve varies, since there are many alignments of match measure greater than 5 Å that are not shown. For each method, Table 4.1 lists, for both the CAT-CAT set and full set, the number of pairs for which good alignments are found (expressed as a percentage of pairs in the set).

We see that the 'Best-of-All' method finds the greatest number of good alignments, both in the CAT-CAT set and the full set, for all four geometric measures; the second best performer is STRUCTAL in both sets and for all geometric measures; the third best performer is SSM (except when using GSAS). Among the programs compared, SSAP performs the worst. When using GSAS, a version of SAS that penalizes alignment gaps, the relative performance is flipped in two cases: CE vs. DALI and CE vs. LSQMAN. This implies that CE finds less fragmented alignments than those found by DALI and LSQMAN. We also see that LSQMAN finds more highly similar alignments, and less moderately similar alignments, than all other methods except STRUCTAL.

When using the similarity cutoff value 5 Å, even the best methods find good alignments for no more than 80% of the CAT-CAT pairs. At the same threshold level, there are good alignments for about 10% of all pairs, even though the CAT-CAT pairs account for only 2.5% of these pairs. This shows that there is a significant amount of structural similarity between structures in different fold classes. The GSAS measure of an alignment will generally be larger than its SAS measure because we reduce the denominator (k) by the number of gaps; for this reason the percentage of pairs below the 5 Å threshold is smaller for GSAS than for SAS. Similarly, the relative size of the values of SI and SAS depends on how the typical length of the shorter structure relates to the value 100. We see that the percentage of pairs below the 5 Å threshold is greater for SI than SAS, implying that the typical length of the shorter structure is less than 100 (we estimate 70 residues). This means that SAS is generally larger than SI.

	Total	SSAP	STRUCTAL	DALI	LSQMAN	CE	SSM
GSAS $\leq$ 5Å	275547 (100%)	832 (0.3%)	**189871** (69%)	5868 (2.1%)	54606 (20%)	24370 (8.8%)	NA
SAS $\leq$ 5Å	539755 (100%)	498 (0.09%)	**286972** (53%)	15648 (2.9%)	103408 (19.2%)	15844 (2.9%)	117385 (21.8%)
SI $\leq$ 5 Å	978531 (100%)	3745 (0.4%)	**497330** (51%)	24767 (2.5%)	201202 (21%)	17142 (1.8%)	234345 (24%)
MI$\leq$ 0.8	880503 (100%)	4579 (0.5%)	**573542** (65%)	31402 (3.6%)	63088 (7.2%)	72974 (8.3%)	134918 (15.3%)

Table 4.2: Contributions to 'Best-of-All' Method. The absolute number of alignments contributed by each method is listed and the percentage of alignments is given in parentheses. The largest contributor is shown in bold.

Table 4.2 lists the proportional contribution of each of the methods to the 'Best-of-All' method, when considering good alignments. In all cases, STRUCTAL is the leading contributor, contributing more than 50%. SSM is the second largest contributor, with more than 15%, followed by LSQMAN with over 7%. SSAP, DALI and CE contribute less to the combined effort; if one of the top contributors were to be omitted these methods could contribute more.

4.3.1 Analysis of the good alignments found by 'Best-of-All'

For each pair of structures we find the best structural alignment, as determined by its GSAS, SAS or SI value. This set of structure pairs is partitioned into four sets: (1) pairs that agree on the three CATH classifiers: Classification/ Architecture/ Topology (the CAT set) (2) pairs that agree only on the first two classifiers (the CA set) (3) pairs that agree only on the first classifier (the C set) and (4) Others. In Figure 4.3 we plot, for several threshold values of the geometric similarity measures (less than 2.5 Å, 3 Å, . . ., 5 Å), the number of alignments found at that level of similarity (upper panel) as well as the percentage of each of the four sets found at that level (lower panel). The rightmost panels show the same analysis for the subset of long alignments (more than 50 residues matched). The threshold values we consider describe structural alignments that vary from highly similar (less than 2.5 Å) to moderately similar (less

than 5 Å).

Figure 4.3 shows that there are many cases of very similar structure pairs that are in different CAT classes; in some cases they are even in different C classes. For example, with a GSAS threshold of 5 Å, less than 40% of the pairs of proteins found are in the same CAT class, even when considering only the long alignments (more than 50 residues matched). Note, that at this level of similarity, only 80% of all the pairs of proteins with the same CAT classification are detected by any of the programs (see Figure 4.2). For both GSAS and SAS, the fraction of the alignments for which both structures are in the same CAT class is biggest for the highly similar pairs (less than 2.5 Å), and lowest for the moderately similar pairs (less than 5 Å). This expected behavior is even more pronounced when we consider only long alignments. Figure 4.3 also shows that the number of good alignments is greatly reduced (by more than a factor of 2) when restricting our attention to long alignments.

4.3.2 Additional evaluation criteria

Challenging alignments for each protein structure alignment method. Table 4.3 lists the total number of pairs of proteins that only the particular method was able to identify as structurally similar. We include only those pairs of structures for which one of the methods found a good alignment (SAS less than 4 Å and more than 35 residues matched), and all other programs found only bad alignments (SAS greater than 6 Å for any length of match). These alignments are by definition challenging for all programs but the successful one. We also present in Table 4.3 a breakdown of these cases according to the C classification of the structures aligned. The complete list of these pairs, along with the best cRMS, the alignment length and the SAS values for all the programs is available online[4]. STRUCTAL contributes the largest number of alignments, followed by LSQMAN and SSM. For all three methods, a significant number of the contributed pairs corresponds to similarities between a 'Mainly α' protein and a 'Mixed α-β' protein. In the case of STRUCTAL, there is also a significant number of pairs involving a 'Mainly β' protein and a 'Mixed α-β'

[4] `http://csb.stanford.edu/rachel/comparison/table_3_web`

Category	Total	SSAP	STRUCTAL	DALI	LSQMAN	CE	SSM
All α-All α	369	0	103	2	212	0	52
(same CAT)[a]	(8)		(5)	(0)	(2)		(1)
All α-All β	61	0	57	0	2	0	2
All α-Mixed α-β	610	0	275	0	243	0	92
All α-Few Sec. Str.	13	0	7	1	4	0	1
All β-All β	37	1	24	1	10	0	1
(same CAT)[a]	(5)	(0)	(4)	(0)	(1)		(0)
All β-Mixed α-β	318	0	260	1	39	0	18
All β-Few Sec. Str.	0	0	0	0	0	0	0
Mixed α-β-Mixed α-β	292	0	130	1	111	0	50
(same CAT)[a]	(8)		(4)	(0)	(1)		(3)
Mixed α-β-Few Sec. Str.	4	0	2	0	2	0	0
Few Sec. Str.-Few Sec. Str.	0	0	0	0	0	0	0
Total	1704	1	858	6	623	0	216

Table 4.3: Categorization of structure pairs with good alignments found only by one method, according to their class (C) classifiers. Good alignments are defined as those where one of the methods finds a good alignment (SAS < 4 Å and matches more than 35 residues), while all other methods find bad alignments (SAS > 6 Å for any length of match). We also tested other definitions of difficult alignments (e.g., longer matches, or less stringent SAS values) and found a similar distribution amongst the different methods (data not shown).
[a]In parenthesis, we list the number of difficult alignments found with structures of the same CAT classification.

Program	SSAP	STRUCTAL	DALI	LSQMAN	CE	SSM
CPU Hours	9042	1906	4515	1790	2642	633

Table 4.4: Total running time of each program on all 2,930 × 2,929 = 8,581,970 pairs. The programs run on an Intel Xeon 2.8 GHz processor with 512 Mbytes of RAM memory.

protein.

Analysis of the performance of the methods on different CATH classes. Figure 4.4 compares the relative success of each structural alignment method on CAT pairs in the four classes of protein structure (level C of CATH hierarchy). This is done for alignments with SAS values between 2.5 Å and 5 Å. We see, that α-α pairs, and α-β pairs are over-represented when the match is good (SAS value less than 3.5 Å), and, consequently, the β-β pairs are under-represented. When the match is less good (SAS value between 4 Å and 5 Å) each method behaves as expected finding different classes of pairs with a frequency that is proportional to the fraction of the pairs in the entire CAT set (shown by the horizontal lines in each panel). Most methods behave similarly in that at a particular SAS value, they detect similar percentages in each of the four classes. Such common behavior is unexpected, as the methods do not necessarily find the same pairs or even similar numbers of pairs. The LSQMAN method is unusual in that it finds more β-β pairs when the match is good (low SAS value) and consequently under-represents α-β pairs. Most of the pairs in CAT are 'Mixed α-β' pairs (71%), followed by 'Mainly β' pairs (23.5%), 'Mainly α' pairs (6.2%) and 'Few Secondary Structures' (0.45%). The fact that most methods detect relatively few β-β pairs of high match quality may mean that matches between β proteins are less good possibly due to the greater deformability of β strands.

Running times. Table 4.4 lists the total amount of central processor unit time (CPU hours) used to compute all the structural alignments for each of the programs. All programs were run under the Linux operating system (RedHat 7.3) on a cluster of dual 2.8 MHz Intel Xeon processor machines, each with 1 Gigabyte of memory. SSM is the fastest method, followed by LSQMAN; SSAP and DALI are the slowest methods.

4.4 Discussion

Reservations about ROC curves. In our opinion, the number of disadvantages of using the ROC curves methodology for comparing structural alignment methods exceeds the number of potential advantages. ROC curves seem like a reasonable way to compare structural alignment methods because each curve evaluates a method with respect to an agreed gold standard (in the case of this study, CATH [87]); moreover, self-evaluation of the methods (i.e. their native scores) can be used without needing any additional geometric match measure. The gold standard, however, is based on a classification, rather than a direct reflection of a similarity measure. For use in a ROC curve, the CATH classification must be converted to a binary similarity measure and this only allows two levels of similarity: "similar" if in the same C, A and T class, and "not similar" if not in the same C, A, and T class. Clearly, the scale of structural similarity is far richer, and this binary view introduces a great deal of "noise". For example, a pair of structures with the same C and A classifiers and another pair with completely different classifiers will both be treated as unrelated structure pairs. More generally, there is a significant amount of similarity between structures that have a different CAT classification; this phenomenon was also observed by Harrison *et al.* [44] and by Kihara and Skolnick [60]. However, the ROC curve analysis ignores, and even penalizes, methods that find these similarities.

As previously pointed out [7, 107], the creators of CATH use information from some of the structural alignment methods; this implies that some of the methods under study are influencing the evaluation procedure. In this regard, it is of particular interest that two methods seem to agree best with the CATH classification but do not produce the best geometric matches. Another interesting detail is that one of these methods, SSAP, which was developed by some of those involved in defining the CATH classification, scores very badly when using its native score but does much better when using the SAS geometric match measure. The other method that seems to have influenced the CATH classification is DALI, the most commonly used alignment program, which has been available for many years through its web server.

When comparing two methods, there are cases in which one of the methods performs better and the ROC curve analysis fails to detect it. This can happen when one of the methods consistently finds better alignments, yet both methods order the alignments similarly. Since ROC curves use only the order of the alignments, the performance of these two methods will appear similar. Indeed, when comparing the ROC curves of DALI and CE using SAS to sort the alignments, we observe this phenomenon. Another example of this is that there are methods that perform well by the ROC curve criteria (i.e. the order of their alignments is consistent with the gold standard), yet find relatively poor alignments as evidenced by their geometric match measures. Lastly, although the 'Best-of-All' method finds the best alignments by definition, its ROC curve suggests that it is a poor performer.

Structural alignment as a geometric optimization problem. We suggest treating structural alignment as an optimization problem in which alignments with the best geometric match measure are sought. This leads naturally to a definition of a combined effort, or the 'Best-of-All' method. It also suggests a methodology for comparing methods: given several different structural alignments of the same pair of structures, it is easy to evaluate which is best based on the geometric properties of these alignments. Equivalently, when a particular method aligns two structures in such a way that it lines up many residues with a small cRMS deviation, the alignment itself serves as proof of its significance; it cannot be considered as an error or a false positive. A method can only fail to find good alignments that are known to exist.

The main observation of this work is that different alignments of a pair of structures found by different programs can be compared directly and evaluated using geometric match measures. This direct comparison also applies to a set of alignments. Thus, the overall quality can be measured via cumulative distributions of the quality of alignments of sets of protein pairs. This also suggests a way to examine cases in which one method succeeds while others fail.

Restricting the evaluation to the alignments. We believe that the principal quality of a structural alignment method is finding a good alignment. Thus, an evaluation of alignment methods should focus on the quality of the alignments. Most notably, the evaluation should not depend on the scoring of the alignments that

the programs provide. Indeed, we show that most methods (apart from DALI and STRUCTAL) are better in finding alignments than in scoring them, or equivalently, do not distinguish good alignments from bad ones. This is consistent with the findings of Sierk and Pearson [107] that many alignment programs greatly overstate the significance of structural alignments. Similarly, comparing all-against-all structures in a selected set avoids the pitfall of coupling the evaluation of the alignment programs and the database of structures used by a server. Since Novotny *et al.* [83] evaluate the aligners and the databases simultaneously, it is hard to directly compare our results with theirs. Furthermore, using servers rather than the standalone versions can give misleading results in terms of the time performance of the programs, as it depends on many extraneous factors (e.g., the network and server loads and the server hardware).

Direct comparison of structural alignment methods Our analysis shows that the combined, 'Best-of-All', method finds the best alignments. Among existing methods STRUCTAL finds the best alignments. The second best method is SSM, but due to a limitation in the program's output, we could not evaluate if the alignments found by the latter have many or few gaps. In terms of speed, SSM is the fastest method, followed by LSQMAN. Clearly, the computing time for the 'Best-of-All' method is the sum of computing time of all the methods it uses. When designing a combined structural alignment method, such as 'Best-of-All', STRUCTAL, SSM and LSQMAN are important contributors, both in terms of quantity, i.e. percent of contribution, and quality, i.e. they excel in finding difficult alignments.

Sierk and Pearson [107] evaluate (among others) CE, DALI and LSQMAN; Novotny *et al.* [83] also evaluate SSM. The ranking by Novotny *et al.* is different from ours; we believe that there are two reasons for this difference: (1) our experiment is significantly larger, and (2) we focus on the structural alignment methods, while their evaluation also depends on the server's database of structures. Our results confirm the observation by Sierk and Pearson that when considering only the ROC curves and using CATH as a gold standard, DALI appears to be the best performer. We also confirm the observation of Shindyalov and Bourne [105] that CE produces alignments with fewer gaps compared to DALI.

Future directions. Understanding protein function and protein evolution are

key aims in structural biology. Both are furthered by detecting all known protein structures that are geometrically similar to a given query structure. The straightforward way of doing this is to maintain a database of all (representative) structures, and compare the query structure to each of the structures in the database, using a structural alignment method. In this study, we evaluate the structural alignment methods that can be "plugged" into this procedure. Following homology modeling [37] one can construct a method that calls different structural alignment methods, and selects the best alignment(s). The geometric match measures are reasonable criteria for this selection. Unfortunately, as the database is fairly large, this is computationally expensive. Ideally, we would like a fast filter that rules out some of the structures. Many approaches are currently tested in order to design such filters, including methods that consider geometric properties of protein backbone [96], as well as probabilistic methods based on contact maps describing protein structures [13]. Certifying that a structure cannot be structurally aligned to another structure is a hard because the certifier must prove that no alignment can be found (for reasons other than the failure of the heuristic search). Designing such filters remains an important area of research.

We touched upon the fundamental difference between classification of protein structures and measuring the similarities amongst them. In particular, we argue that converting a classification gold standard to a binary gold standard similarity measure is too crude of an approximation. It would be interesting to study similarities among structures that are classified differently and cases in which the classification is the same, yet all methods indicate that there is no similarity. This can be due to classification errors, or, more interestingly, due to protein structures that are intrinsically hard for structural alignment methods.

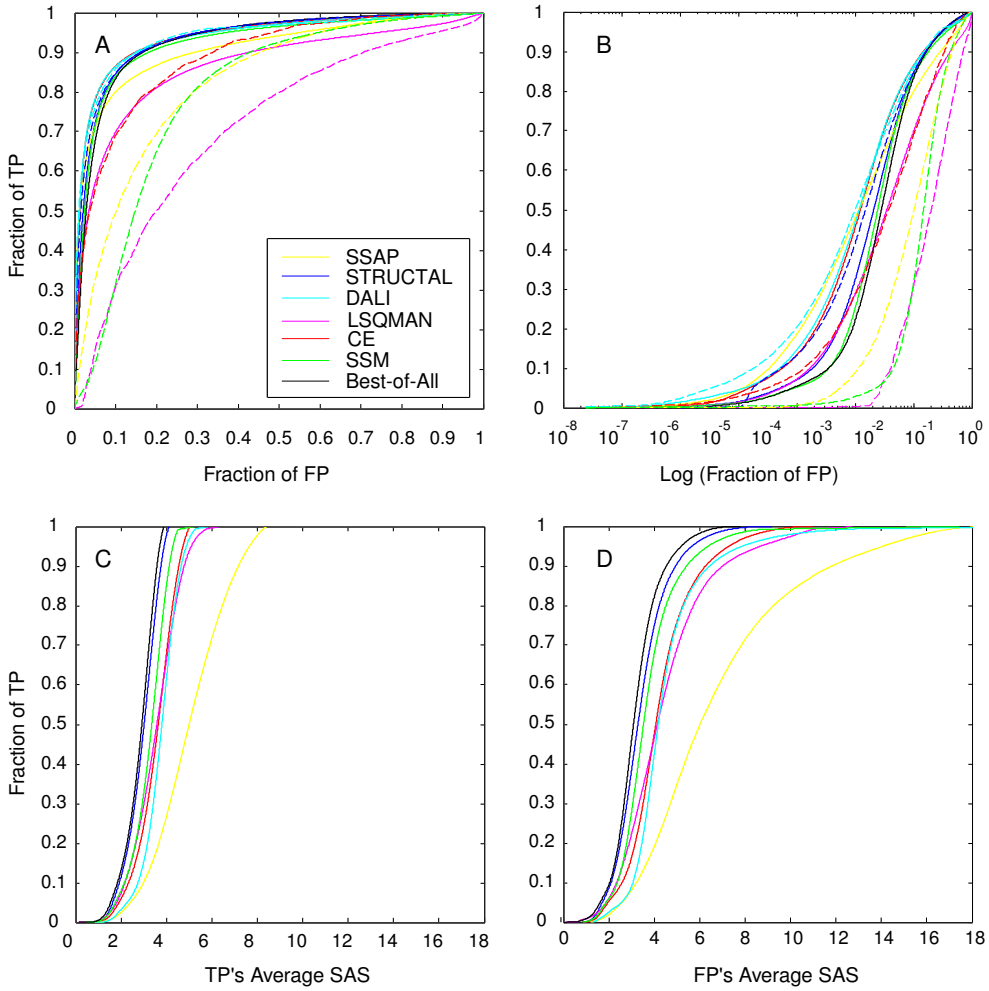

Figure 4.1: ROC curves for the structural alignment methods SSAP, STRUCTAL, DALI, LSQMAN, CE, and SSM. For varying thresholds, the fraction of true positives (TP) are the alignments below the threshold that have the same Class/ Architecture/ Topology CATH classification; the fraction of false positives (FP) are those below the threshold that have a different CAT classification. Here the alignments are sorted by their native scores, (those given by the programs and shown as dashed lines) or by the geometric match measure SAS (solid lines). In solid black, we plot the ROC curve of the 'Best-of-All' method, the best alignments (in terms of SAS) found by all methods. In Panel A, we plot the fractions of FP against the fractions of TP, and in the Panel B we plot $\log_{10}$(fraction FP) against fractions of TP, to better see performance at low rates of false positives. In Panels C and D we plot for every threshold the average SAS value of the TP and the FP alignments below that threshold. Methods that perform better in terms of their ROC curves climb to high TP values very quickly (i.e. at low FP values). We see, that the performance of the methods depends on whether the alignments are sorted by the SAS geometric match measures or the native scores. Furthermore, some of the best methods as judged by the ROC curves (such as DALI and SSAP) do not produce the best alignments as indicated by the Average TP SAS value; they seem to do well because they find even worse Average FP SAS values.

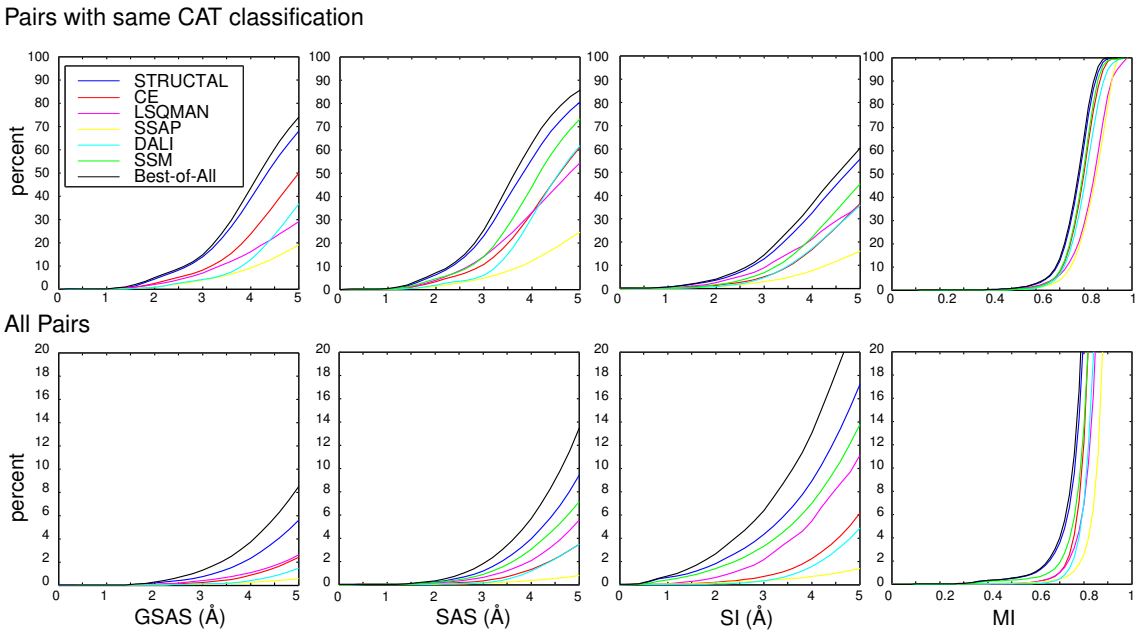

Figure 4.2: Comparison of the quality of the alignments produced by the methods SSAP, STRUCTAL, DALI, LSQMAN, CE, and SSM, using four geometric match measures: GSAS, SAS, SI, and MI. For each geometric measure and for each method, we plot a cumulative distribution. This gives the number of alignments (expressed as a percentage of the total number of alignments in the set considered) that is found with a geometric match score better than the particular threshold plotted along the x-axis. A lower value of the geometric match measure is better in all cases. In the upper panels, we consider the set of 104,309 pairs that have the same Class/Architecture/Topology (CAT) classification; in the lower panels we consider all pairs (these number 4,290,985). Better performing methods find more alignments (greater values along the y-axis) with better scores (smaller values on x-axis). The MI measure is always between 0 and 1, whereas the other measures are unbounded. For GSAS, SAS, and SI, we use a cutoff value 5 /AA, to focus on good matches. The figure also shows the cumulative distribution of the 'Best-of-All' method - a method that returns the best alignment found by any of the above methods. This method is clearly the best performer in all categories. Among the existing methods, for each of the geometric match measures, STRUCTAL is the best performer; the next best method is SSM.

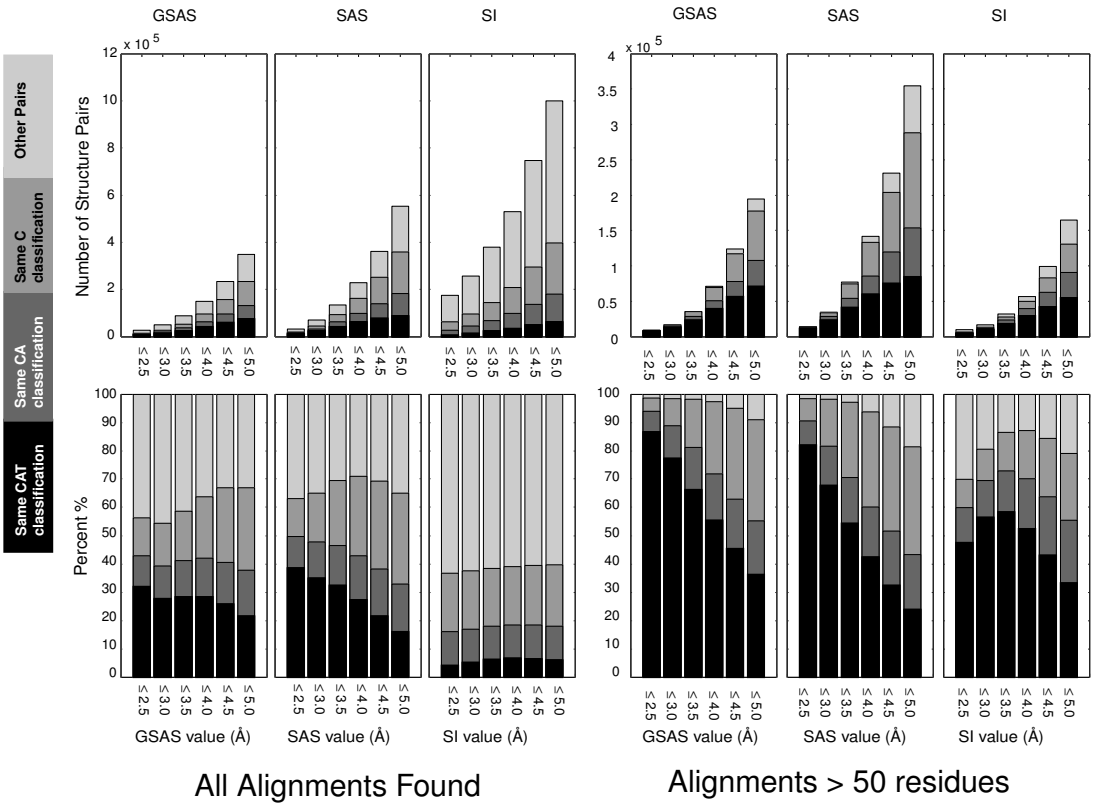

Figure 4.3: The composition of the set of structure pairs that have good alignments demonstrates the significant amount of structural similarity across CATH fold classes. These 'Best-of-All' alignments are divided into four categories depending on the similarity of the CATH classification of the two aligned structures. These four categories (color-coded from black to light gray) are: (1) both structures have the same C, A and T classifiers (CAT set), (2) both structures have only the same C and A classifiers (CA set), (3) both structures have only the same C classifier (C set), and (4) both structures have different C classifiers (Other pair set). We consider all good alignments, i.e. of low GSAS, SAS or SI value (left hand 6 panels), as well as the subset of good alignments with more than 50 matched residues (right hand 6 panels). The upper panels give the number of good alignments, and the lower panels plot the percentage of alignments found at that level of similarity for each category. All methods find many examples of highly similar structures that CATH classifies differently.

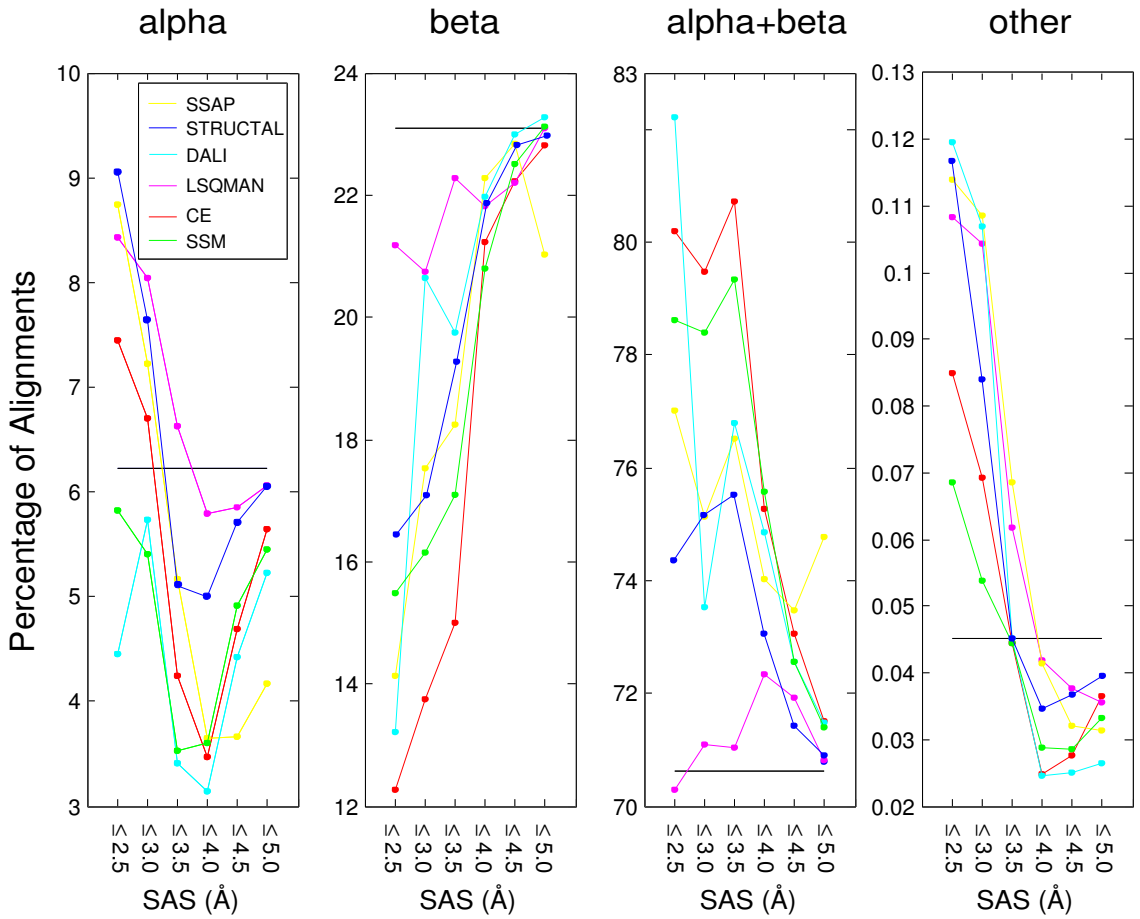

Figure 4.4: Comparing the performance of the different structural alignment programs when aligning different classes of structures. We consider all 104,309 pairs of the same fold class (i.e. same CAT), and partition them according to their C classification into four classes: 'Mainly α', 'Mainly β', 'Mixed α-β' and 'Few Secondary Structure' (from left to right). For each of the programs, and for each SAS threshold value, we plot the percent of alignments found below it. The percent of pairs of each group, among all pairs, is the horizontal line. We see, that 'Mainly α' pairs, and 'Mixed α-β' pairs, are over-represented in the high geometric similarity region, while the 'Mainly β' pairs are under-represented. Generally, all methods have similar behavior, with the exception of LSQMAN, which is less successful, compared to all the other methods at detecting 'Mixed α-β' pair similarity when the geometric similarity is high (this leads to a compensatory increase in recognition of alignments of 'Mainly α', 'Mainly β', and 'Few Secondary Structures' pairs).

Chapter 5

Efficient approximations of protein structure

We represent the polypeptide chain by a sequence of fragments that are concatenated without any degrees of freedom. The fragments are chosen from a library of fragments that are representative of the PDB and fitted to the native structure using a greedy build-up method. This gives a one-dimensional representation of native protein three-dimensional structure whose quality depends on the parameters of the library. Here we construct libraries by clustering the fragments observed in the PDB and selecting one representative from each cluster. Each library is characterized by the quality of fit (accuracy) and the number of allowed states per residue (complexity). We find that the accuracy depends on the complexity and varies from 2.9 Å for a 2.7-states per residue model based on fragments of length 7 to 0.76 Å for a 15-states per residue model based on fragments of length 5. Our goal is to find representations that are both accurate and economical (low complexity). The models defined here are substantially better in this regard: with 10 states per residue we approximate native protein structure to 1 Å compared to over 20 states per residue needed previously.

5.1 Small libraries of protein fragments

We cluster fragments of protein backbone and select one representative per cluster to form a library. The libraries[1] have fragments of different lengths f, varying from four to seven residues, and different sizes s (i.e., different number of clusters) varying from 4 to 300.

5.1.1 Fragments data sets

A set of proteins from the PDB with the most reliable structural data served as our initial data set for the clustering analysis. We use 200 unique protein domains as defined by SCOP version 1.57 [81] with the highest-ranking SPACI [8] scores (see Table 5.1). The 200 domains, all with a SPACI score greater than 0.534, have a total of 36,397 residues. Here, we approximate the chain path describing the fold of each of these proteins by the atomic coordinates of its C^{α} atoms. We extract four training sets of protein backbone fragments, of fragment lengths ranging from four to seven residues. Each of these sets is comprised of all consecutive non-overlapping fragments of the appropriate fixed length, starting at a random initial position. It is not advantageous to include overlapping fragments in these sets, as any two neighboring fragments are very close to each other solely because they have a large overlapping part. This structural overlap introduces noise into the training set and makes the clustering task significantly harder. The numbers of fragments in our data sets are 8949, 7123, 5910, and 5029 for the four, five, six, and seven residue data sets, respectively.

Two special characteristics of our fragment data sets that need to be considered before clustering are the outlier fragments and the very high concentration of α-helical fragments. Outliers are fragments that have a relatively large cRMS deviation from all other fragments; these cannot represent common structural protein motifs. Thus, we facilitate the clustering task by weeding out these outliers according to a threshold. In all cases approximately 10% of the fragments are discarded using the threshold values 0.074 Å, 0.307 Å, 0.487Å, and 0.755 Å for the data sets of fragments of length four, five, six, and seven residues, respectively. Namely, we eliminate all

[1] available online at `http://csb.stanford.edu/rachel/fragments`

dlgci__	0.78	1.33	d1rhs__	1.36	0.72	d1a4ia2	1.50	0.62	d1d3va_	1.70	0.57
d1cbn__	0.83	1.23	d1qh4a2	1.41	0.72	d1c1ka_	1.45	0.62	d7odca1	1.60	0.57
d3pyp__	0.85	1.20	d1qh4a1	1.41	0.72	d1byqa_	1.50	0.62	d1vns__	1.66	0.57
d1rb9__	0.92	1.17	d1qaua_	1.25	0.72	d1aie__	1.50	0.62	d1ctf__	1.70	0.57
d2pvba_	0.91	1.15	d3ebx__	1.40	0.71	d1dpta_	1.54	0.62	d1czfa_	1.68	0.57
d3lzt__	0.92	1.15	d2eng__	1.50	0.71	d1bx4a_	1.50	0.62	d1rzl__	1.60	0.57
d1bxoa_	0.95	1.10	d1qhva_	1.51	0.70	d1qgua_	1.60	0.62	d1tx4a_	1.65	0.57
d2fdn__	0.94	1.10	d2end__	1.45	0.70	d1c3wa_	1.55	0.62	d1ako__	1.70	0.57
d7a3ha_	0.95	1.09	d1bsma2	1.35	0.70	d1hfes_	1.60	0.61	d3cla__	1.75	0.57
d1nls__	0.94	1.07	d1bsma1	1.35	0.70	d1hfel1	1.60	0.61	d1burs_	1.80	0.56
d1b0ya_	0.93	1.07	e1pid1b	1.30	0.70	d1qgwa_	1.63	0.61	d1kid__	1.70	0.56
d1byi__	0.97	1.07	e1pid1a	1.30	0.70	d1orc__	1.54	0.61	d2cpga_	1.60	0.56
d1cex__	1.00	1.07	d3euga_	1.43	0.69	d1qsaa1	1.65	0.61	d2bbkl_	1.75	0.56
d1ixh__	0.98	1.06	d3vub__	1.40	0.68	d1dpsa_	1.60	0.61	d1qipa_	1.72	0.56
d1a6m__	1.00	1.05	d1qfma1	1.40	0.67	d1nox__	1.59	0.61	d1ttba_	1.70	0.56
d1aho__	0.96	1.04	d1bgf__	1.45	0.67	d1b3aa_	1.60	0.61	d1qhfa_	1.70	0.56
d1cxqa_	1.02	1.03	d1aba__	1.45	0.67	d1cipa1	1.50	0.61	d1dhn__	1.65	0.56
d2erl__	1.00	1.02	d1qtsa2	1.40	0.67	d1kpf__	1.50	0.60	d2ahja_	1.70	0.56
d1mfma_	1.02	1.01	d1sgpi_	1.40	0.67	d1lam_1	1.60	0.60	d3stda_	1.65	0.56
d1lkka_	1.00	1.00	d1di6a_	1.45	0.67	d1krn__	1.67	0.60	d1yvei1	1.65	0.56
d3sil__	1.05	0.99	d1utg__	1.34	0.67	d1gai__	1.70	0.60	d1pda_2	1.76	0.56
d2igd__	1.10	0.98	d7atja_	1.47	0.66	d1bfg__	1.60	0.60	d1vcc__	1.60	0.56
d1qj4a_	1.10	0.94	d1yge_2	1.40	0.66	d1d7pm_	1.50	0.60	d1pdo__	1.70	0.56
d5pti__	1.00	0.92	d1yge_1	1.40	0.66	d1moq__	1.57	0.60	d1utea_	1.55	0.55
d1rgea_	1.15	0.92	d1tc1a_	1.41	0.66	d1qq5a_	1.52	0.60	d1ush_1	1.73	0.55
d1bkra_	1.10	0.92	d1mla_1	1.50	0.66	d1ubpc1	1.65	0.60	d1cjca2	1.70	0.55
d1nkd__	1.07	0.92	d1poa__	1.50	0.66	d1ubpa_	1.65	0.60	d1b2pa_	1.70	0.55
d1swua_	1.14	0.91	d2cba__	1.54	0.65	d3cyr__	1.60	0.59	d3btoa1	1.66	0.55
d1a7s__	1.12	0.88	d3pte__	1.60	0.65	d2ilk__	1.60	0.59	d1kpta_	1.75	0.55
d1mun__	1.20	0.88	d1pina_	1.35	0.65	d1ppn__	1.60	0.59	d1gsoa3	1.60	0.55
d1vfya_	1.15	0.85	d1g3p_1	1.46	0.65	d1kapp1	1.64	0.59	d1gsoa2	1.60	0.55
d1jhga_	1.30	0.83	d1cyo__	1.50	0.64	d2cpl__	1.63	0.59	d1gsoa1	1.60	0.55
d1d4oa_	1.21	0.82	d1qrea_	1.46	0.64	d1b6a_2	1.60	0.59	d1dmr_2	1.82	0.55
d1qu9a_	1.20	0.82	d1dgfa_	1.50	0.64	d1b6a_1	1.60	0.59	d1atza_	1.80	0.54
d3chbd_	1.25	0.82	d1whi__	1.50	0.64	d3grs_3	1.54	0.59	d1fnd_2	1.70	0.54
d1mroa1	1.16	0.81	d1ezm_2	1.50	0.64	d1qqqa_	1.50	0.59	d1fnd_1	1.70	0.54
d1ifc__	1.19	0.81	d1ezm_1	1.50	0.64	d1mrj__	1.60	0.59	d1a44__	1.84	0.54
d7rsa__	1.26	0.80	d1dcia_	1.50	0.64	d1aop_3	1.60	0.59	d1vhh__	1.70	0.54
d1dcs__	1.30	0.79	d1b67a_	1.48	0.64	d1php__	1.65	0.59	d1aoha_	1.70	0.54
d2pth__	1.20	0.79	d1qh5a_	1.45	0.63	d1csh__	1.60	0.58	d1doza_	1.80	0.54
d1amm_1	1.20	0.78	d1b4va2	1.50	0.63	d1t1da_	1.51	0.58	d1db1a_	1.80	0.54
d2lisa_	1.35	0.78	d1b4va1	1.50	0.63	d1ajsa_	1.60	0.58	d1thw__	1.75	0.54
d1cy5a_	1.30	0.77	d8abp__	1.49	0.63	d1qs1a1	1.50	0.58	d1tfe__	1.70	0.54
d1aac__	1.31	0.77	d1ah7__	1.50	0.63	d1a1ia1	1.60	0.58	d1svy__	1.75	0.54
d1qdda_	1.30	0.76	d1ptf__	1.60	0.63	d1smd_1	1.60	0.58	d1mjha_	1.70	0.54
d1dg6a_	1.30	0.76	d1rie__	1.50	0.63	d1b8za_	1.60	0.58	d1bm8__	1.71	0.54
d1msi__	1.25	0.75	d3ezma_	1.50	0.63	d1ay7b_	1.70	0.58	d2gsta1	1.80	0.53
d1qksa2	1.28	0.75	d1bfd_2	1.60	0.63	d1fmk_3	1.50	0.58	d1pcfa_	1.74	0.53
d256ba_	1.40	0.73	d1ra9__	1.55	0.62	d1phc__	1.60	0.58	d1mtyg_	1.70	0.53
d1bi5a1	1.56	0.72	d1dfma_	1.50	0.62	d1qgxa_	1.60	0.57	d1iiba_	1.80	0.53

Table 5.1: PDB identifiers of the training set proteins, the structure resolution and SPACI [8] score. The training set is the 200 polypeptides with the highest SPACI scores from SCOP v.1.57 [81].

1.1bmv	A.2hhb	2sod	1fdx	B.1grc	4tln	1rei	I.1tec	A.1rbb
2fd2	1bp2	A.2taa	3adk	1lh4	B.2pka	5tnc	2aat	P.2tmv
A.2ccy	2pcy	1cse	A.4mdh	3icd	1pcy	C.2dfr	A.1csc	2cna
1.1rmu	A.2hla	2ssi	1fx1	B.2at1	5cpa	1rhd	I.4sgb	A.1tnf
2gbp	1cc5	A.2tbv	3b5c	1lz1	B.2sod	6acn	2act	R.1wrp
A.2dfr	2pgk	1cts	A.5xia	3pgm	1phh	E.4er4	A.1fc1	2cpp
1.2mev	A.2hmg	2stv	1hip	B.2dfr	5cpv	1rnt	L.1f19	A.2.4ts1
2gn5	1cd4	A.2utg	3blm	1mba	B.3hmg	8adh	2alp	2cyp
A.2dhf	2rsp	1cy3	A.7cat	4ait	1pp2	E.4tmn	A.1grc	A.256b
1abp	A.2pab	3.2mev	1hoe	B.2hhb	5ebx	1sgt	L.1prc	
2i1b	1cla	A.2ypi	3ca2	1mbd	B.8api	8cat	2aza	
A.2gap	2sga	1eca	B.1coh	4ape	1ppt	H.1prc	A.1p09	
1acx	A.2pka	3.2r06	1l12	B.2hla	5ldh	1tim	M.1prc	
2liv	1cms	A.4dfr	3fxc	1ovo	C.1fc2	9pap	2cab	
A.2gls	2sns	1est	B.1csc	4sbv	1pyp	I.1cho	A.1pfk	
1bds	A.2rsp	351c	1lh1	B.2kai	5mbn	2.1bmv	O.2gd1	
2lzm	1crn	A.4hvp	3gpd	1paz	C.1prc	A.1.4ts1	2cdv	

Table 5.2: PDB identifiers of the Park and Levitt [89] test set

fragments whose cRMS deviation from the closest other fragment in the data set is greater than the threshold value. Another unique characteristic of our training set is a highly populated region of fragments from α-helices, which complicate the clustering procedure.

5.1.2 Clustering the fragments

A distance measure in structure space. We use cRMS (Equation 2.2) to measure structural deviation between two fragments. For three equal fragments of equal length A, B, and C, cRMS satisfies the triangle inequality:

$$\text{cRMS}(A, C) \leq \text{cRMS}(A, B) + \text{cRMS}(B, C)$$

and thus a reasonable choice for use in clustering [25]. We considered using other measures such as:

1. the root mean square value of the f residue (ϕ, ψ) torsion angles, where f is the fragment's length
2. the root mean square deviation of the $f - 3$ chain α angles (the torsion angle defined by 4 consecutive C^{α} atoms [72])
3. dRMS deviation (Equation 2.1).

None of these alternatives were satisfactory as the (ϕ, ψ) torsion angles were too noisy and reflective of local change, the a angles were too coarse a description of the fragment shape and the dRMS deviation does not discriminate a right-handed from left-handed structure.

Simulated annealing k-means. We cluster each of the different length fragment data sets using k-means simulated annealing, a clustering heuristic that uses simulated annealing to find better cluster centroids in k-means clustering. K-means simulated annealing repeatedly runs k-means clustering and then merges two clusters and splits another, in a Monte Carlo fashion. The clusters to be merged are selected at random, with nearby clusters more likely to be chosen; the cluster to be split is also selected at random, with disperse clusters more likely to be selected. We tried a number of different scoring functions and the one that performed best was total variance of the clustering (the sum over all clusters of the square of the distance of any fragment to its cluster centroid). The desired number of clusters is given to the clustering procedure as input and the improvement step described maintains the number of clusters. This scheme surpasses normal k-means by its improved handling of the wide range of fragment concentrations (there are many more α-helical fragments), and by its insensitivity to the initial choice of cluster centers. It even works a little better than the much more time-consuming hierarchical clustering method that merges clusters based on the maximum distance between any members of the different clusters. In appendix A we detail the simulated annealing k-means clustering heuristic and compare it to other clustering methods.

5.1.3 Complexity of a library

The complexity of a library measures the size of the space of structures that can be built using the library, normalized so that it is independent of the lengths of the approximated proteins. Intuitively, it is the "density" of the approximation net, where a more complex library generates a denser net. Let L be a library of s fragments, each f residues long. The *complexity* of L is defined as $s^{1/(f-3)}$, and is a property of the library that varies with its size and fragment length. For the detailed calculation

of the size of the space see Section 5.3.1.

5.2 Evaluating the fragment libraries

We consider a library "good" if it approximates real protein structure accurately. The clustered fragments create a library that represents well all fragments in the training set. We aspire to a set of fragments that also represents well all protein motifs (of this length) found in real proteins. Thus, we fit the library fragments to a test set of protein structures that is independent of the training set to evaluate the quality of the library.

We use the test set suggested by Park and Levitt [89]. It has 145 proteins of different structural motifs varying in length from 36 to 753 residues. For completeness, Table 5.2 lists their PDB identifiers. As with the training set, we simplify the chain paths of the test set folds by the atomic coordinates of their C^{α} atoms. In addition to its completeness, using the Park and Levitt facilitates the comparison with their results.

5.2.1 Local fit

Here, we measure the fit of the library fragments to the local conformation of the test set proteins; we denote this measure by 'local fit'. For this purpose, we break each protein structure to *all* its overlapping fragments of the specified length, f. The best local fit approximate structure for a protein is constructed, in linear time, by finding for each of the protein fragments the most similar fragment in the library (in terms to cRMS). Averaging this cRMS value over all the fragments in all the proteins in the test set, gives the local fit score for the particular library.

Table 5.3 summarizes the accuracy of the best local fit approximations for all libraries studied. Figure 5.1 plots this data as a function of the complexity. We also calculate the average cRMS deviation of the best local fit approximations of the test set proteins using the five and six residue fragment libraries published by Micheletti *et al.* [78]. Figure 5.1 shows that the fragments of the proteins in the test set can be

described very well by any of the libraries considered: the average cRMS deviation is below 1 Å in all cases. For libraries of a fixed fragment length, the accuracy of the local fit approximations is improved when the complexity (or the library size) is increased. This makes intuitive sense: libraries with a greater variety fit the fragments of the test set proteins better. For a library of the same complexity, the accuracy of the local fit approximations is improved with shorter fragments. This also makes sense: shorter fragments give a better local fit as there are fewer C^{α} atoms involved in each fragment-to-fragment comparison. Stated differently, there are six additional degrees of freedom for the rigid body rotation and translation of each fragment. With shorter fragments, there are therefore, more degrees of freedom to locally approximate a given length protein structure.

Dependency on the polypeptide length. We also consider the dependency of the fit on the length of the approximated protein. Figure 5.2(a) plots the cRMS deviation of the local fit approximations of all the proteins in the test set versus the lengths of the proteins, for one representative library of 20 fragments, five residues each. We see that the accuracy does not vary with the chain length. Similar behavior was observed in other libraries (data not shown).

5.3 Approximating proteins using libraries of fragments

One possible, yet unsatisfactory, way to construct contiguous three-dimensional structures from the library fragments is to concatenate the best local fit library fragments found in Section 5.2.1. If the first C^{α} atom of each fragment is superimposed on the last C^{α} atom of the preceding fragment, one would need to specify the relative orientation of the two fragments. This could be done using two polar coordinates but is, in any case, unsatisfactory as the reconstruction would not be discrete in that one would need to specify the list of fragments as well as the values of the continuous polar angles. More succinctly, it does not suffice because it does not offer a one-to-one correspondence between the string of library fragment codes and the global

three-dimensional structure.

5.3.1 Global fit

Instead, we use a scheme denoted 'global fit' for constructing chains from protein fragments: when adding a fragment, we position it by best superimposing its first three C^{α} atoms onto the last three C^{α} atoms of the already constructed prefix of the chain. Even if the two C^{α} triplets do not match perfectly, this will uniquely define the relative orientation of the fragments provided that the atoms of each C^{α} triplet do not lie along a line. In polypeptide chains, the distance between consecutive C^{α} atoms is fixed at 3.8 Å and the angle formed by three consecutive C^{α} atoms is between 90° and 130° [72].

Figure 5.3 illustrates a two-dimensional analog of this scheme for constructing two structures from fragments of a four-element library. Notice that in two dimensions, any two (rather than three) consecutive amino acids can be superimposed on any two consecutive amino acids in another fragment. We emphasize that the library fragments are used as mere templates — any fragment can be used repeatedly along the constructed chain. In this global fit scheme, a structure is completely described by the list of library fragments that construct it. There is a one-to-one correspondence between the space of all approximations and the space of all strings of library labels. Therefore, the space of all approximations constructed using a library is discrete, and when the length of the target structure is fixed it is also finite.

The number of structures that can be constructed using a library L depends on the number of fragments s and their length f. When constructing a chain from fragments, each library fragment extends the chain by $f-3$ residues, because 3 overlap it with the previously constructed part of the chain. The number of fragments n_f needed to build an n residues chain is the first integer larger than or equal to $n/(f-3)$. Thus, a string of n_f fragment labels from the library L fully defines an approximating structure of n residues. The size of the approximation space is equal to the number N of such strings where $N = s^{n_f} = s^{n/(f-3)} = (s^{1/(f-3)})^n$. For the purpose of defining the complexity of the library, we normalize this size, so that it is independent of the

lengths of the approximated chains. Thus, the complexity is the average number of states per residue, or equivalently the n-th root of N: $s^{1/(f-3)}$. This definition of complexity follows the convention set by Park and Levitt [89], and offers an extension to their results regarding the complexity and accuracy of discrete approximations of protein structures.

Finding good global fit approximations. While the best local fit approximation is easily found by finding the library fragment that best fits each local fragment, the sequence library fragments needed for the global fit is much harder to find. The optimal sequence of library fragments must define the three-dimensional structure with the minimal cRMS deviation from the real structure of the target protein. The number of possible sequences of fragments is, unfortunately, exponential in the length of the protein. Thus, it is impossible to consider all sequences in search for the best global fit approximation. Therefore, we follow Park and Levitt [89] and use a greedy algorithm for finding a good rather than the best global fit approximation. Starting at the N-terminus, we construct approximations for increasingly larger segments of the protein. Given a partial approximation, we extend it using the best library fragment, i.e., the one whose concatenation yields a structure of minimal cRMS deviation from the corresponding segment in the protein. Recall, concatenation is achieved by superimposing the first three residues of the added fragment on the last three residues of the already constructed segment so that $f - 3$ residues are added each time. We repeat this process until the C-terminus of the target protein is reached. This process is deterministic and takes linear time.

An important property of this model-building method is that each step is local, while the evaluation criterion of its goodness of is global. A fragment used in the construction of the approximating structure influences the overall accuracy of the approximation not only via the accuracy of the local protein segment it describes, but also through the positioning it determines for its following fragments. Consequently, it may be beneficial to be less greedy: a less well-fitting library fragment may allow better positioning of subsequent fragments, improving the overall quality of the approximation.

We improve our algorithm by keeping a set of candidate structures for extension,

rather than a single one as described above. Specifically, we make the algorithm slightly less greedy and keep a set (or heap) of the best N_{keep} greedily constructed approximations for the segment of the protein approximated thus far. At each step, we extend each of the N_{keep} approximations with all possible library fragments, and then greedily keep the best N_{keep} approximations. This algorithm is still greedy, and therefore does not guarantee the globally optimal solution, yet it explores a slightly bigger part of the approximations space. Greedy algorithms like this were first used in computational biology by Vasquez and Scheraga [123, 124] to build-up low energy conformations of polypeptide chains.

The search procedure of the global fit approximations uses a heap storing the best approximations found so far; the heap size should be selected to balance between the desire to explore a greater (polynomial) portion of approximation space and the reality of run time and memory constraints. The number of possible global fit approximations to a target protein is exponential and therefore it is impossible to explore them all, instead only a heap of N_{keep} best approximations is maintained. The running time of the procedure is linear $O(N_{\text{keep}}n|L|)$, and maintaining the heap requires $O(N_{\text{keep}})$ memory.

Figure 5.4 plots the average accuracy of the best global fit approximations found for the proteins in the test set, versus the heap size used in the construction procedure, for one representative library of 20 fragments, five residues each. As expected, better approximations are found when using a larger heap. However, the accuracy improves dramatically with increasing heap size for small heaps and remains relatively constant for larger values. Therefore, in this study we used a heap size of 4000 when searching for global fit approximations, which is an appropriate balance between the quest for accuracy and the limitations on running time. Similar behavior was observed in all libraries we considered.

Global-fit approximations. Table 5.3 also summarizes the accuracy of global fit approximations constructed from the libraries considered in this study. Figure 5.5 plots this data as a function of the library complexity. The average cRMS deviation of the global fit approximations over the test set varies from 2.58 Å for the lowest complexity library to 0.76 Å for the highest complexity library. The insert in Figure 5.5

plots this data in a log scale along with linear regression lines. We also compare our results to those of: (1) Park and Levitt [89], where the test set and the complexity measure are the same, so their results are just quoted here; and (2) Micheletti *et al.* [78]. In the latter case we use the libraries of 5 and 6 residues published online [77], construct global fit approximations for the test set and calculate the average cRMS deviation between the test set and its approximations.

Figure 5.5 offers insight to the relationship between libraries of fixed fragment length and varying complexity, as well as the relationship between libraries of fixed complexity and varying fragment length. For a fixed fragment length, more complex libraries offer better global fit approximations. This observation is expected: the complexity of libraries of fixed length depends on the number of fragments in the library and libraries with greater variety will result in more accurate approximations. More surprisingly, for a fixed complexity, libraries of longer length fragments give better global fit approximations. All the global fit cRMS data from our libraries of different complexity, c, and fragment length, f, can be well fit by a single function:

$$\text{global fit cRMS} = e^{(0.094f+1.373)}(\text{Complexity})^{-0.1039f-0.280},$$

or more simply

$$(\text{global fit cRMS}) \propto (\text{Complexity})^{(-0.1039f-0.280)}$$

and

$$(\text{Complexity}) \propto (\text{global fit cRMS})^{(0.1039f+0.280)}.$$

Park and Levitt [89] found that for model that used non-optimized (ϕ, ψ) torsion angle states

$$(\text{global fit cRMS}) \propto (\text{Complexity})^{-0.5}.$$

For a fragment length of 4, which is most like the (ϕ, ψ) states, the corresponding dependence is

$$(\text{global fit cRMS}) \propto (\text{Complexity})^{-0.7}.$$

For longer fragments, the power become more negative so that for a length of 7, the dependence is

$$(\text{global fit cRMS}) \propto (\text{Complexity})^{-1.0}.$$

This more rapid fall-off of global fit cRMS with Complexity for longer fragments means that the models based on longer fragments can model proteins better for a given complexity.

Figure 5.6 plots the average cRMS deviation of the local fit approximations from the proteins in the test set, versus the same measure of global fit approximations, for all libraries considered in this study. For any particular library, the local fit cRMS is always smaller than the corresponding global fit cRMS. This is to be expected as the local fit completely ignores the connection between adjacent fragments along the chain. These results show that local fit approximations can be used to predict the accuracy of the global fit approximations: libraries that provide accurate local fit approximations will also provide accurate global fit approximations. It is clear that for the same level of global fit cRMS deviation, the local cRMS deviation decreases sharply with fragment length.

Dependency on the polypeptide length. We also consider the dependency of the accuracy of the approximations on the length of the approximated protein. Figure 5.2(b) plots the cRMS deviation of the global fit approximations of all the proteins in the test set versus the lengths of the proteins, for one representative library of 20 fragments, five residues each. We see that the accuracy of the global fit approximations is only very slightly dependant on the chain length. Similar behavior was observed in other libraries.

Finally, to demonstrate the range in quality of fit, Figure 5.7 shows three approximations of 1tim to various accuracies.

5.4 Discussion

Independence of test set. The training set used to compile the libraries and the test set used to evaluate them are independent of one another. The training set

Fragment length	Library Size	Complexity (States/ residue)	Average local fit cRMS (Å)	Average global fit cRMS (Å)
4	4	4.00	0.39	2.23
4	6	6.00	0.35	1.64
4	7	7.00	0.33	1.48
4	8	8.00	0.32	1.39
4	10	10.00	0.30	1.12
4	12	12.00	0.28	1.01
4	14	14.00	0.26	0.92
5	10	3.16	0.57	2.57
5	20	4.47	0.47	1.85
5	30	5.48	0.43	1.59
5	40	6.32	0.40	1.41
5	50	7.07	0.39	1.28
5	60	7.75	0.37	1.20
5	80	8.94	0.35	1.06
5	100	10.00	0.34	0.99
5	150	12.25	0.31	0.86
5	225	15.00	0.29	0.76
6	40	3.42	0.65	2.30
6	60	3.91	0.59	2.02
6	70	4.12	0.58	1.92
6	80	4.31	0.56	1.87
6	100	4.64	0.54	1.72
6	200	5.85	0.48	1.41
6	300	6.69	0.45	1.26
7	50	2.66	0.85	2.89
7	100	3.16	0.76	2.41
7	150	3.50	0.72	2.16
7	200	3.76	0.68	2.04
7	250	3.98	0.66	1.91
Micheletti *et al.* fragment libraries [78]				
5	40	6.32	0.48	1.64
6	100	4.64	0.57	1.88

Table 5.3: Average accuracy of global and local cRMS deviations. The complexity is the average number of states per residue: for a library L of s fragments, each f residues long, the complexity is $s^{1/(f-3)}$.

for our procedures is a collection of fragments extracted from proteins with accurate structural data (based of their SPACI [8] scores), while the test set is an accepted set for testing questions of this type. Although lack of overlap was not a criterion used to select the training set, there is only one protein (256b) that is in both sets. This independence of these two sets assures that the results presented do not follow from learning a specific set of proteins, but rather from learning properties of protein structure.

Local-fit approximations. Local fit approximations are interesting even though the resulting structures consist of disjoint fragments that can only be found by fitting a known protein backbone. In building these local approximations we seek, as studies before us [78, 121], a short list of fragments that is representative of all fragments of known proteins. Local fit approximations capture this notion of similarity, and offer an efficient, linear time method of evaluating libraries of fragments. Comparison between the accuracy of local fit approximations using libraries found in this study and those previously constructed by Micheletti *et al.* [78] indicates that the elaborate clustering scheme used here leads to better results. In addition, local fit approximations serve as predictors to the accuracy of the computationally more expensive global fit approximations.

Building better approximation models. Park and Levitt [89] and Rooman *et al.* [97] showed that discrete approximation models for protein structure that take the uneven distribution of residue conformations in real proteins into account are more accurate than models of comparable complexity that do not. The discrete models they constructed treat all residues along the chain equally, in the sense that each residue can have any one of c conformations (where c is the complexity of the model). These conformations are described by the pair of angles (ϕ, ψ) that defines the positioning of the residue with respect to the previous residue along the chain.

The discrete approximations we construct with libraries of four residue fragments are equivalent to the optimized models considered in earlier studies [89, 97]. Indeed, the complexity of the library L_4^s of size s is $s^{1/(4-3)} = s$, or simply the number of its elements. In effect, the library fragments encode the (ϕ, ψ) angle pairs of their last residue with respect to the previous parts of the approximation, while the first three

residues position the fragment. The results we achieve with libraries of fragments of four residues are similar to those obtained previously by others. Park and Levitt's best four state model has an accuracy of 2.22 Å when approximating the test set of proteins, while the best four state model we found achieves an accuracy of 2.23 Å. Rooman *et al.* found a six state model with an average accuracy of 1.74 Å over the proteins test set, compared with an average accuracy of 1.64 Å in our six state model.

The conformations of consecutive residues along the backbone of proteins are correlated to one another. Under the reasonable assumption that the libraries we find of a specific size and fragment length are optimal, our results show this correlation between conformations of neighboring residues. For illustration purposes consider the correlation of conformations of pairs of neighboring residues, which is reflected in the relative accuracy of models from libraries of four and five residue fragments, respectively. Imagine that the conformation of two consecutive residues is independent and without any correlation. If L_4^s is an optimal s-element library of four-residue fragments, it can be used to construct equivalent library $L_5^{s'}$ of five residue fragments (concatenate all pairs of four residues with a three residue overlap to give a new library containing $s' = s^2$ five residue fragments). Clearly, L_4^s and $L_5^{s'}$ span exactly the same space of approximating structures and have the same complexity, s. If the conformation of two consecutive residues along the chain was independent of one another, and L_4^s is optimal, then $L_5^{s'}$ is optimal too. This would mean that global fit models would have the same accuracy for libraries of four and five residue fragments. Here we find that for the same complexity, models with five-residue fragments are significantly more accurate than those with four residue fragments (In Table 5.3 for a complexity of 10, the five-residue fragments fit to 0.99 Å whereas four-residue fragments fit to 1.12 Å) indicating very significant correlations.

Our use of fragment libraries in construction of approximation nets of protein structure space exploits the correlations of conformations along the backbone to achieve better low complexity models. This effect is particularly important if one wants to reproduce native structures to better than 1 Å. Here we can achieve such accuracy with a complexity of 12 for a four-residue fragment or 10 for a five-residue fragment. By comparison, the Park and Levitt model would require a complexity of

over 50 states-per residue to achieve an accuracy better than 1 Å. As the number of possible conformation for a chain of length n residues, depends on (Complexity)^n, these differences have a huge impact on the size of the particular protein's conformation space. We expect to be able to get even better results with libraries of six or seven-residue fragments. Unfortunately, very large data set of refined protein coordinates are needed to make reliable libraries for the longer fragments. Here, we have a 250 fragment library of length 7 that has a complexity of 4 and attains a global fit cRMS of 1.91 Å. To obtain a global fit cRMS value better than 1 Å, would require a complexity of about 8 and a library of $8^4 = 4096$ fragments. With the rapid pace of protein structure determination, we believe that such a library may soon be possible.

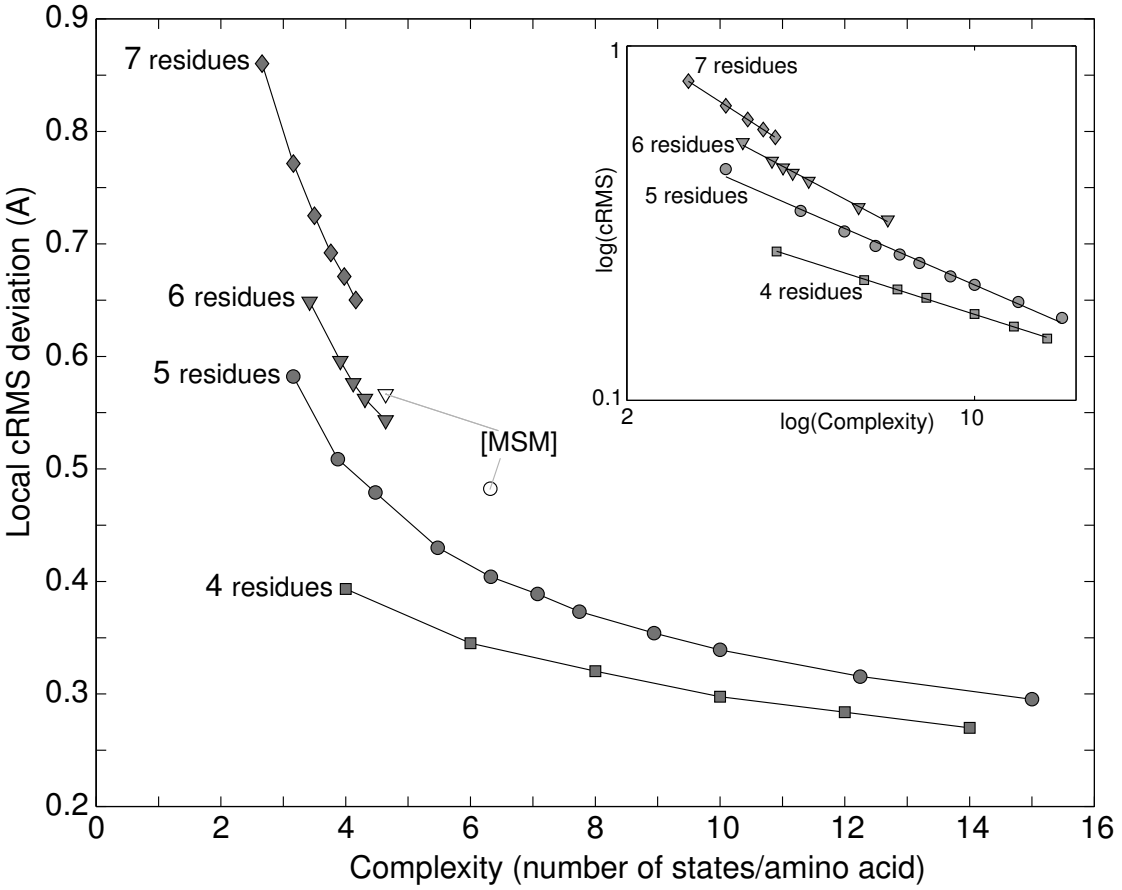

Figure 5.1: The average local cRMS deviation of test set proteins constructed using various libraries is plotted against the complexity of the library. The libraries vary by size and are of fragments of lengths 4 (squares), 5 (circles), 6 (triangles) and 7 (diamonds), respectively. The complexity is determined by the library size and the fragment length as $s^{1/(f-3)}$. For fixed fragment length, f, more complex libraries with more members, s, give more accurate approximations. The insert shows the same data in log scale: the linear fit of this data is $y = -0.313x - 0.450, y = -0.427x - 0.103, y = -0.518x + 0.186$ and $y = -0.633x + 0.459$ for fragment lengths of $4, 5, 6,$ and 7, respectively. More generally, local fit cRMS depends on library Complexity and fragment length, f, as: log(local fit cRMS) $= A \times$ log(Complexity) $+B$, where $A = -0.1051(f - 1)$ and B $= 0.3016f - 1.6358$

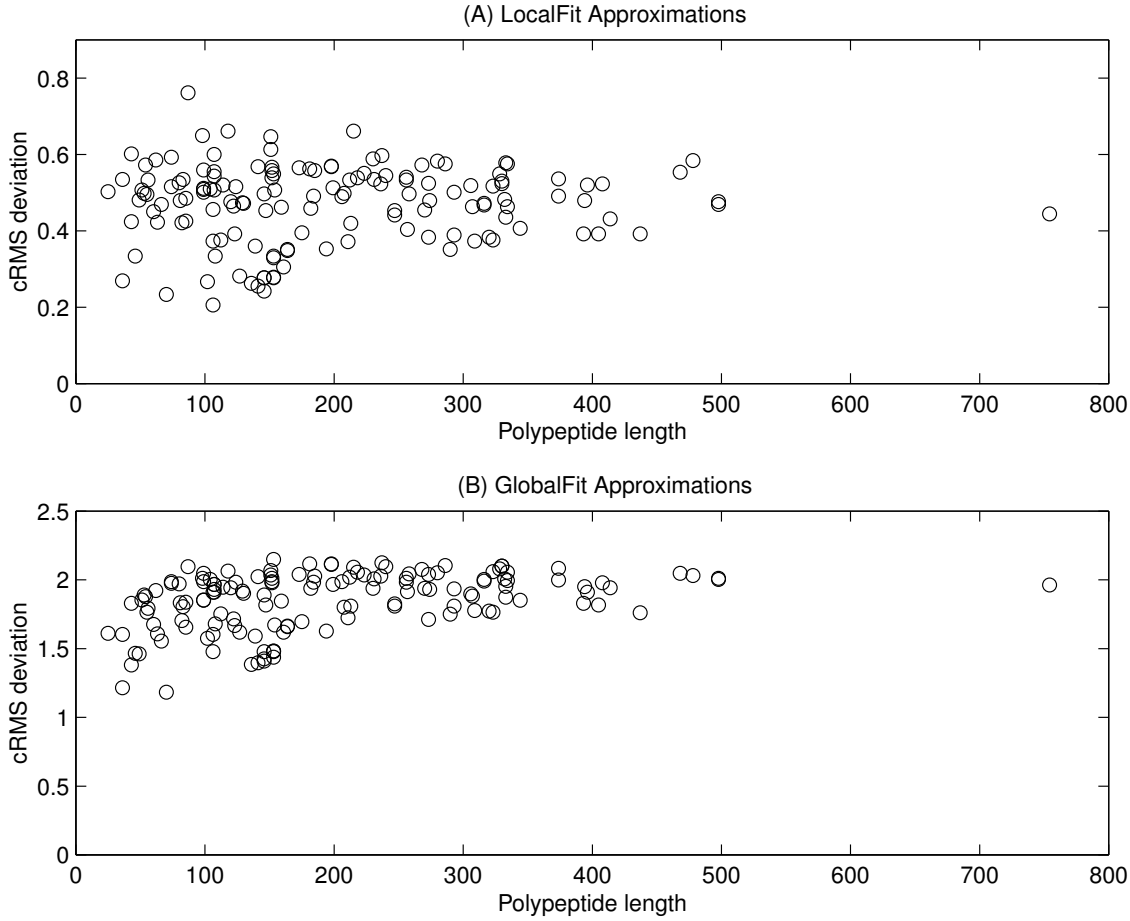

Figure 5.2: Accuracy of global fit and local fit approximations as a function of the chain length. We see, that the local fit accuracy is independent of the chain length, while the global fit accuracy decreases slightly when approximating long chains.

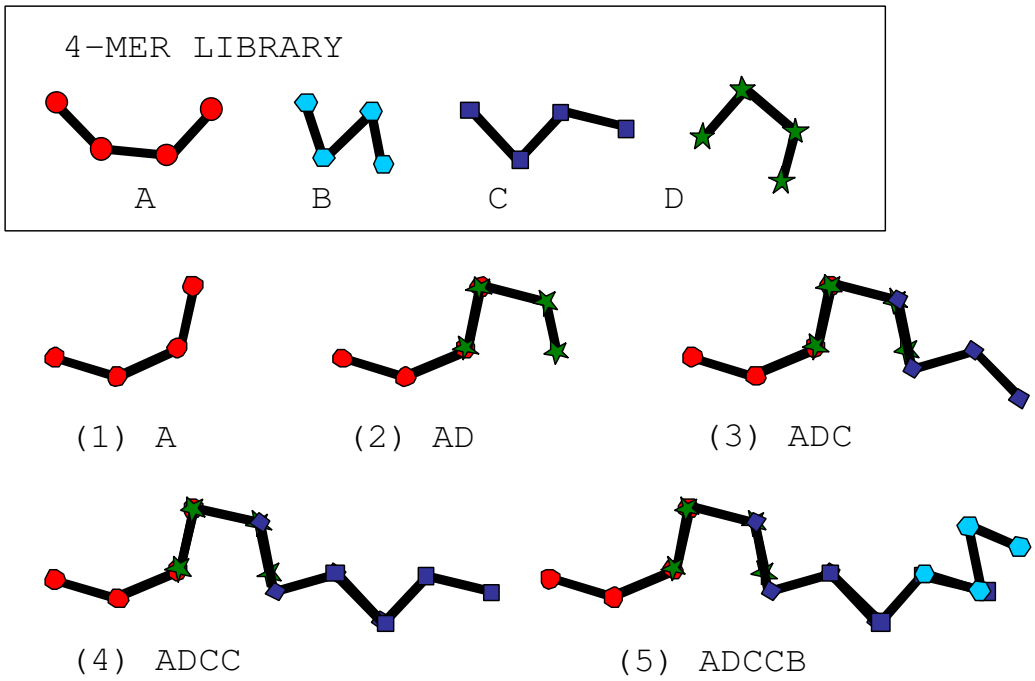

Figure 5.3: A simple two-dimensional example showing the construction of a chain formed by superimposed fragments. The 4-state library contains 4 fragments, each 4 residues long and is enclosed in the box. The fragments have library labels of A, B, C, and D. The first two residues of any fragments can be superimposed on the last two residues of the preceding fragment thus determining the position of the fragment. The construction process of the chain ADCCB is shown.

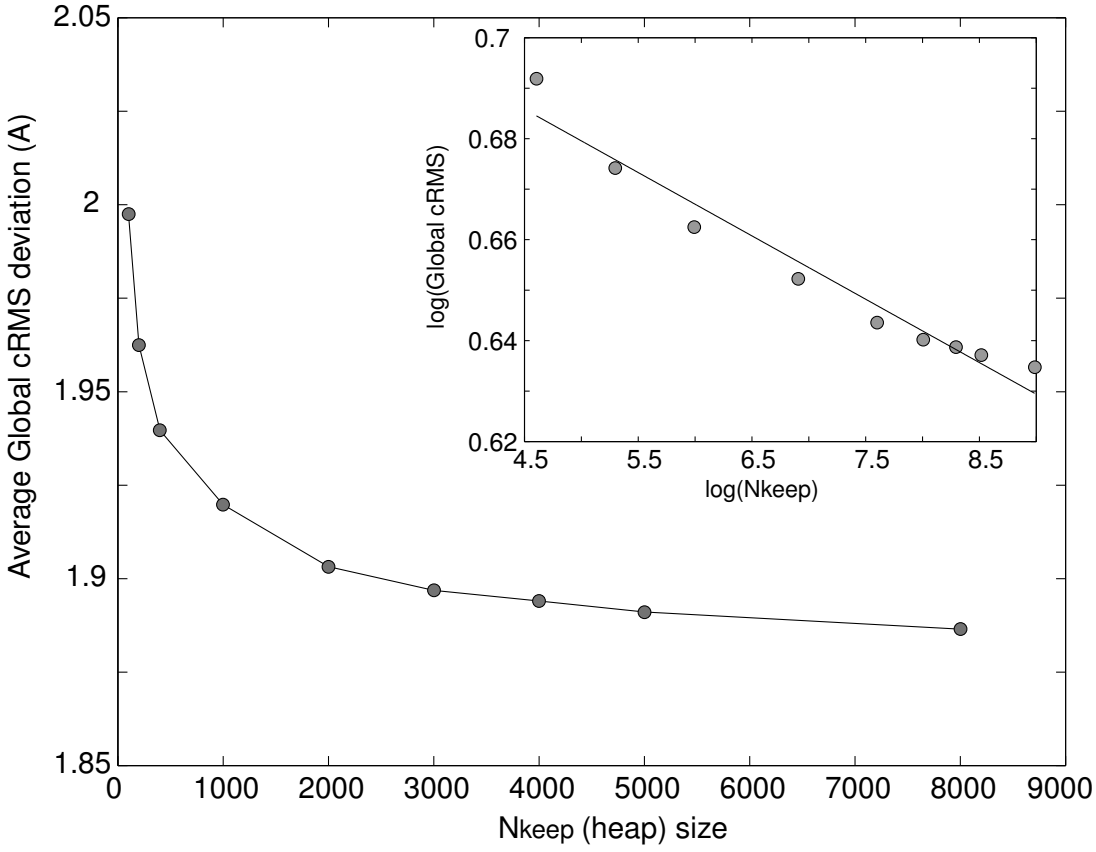

Figure 5.4: cRMS deviation of best approximation, averaged over the Park and Levitt [89] data set as a function of N_{keep} size. The library used to compile this data is of 20 fragments each 5 amino acids long. The insert shows the log(global fit cRMS) as a function of log(heap size). This same functional behavior is observed in other libraries as well, making $N_{\text{keep}} = 4000$ a reasonable choice for reconstruction from all libraries. From the log-log plot in the insert, we find: $\log(\text{global fit cRMS}) = -0.013 \log(N_{\text{keep}}) + 0.742$ or global fit cRMS $= 2.1 N_{\text{keep}}^{-0.013}$

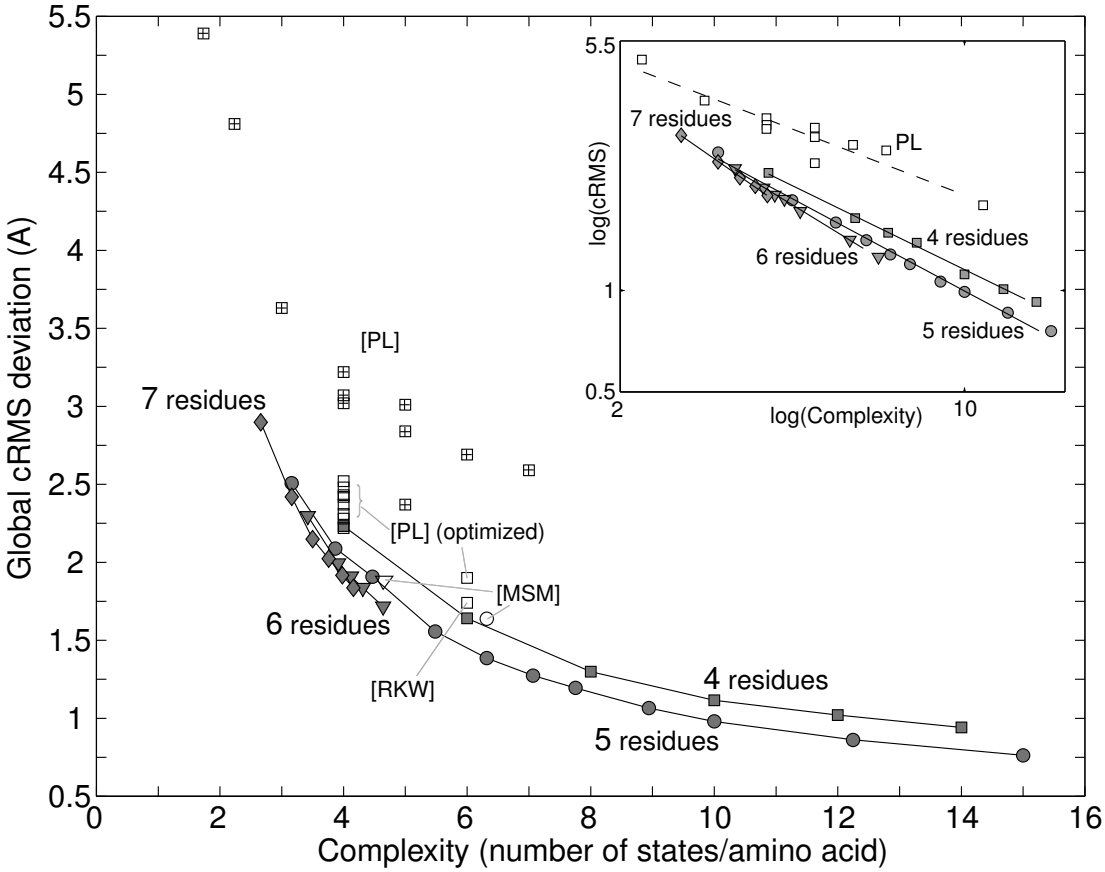

Figure 5.5: The average global cRMS deviation of the test set proteins is plotted as a function of the complexity of the library used for constructing the approximations. The libraries vary in size and are of fragments of lengths 4 (squares), 5 (circles), 6 (triangles) and 7 (diamonds), respectively. The libraries compiled in this study are shown in opaque shapes, while the libraries of Park and Levitt [89] (libraries with 4 residue fragments) and Micheletti *et al.* [78] (libraries with 5 and 6 residue fragments) are shown in hollow shapes. The insert shows the same data in log scale: the linear fit of this data is $y = -0.712x + 1.78, y = -0.78x + 1.80, y = -0.895x + 1.93$ and $y = -1.016x + 2.05$ for fragment lengths of $4, 5, 6$, and 7, respectively. More generally, the global fit cRMS depends on library Complexity and fragment length, f, as: $\log(\text{global fit cRMS}) = A \log(\text{Complexity}) + B$, where $A = -0.1039f - 0.280$ and $B = 0.094f + 1.373$.

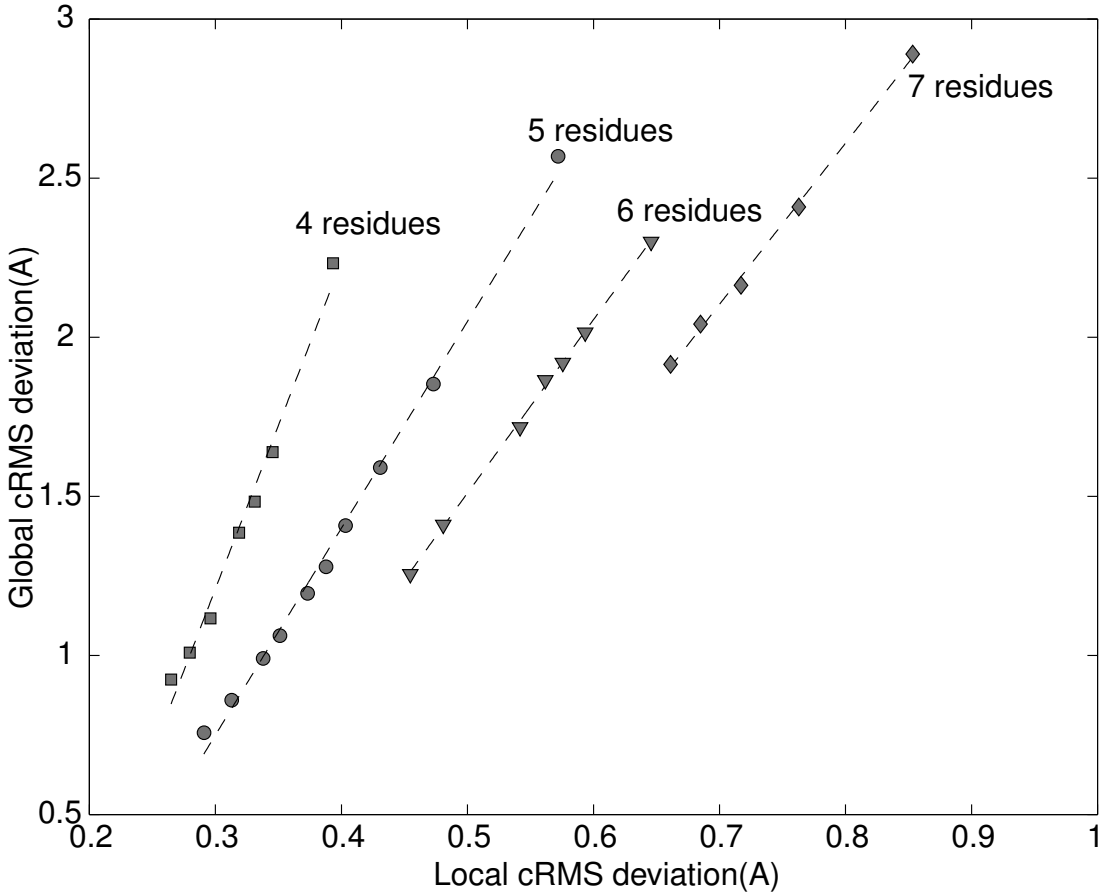

Figure 5.6: Global fit accuracy as a function of local fit accuracy. The average cRMS deviation of the global fit approximations is plotted as a function of the average cRMS deviation for local fit approximations for all proteins in the data set. Libraries of fragments of length 4, 5, 6 and 7 of various sizes are plotted here. The data in this figure is the same data of figures 5.1and 5.5, re-plotted for illustration. When fitting a line to the data, we have: $y = 10.277x - 1.874$, $y = 6.5x - 1.2, y = 5.45x - 1.21$ and $y = 5.066x - 1.443$ for fragment lengths of 4, 5, 6 and 7, respectively.

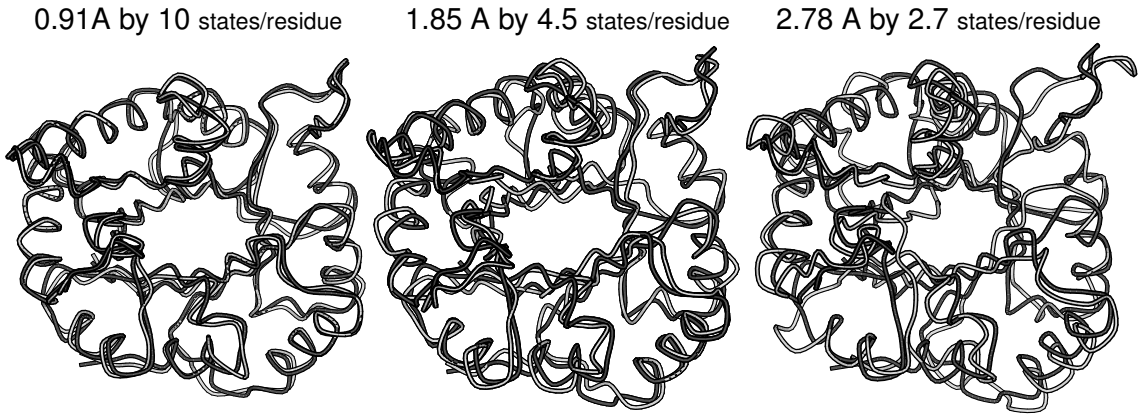

Figure 5.7: Three global fit approximations to the alpha-beta barrel protein with PDB identifier 1tim. The protein is drawn [68] in black and the approximations in gray. The libraries used when modeling the protein in: (a) has 100 fragments 5 residues each and achieves an overall cRMS distance of 0.9146 Å (10 states per residue); (b) has 20 fragments 5 residues each and achieves an overall cRMS distance of 1.8454 Å (4.47 states per residue), and (c) has 50 fragments of 7 residues each and achieves an overall cRMS distance of 2.7805 Å (2.66 states per residue). The clear improvement in global fit with increasing library complexity is apparent.

Chapter 6

Applications of Structure Approximations

We rely on the efficient approximation nets defined by the fragment libraries for generating candidate structures for protein structure prediction. As shown in Chapter 5, the fragment libraries give a set of structures that approximates real proteins well and whose local conformation is common in the PDB. Also, the set of structures that can be constructed by a library has a one-to-one correspondence with the set of strings of labels (up to some fixed length) of the library's fragments. Indeed, we can generate decoys by sampling these structures, and do so by sampling strings (or the fragments added when constructing the chain). Also, we can exhaustively enumerate all short chains for constructing loops. We conclude the chapter by pointing to other (unimplemented) applications of these approximations.

6.1 Decoy Generation

To assist in predicting protein structure, we propose candidate structures, or decoys, that are self-avoiding and compact structures. We sample them from the approximation net implied by a library, by randomly choosing the extending fragments. In this scheme, we only use the geometric nature of the protein (including its secondary structure properties), while ignoring all specific details of its amino acid sequence.

Despite the extreme simplicity of this method, the sets of decoys generated include many structures that have a cRMS deviation smaller than 6 Å from the native conformation. This method works well for small all-α proteins, and reasonably well for an α/β protein.

6.1.1 Methods

We use a specific library of 20 fragments, each five residues long[1]. We build the decoys chain by repeatedly extending the chain with library fragments, until reaching n residues. When adding a fragment, the first three residues of the fragment are used for positioning and orienting the fragment in space (with respect to the already constructed prefix of the chain), thus extending the chain by $5 - 3 = 2$ residues. Recall that a string of n_f fragments encodes a unique structure of $5 + 2(n_f - 1) = 2n_f + 3$ residues; equivalently, a structure of n residues that is constructed from library fragments is encoded by a string of $(n-3)/2$ library elements. The structures constructed from this library are protein-like in that their local chain conformations are similar to those observed in real proteins: the average local-fit deviation on the Park and Levitt test set is 0.47 Å. This library also generates structures that approximate proteins: the average global-fit deviation on the test set is 1.85 Å (See Table 5.3).

Sampling decoys. The total number of structures of n residues that can be constructed using this fragment library is $20^{(n-3)/2} \approx 4.47^n$. This set grows exponentially with the protein length, making enumeration tractable only for short chains (see Section 6.2). Instead, we add fragments at random, either uniformly or with a preference that is based on the "predicted" secondary structure sequence of the target protein.

In the biased scheme, denoted *biased sampling*, we choose the next fragment of the chain at random, according to a distribution of the library elements of the secondary structure sequence of the two residues that are added by this fragment. Secondary structure offers a coarse description of the backbone shape and can be predicted fairly well [56]. For a protein p of n residues, to a first approximation the secondary

[1] available at `http://csb.stanford.edu/rachel/fragments/libs/libs_dat/lib_20_z_5.txt`

structure sequence $t(p)$ is a string of symbols H, E and C, which indicates, for each residue, whether it is a part of a helix (H for helix), a strand (E for extended) or neither (C for coil). More formally: $t(p) \in \{H, E, C\}^n$, and $t^i(p)$ is the secondary structure at the ith position. We assume a perfect secondary structure predictor and use STRIDE [33] to calculate it from the protein's solved native structure. We emphasize that the protein sequence influences the decoy generation process only via the use of the "predicted" secondary structure sequence.

We associate with every fragment e a two-letter secondary structure string $t_2(e)$. We use the last two residues, because unlike the first three which position the fragment, these extend the constructed chain. Unfortunately, it is hard to associate a longer secondary structure sequence with each fragment (of three residues or more) due to insufficient statistics. By our experiments, other positions of the two residues in the fragment achieve similar results (data not shown). $t_2(e)$ is the last two letters of the full secondary structure string of a fragment $t(e)$, which is the segment in the secondary structure string of the originating protein that corresponds to the position of the fragment in the protein chain. We calculate the secondary structure string of the protein using STRIDE [33].

Next, we estimate the conditional probability $P(l|\sigma)$ for every library element l, and every two-letter secondary structure sequence $\sigma \in \{H, E, C\}^2$. These values are calculated from Bayes' law and $P(l)$, $P(\sigma|l)$ and $P(\sigma)$, that is, $P(l|\sigma) = \frac{P(\sigma|l)P(l)}{P(\sigma)}$. We estimate these probabilities from the observed frequencies in the clusters of fragments. Denote by cluster(l) the cluster of fragments with fragment l as its centroid. $P(l)$ is the probability a fragment is from cluster(l), and it is estimated by the fraction of the cluster's fragments in all fragments. $P(\sigma|l)$ is the probability that a fragment e in cluster(l) satisfy $t_2(e) = \sigma$. This is estimated by

$$P(\sigma|l) = \frac{|\{e \in \text{cluster}(l) | t_2(e) = \sigma\}|}{|\text{cluster}(l)|}.$$

As we are estimating probabilities by frequencies, the probability of an event may be too small for our sampling: when we do not observe any instances of a secondary

structure sequence σ in the cluster of a library fragment l, we assign a small probability $P(\sigma|l) = \epsilon$ (rather than 0, the observed frequency) to compensate for the finite sampling. Finally, $P(\sigma)$ is an easily calculated normalizing factor. Thus, the probability of sampling the sequence of library fragments $l^1, l^2, \ldots, l^{(n-3)/2}$, given the secondary structure sequence $t(p)$ is

$$P(l^1, l^2, \ldots, l^{(n-3)/2}|t(p)) = \Pi_{i=1}^{(n-3)/2} P(l^i|t^i(p)).$$

Self-avoiding and compact decoys. Our decoy structures satisfy two necessary geometric properties of proteins: self-avoidance and compactness. Self-avoidance requires a *lower* bound on the separation of any two C^α atoms, here we use 2.5 Å. Compactness requires an *upper* bound on the separation of any two C^α atoms, here we use B Å, where B varies from 20 Å to 60 Å. As could be expected, the majority of the structures are either not compact or self-intersecting, and this phenomenon is more pronounced in the more compact structures (smaller B). Since construction of a decoy structure is an expensive computation (constructing an n residue structure involves $(n-3)/2$ superpositioning steps, this could be problematic. Simply constructing decoy chain prefixes and discarding all the ones that fail to satisfy these geometric properties is too wasteful. Instead, we follow Rosenbluth and Rosenbluth [99] and use Monte-Carlo chain growing to sample only allowed chains.

Monte-Carlo chain growing. This technique considers only viable, or compact and self avoiding, extensions to the chain. As we construct the chain, we update the statistical weight of each chain to account for the altered distributions. For $l = 1, \ldots, 20$ define $v^i(l)$ to be 1 if l is valid as the ith fragment in the chain and 0 otherwise. For every extension of the chain, $v^i(l)$ is computed for all l values, increasing the computational cost of every chain extension by 20 superpositioning computations. We sample the extending fragment, either uniformly or with a bias, from the subset of valid fragments for the ith position, namely from $\{l|v^i(l) = 1\}$. If no library fragment is valid at the ith position, we discard the chain and restart the decoy generation process. Here, the already constructed chain is maintained unless we encounter a "dead end". In cases where some of the library fragments are

non-valid, we re-normalize the probabilities to sum to 1 by dividing the probability by $w_i = \sum_{i=1}^{20} v^i(l)P(l|t^i(p))$, the total weight of all valid fragments for that position. Note that when sampling uniformly the secondary structure sequence does not influence the probability value ($P(l|t^i(p) = 1/20$ for all $t^i(p)$).

Sampling only valid structures, implies that the sampling is from a different distribution than the one intended. Consequently, when estimating the value of a random variable, we need to account for the restricted sampling (only from a subset of all fragments) in different positions along the chain. The random variable of a sampled chain should be weighted by the portions of space we restricted ourselves to during the construction process. The weight of a decoy structure is $w = \Pi_{i=1}^{(n-3)/2} w_i$, or the product of all the constraints enforced along the way. The estimate for a random variable r from N samples should therefore be $\frac{1}{N}\sum_{k=1}^{N} r_k w_k$, where r_k is the random variable value for the k-th constructed chain and w_k is its weight.

6.1.2 Results

We generate decoys for four proteins with PDB codes 1enh, 4icb, 2cro and 1ctf, all are relatively short (54, 76, 65 and 69 residues respectively). The first three proteins are all alpha: 1enh has three helices, 4icb and 2cro each have four helices; 1ctf is an alpha and beta protein with three helices and three strands. Using the greedy procedure described in Chapter 5, we construct approximations to the structures of these proteins using the library of 20 five residue fragments. Since this procedure uses the known native structure, it is clearly not a valid for generating protein decoys. Nonetheless, it assures us that the discretized conformation space generated from this library has structures that are sufficiently similar to the structures of the target proteins. The best approximations we found are of 1.41 Å, 1.60 Å, 1.41 Å, and 1.43 Å cRMS from the native structures of 1enh, 4icb, 2cro and 1ctf, respectively.

Figure 6.1 shows the histograms of the cRMS deviations of the structures from several 1enh decoy sets from the native conformation. Each decoy set has $400,000$ decoys; we use the following thresholds for the maximal distance constraints: 20 Å (panel (A)) , 30 Å (panel (B)), 40 Å (panel (C)), and 50 Å (panel (D)); the maximal

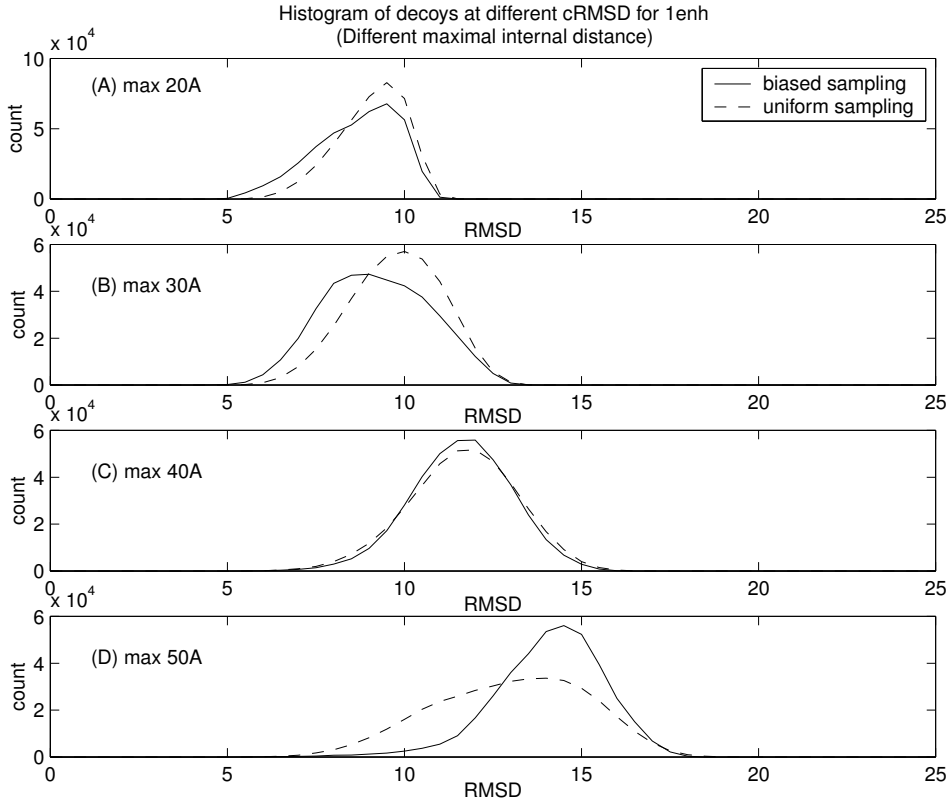

Figure 6.1: Distribution of the cRMS deviations (in Å) of 400,000 decoys for protein 1enh when enforcing maximum distance constraints of (A) 20 Å (B) 30 Å (C) 40 Å and (D) 50 Å, using biased and uniform sampling. The most native like decoys are the compact one that are sampled using a bias based on the protein's secondary structure.

distance in the native structure of 1enh is 26 Å. In each case we construct two sets: one using uniform sampling (dashed line), and one by biased sampling (solid line). Here, better decoys sets have lower cRMS deviations or their histograms are shifted to the left.

Figure 6.1 shows that the decoys sets with a tighter compactness constraint have more native-like structures. Biased sampling generates better decoy sets, as compared to uniform sampling, when the structures are relatively compact (maximal distance less than 30 Å). Surprisingly, biased sampling generates worst decoy sets when the structures are relatively open (maximal distance greater than 40 Å). In the case of

1enh, the open structures that biased sampling produces are open chains with rigid helical parts embedded in them. These open structures are all fairly far from the native state, yet the more compact ones seem to be more native-like (even though they do not actually resemble the protein's structure). The decoys sampled with bias, are constrained by the rigid parts along the chain, and have fewer positions (the non-rigid ones) in which they may turn back to make a compact structure. Thus, they appear less native-like. The different shapes of the histograms further support this explanation: the uniform sampling histogram is broader when the maximal distance constraint is relaxed, while the biased sampling histogram is of the same width only shifted to the right; this behavior is even more pronounced when considering larger maximal distances (e.g., 80 Å; data not shown). Similarly, the best decoy sets generated for the other proteins were the ones generated using biased sampling, enforcing highly compact structures.

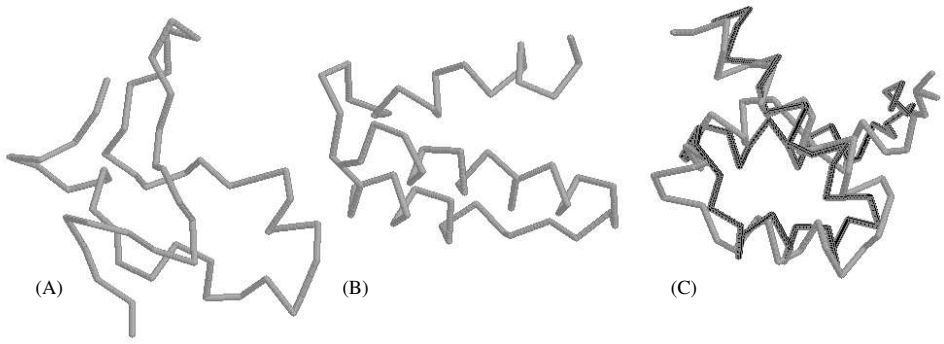

Figure 6.2: Three decoy structures for 1enh: (A) A typical uniform sampling decoy (5.94 Å cRMS deviation from native structure). (B) A typical biased sampling decoy (5.72 Å cRMS deviation). (C) The best decoy, generated by biased sampling, superimposed on the native structure (3.9 Å cRMS deviation, the decoy is in the darker tone).

The biased sampling generates decoys with secondary structure, which appear more "protein-like", while the uniform sampling generates decoys with a more "scrambled" look. Figure 6.2 shows examples of sampled decoys: Panel A shows a typical "scrambled" decoy, Panel B shows a typical decoy generated with biased sampling. The best decoy is 3.9 Å cRMS from the native structure and both are superimposed

in Panel C. We see similar behavior in all four decoy sets.

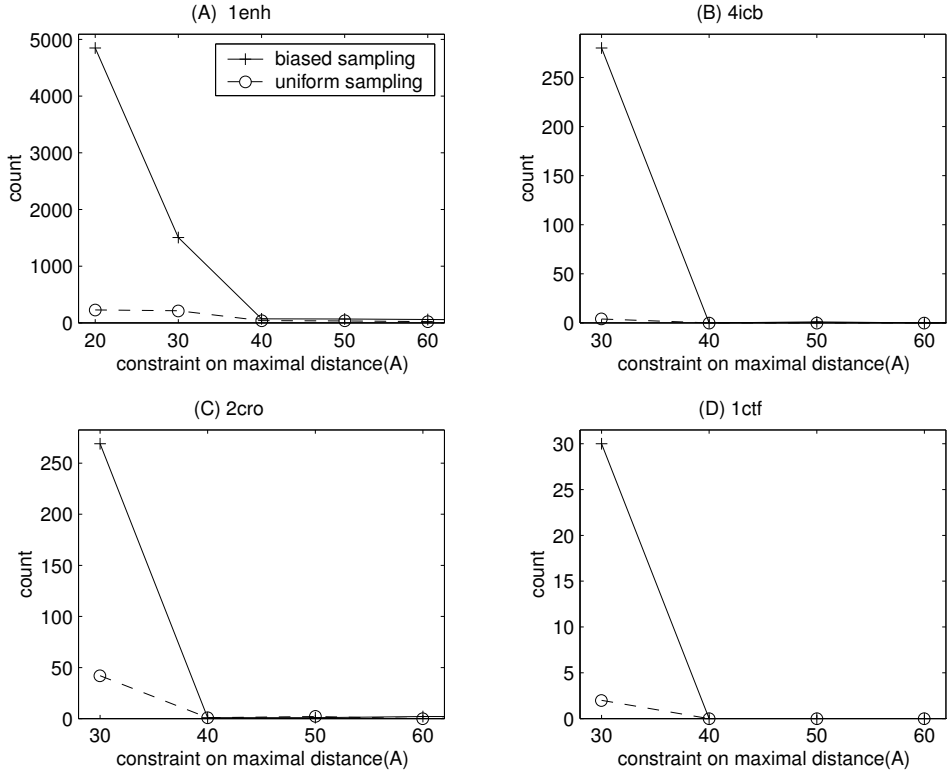

Figure 6.3: The number of good decoys (cRMS deviation < 6 Å) as a function of the compactness constraint for 1enh, 4icb, 2cro and 1ctf. Each of the graphs shows the number found when generating decoys using biased (solid line) and uniform sampling (dashed line).

Following Reva *et al.* [95] we consider a decoy "good" if its cRMS deviation is less than 6 Å from its target protein native structure. Figure 6.3 shows the number of good decoys found by the two sampling procedures, for the proteins 1enh, 4icb, 2cro, and 1ctf. The maximal distance between two C^{α} atoms in the native structures of these proteins is 26 Å in 1enh, 30 Å in 4icb, 26 Å in 2cro and 30 Å in 1ctf. In all cases, biasing the sampling generates more good decoys than sampling uniformly; this effect is strong for 1enh and more subtle for 1ctf. Also, enforcing strict compactness constraints generates better decoys. The total number of good decoys found is encouragingly high at 1507, 280, 269 and 30 structures out of 400,000 generated for 1enh, 4icb, 2cro and 1ctf, respectively, when using a compactness constraint of 30 Å. For 1enh, the

results are even better with a compactness constraint of 20 Å with 4849 good decoy structures.

The cRMS deviation of the best decoys found was less than 5.0 Å cRMS from the corresponding native structures: 3.94 Å for 1enh, 4.23 Å for 4icb, 4.94 Å for 2cro and 4.95 Å for 1ctf. In all cases, the best decoy structure is found with a 30 Å compactness constraint and biased sampling.

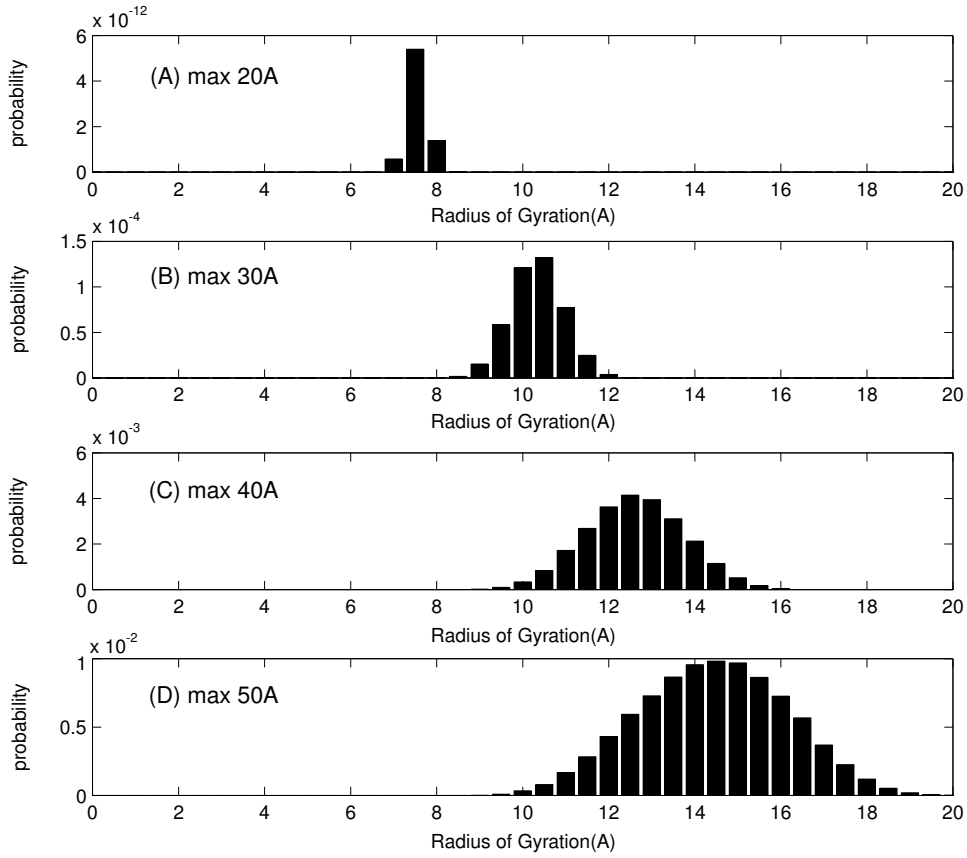

Figure 6.4: The estimation of distribution of radius of gyration for 1enh decoys with biased sampling. The maximum distance constraint is (A) 20 Å (B) 30 Å (C) 40 Å, and (D) 50 Å.

Figure 6.4 shows the estimated distribution of the radius of gyration of the decoy structures generated with biased sampling for 1enh. The distribution also normalized by 400,000 so that the integral of the histogram gives an estimation of the fraction of structures occupied by geometrically valid decoys. As expected, decoys generated

with a smaller maximal distance constraint have smaller radii of gyration. We can also see that the fraction of space occupied by the valid decoy structures decreases dramatically as the maximal allowed distance is decreased.

6.1.3 Discussion

We generate decoys by sampling a discrete and relatively small space of conformations that approximates proteins well. As the space is far larger than our sampling set, we try to sample the relevant regions, i.e., compact, self-avoiding structures with the secondary structure similar to the target protein. The two prominent characteristics of our decoy generation scheme are: (1) the amino acid sequence of the target proteins is not considered — only its secondary structure is used, and (2) the construction is based on the local geometry of other native proteins.

Almost all decoy generation techniques incorporate the amino acid sequence of the target protein. The information flow from the amino acid sequence into the construction of the decoys varies: some import it via the scoring function [115] while others search the PDB for structures with similar amino acids and generate decoys with resembling structural pieces. In particular, Simons *et al.* [110] seek structures with similar consecutive triplets of amino acids while Huang et al. [53] consider non-consecutive pairs of residues with similar amino acids. Here, we explore the boundaries of decoy generation schemes by asking: "How good are decoys when using only secondary structure information?". Surprisingly, although we use significantly less information about the proteins, the decoys we generate are qualitatively very similar to those found by others. For instance, the best decoy for 4icb found by Huang *et al.* [53] has a cRMS deviation of 5.0 Å from the native conformation of the protein (compare with our value of 4.9 Å). Simons *et al.* [110, 109] found better decoys than ours: the best 2cro decoy at 4.2 Å (compare with 4.9 Å), the best 1ctf at 3.46 Å (compare with 4.9 Å) and the best 5icb decoy at 3.74 Å (compare with 4.2 Å).

This decoy generation method uses only the local geometric properties of protein backbones as implied by the shape of the fragments in the library. We do not employ any observations regarding common geometric structures of non-consecutive residues.

The method works best when generating decoys for all alpha proteins, and not as well for alpha and beta proteins. This suggests that conforming to correct local geometric features along the chain may suffice to imply the structure of an all-alpha protein. In a way, it is complementary to the work of Fain and Levitt [27], which showed that conforming to correct non-local geometric features suffices to imply the structure of an all-alpha protein. Indeed, the stabilizing interactions in helices are close along the chain, while those that involve beta strands are not, and hence are not captured correctly by our scheme.

These observations suggest several directions for improving the generation of decoys from fragment libraries: (1) incorporate non-local geometric features; (2) incorporate a scoring function into the scheme allowing searching rather than sampling; (3) enrich the library with amino acid sequence information, and use this information to bias the sampling.

6.2 Loop Building

Homology (or comparative) modeling techniques address the easier task of predicting protein structure given the structure of a closely related (homolog) protein. The predicted structure uses the structure of the homolog protein as a template and modifies it in the regions in which the two differ [9]. Initially, we align the sequences of the target and template proteins, and the parts of the template structure corresponding to conserved regions in this alignment are copied, defining the three-dimensional framework for the structure of the target protein. The variable regions in the alignment correspond to the gaps in the framework. These are usually the results of substitutions, insertions and deletions of residues between members of the same structural family, and frequently corresponds to exposed loop regions that connect elements of secondary structure in the protein fold. In the regions that do not align well, we remove existing parts and reinsert new structures of the appropriate length that fit between the end points. Figure 6.5 gives an overview of homology modeling.

As in *ab initio* structure prediction, we assume a scoring function that can score the generated structures and separate the native like structures from the unrelated

ones. We emphasize that the framework is not exact, and we are only looking for approximate solutions. Thus, in testing the fitness of a candidate loop in a gap of the framework, we tolerate 1 Å in the positions of its end points. This 1 Å should be compared to the length of each link of the loop, i.e., 3.8 Å. Despite this approximation, loop building remains a very difficult, unsolved problem in comparative protein modeling.

6.2.1 Methods

We build protein loops by concatenating fragment from several fragment libraries, all five residues long. For longer loops, we mitigate the combinatorial explosion by building in two parts, one starting at each of the ends of the gaps in the framework and testing for closure where the two parts should join. The coarseness of the sampling is defined by the size of the library of fragments used to build the loop. This method combines the advantage of exhaustive enumeration, with the ability of the database approach to generate segments that are locally physically reasonable.

Searching the database for loops. For comparison, we search for existing loop structures, following a procedure suggested by Jones & Thirup [55] and Summers & Karplus [114]. We search in a non redundant set of 3,307 SCOP 1.63 [81] domains, whose structures were accurately determined using X-ray crystallography ($R < 2$ Å); this set can be retrieved from the ASTRAL compendium [8]. The representative set has diverse sequences: no two have a BLAST [3] E-value greater than 0.0001. A loop fits a template when we can accurately superimpose the anchor residues of the loop on the anchor residues of the gap in the template. The anchor residues are the three C^{α} atoms immediately preceding and following the candidate loop (note that this filter is sequence independent) and the gap in the template, respectively; we consider their $\binom{6}{2} = 15$ interatomic distances, denoted RNn in the loop and TDn in the template structure. We quantify the agreement of the loop and the template by the root mean square deviation:

$$RMS(TDn, RNn) = \sqrt{\frac{1}{15}\sum_{i=1}^{15}(TDn(i) - RNn(i))^2}$$

The selected loops must also match specific conformational restrictions related to the *sequence* of the chain. Non-Gly residue conformations must satisfy ($\Phi_{Non-Gly} < 0$, $\Psi_{Non-Gly} > 0$). Pro residue conformations are restricted to $\Phi_{Pro} \in [-90°, -30°]$ and $\Psi_{Pro} \in [-85°, 0°]$ or $\Psi_{Pro} \in [115°, 175°]$, and residues X preceding a Pro must be characterized by $\Phi_X \in [-210°, -30°]$ or $\Phi_X \in [30°, 90°]$, and $\Psi_X \in [60°, 180°]$ [114]. All loops that satisfy all these criteria and that agree with the anchor residues with RMS ≤ 1 Å are returned by the database search.

Generating loops using libraries of small protein fragments. We use libraries[2] of $L = 20, 40, 60, 80$ and 100 fragments, each five residues long. Recall that the fragment libraries do not contain any information about the sequence of the proteins from which they were built. We construct all loops generated by the library fragments, and select the ones that fit between the anchor residues. Unfortunately, most loops do not fit. Note that due to the noisy nature of experimentally determined structures, we accept loops that fit within 1 Å in the atomic position.

Strategy A: Unidirectional construction (for short loops). There are several (essentially equivalent) ways for enumerating all loops of length l. We can construct a chain of $l + 5$ atoms, starting by overlapping 3 residues on the anchor points on the N terminal side of the gap in the framework, and test the position of the last two atoms of the chain, with respect to the position of the first two anchor points of the anchors at the C terminal side of the loop. Overlapping the last 2 atoms guarantees the "protein-like" structure at the attachment point. Similarly, we can construct a loop from the C-terminus towards the N-terminus. Alternatively, we can construct systematically all chains of $l + 4$ atoms and try and overlap 2 anchor residues on each end. Among these, we use the first strategy described above; a two-dimensional analog of this option is depicted in Figure 6.6. The total number N of chains of length

[2]available at `http://csb.stanford.edu/rachel/fragments/`

ll to consider is

$$N = s^{\lceil \frac{ll}{2} \rceil + 1}, \tag{6.1}$$

where s is the number of fragments in the library. Only a small fraction of N corresponds to valid loops, i.e., chains that solve the loop closure problem.

Strategy B: Bidirectional construction (longer loops). While strategy A is well adapted for short loops ($ll < 9$), it fails for longer loop because of the combinatorial explosion in the number N of chains to generate (see Equation 6.1). Instead, we generate half loops starting from the two anchors in the framework, and assemble the half loops that (approximately) overlap at their end points. Using this procedure, we enumerate chains of length that is only half the length of the loop, resulting in a reduction of the running time by a factor of 2 in the exponent. Two half-loops meet when the last 2 C^α atoms of the first half-loop are approximately positioned on the first 2 C^α atoms of the second half-loop. Overlapping 2 atoms, guarantees that the positioning of the residues around the meeting point is "protein-like". A two-dimensional analog of this type of construction is described in Figure 6.6. Since storing all half-loops is impractical, we generate loops via three steps:

(i) Mark all positions in space that are end points of the last two C^α atoms of a half loop generated from the N-terminal anchor. Since the data points are noisy, voxels with 1 Å resolution suffice to describe the three-dimensional space. The amount of memory used to store these end pairs can be further reduced if we take in account the fact that the distance between two consecutive C^α atoms is fixed.

(ii) Enumerate all half loops generated from the C-terminal anchor, and store those that are part of a valid loop. These are the half loops with end points that fall in the same voxel as previously marked positions. We also mark voxels that are the meeting points of two half-loops.

(iii) Re-generate the first half-loops. For every half-loop that ends in a marked voxel, lookup the complementing half-loops and assemble the full loops.

We implemented both methods and observed similar results for short loops.

6.2.2 Results

Accuracy of loop predictions. There will generally be a range of accuracy in predicting loop conformation for different loops. It is therefore necessary to assess the quality of a method by testing it on a large selection of loops. Our test set, which we refer to as the SALI set, contained 427 loops selected from the list reported by Sali and co-workers [30]. The length of the loops ranged from four to fourteen residues, and it defines the residue numbers of the target loops. The initial SALI set contained 40 proteins for each loop length; we remove proteins that are now considered obsolete in the PDB.

The accuracy of a single loop prediction is measured by comparing it to its native conformation. A large variety of criteria for comparing loop conformations exist. They range from cRMS measures on different sets of atoms (C^α only, or all main chain atoms), to dihedral angle and dihedral angle class comparison. Here, we use cRMS, computed over Cartesian coordinates, which is sometimes called "global" cRMS [30]. It is computed by finding the optimal superposition [57] of the anchor residues of the test loop and native loop, respectively, and summing the subsequent differences in the positions of the C^α of the two loops. The global cRMS provides both a measure of the local fitness of the candidate loop with respect to the native loop, and a measure of the quality of its positioning in the framework.

We compare our approach to naively searching the database for loop candidates of appropriate length and selecting the candidates with well-fitting anchor residues. This is different from our approach of systematically exploring a discrete approximation of the conformational space accessible to the loop considered. Note that the tests described in this study do not reproduce true homology modeling experiments as we use the native conformation of the protein as the framework to build the loop.

Figure 6.7 shows the average cRMS of the best fitting fragment-based loop computed over all target loops of a given length ll, for different libraries of fragments. Table 6.2.2 gives the range of cRMS for best fitting loops over all target loops of a

Loop	Number of fragments in Library (s)					
Length	20	40	60	80	100	DB
4	NA	NA	NA	0.31–4.75	0.32–4.21	0.21–2.47
5	NA	NA	0.93–4.87	0.54–4.66	0.28–3.02	0.38–3.33
6	NA	NA	0.46–4.60	0.34–3.90	0.38–3.42	0.28–3.51
7	0.43–4.84	0.56–4.49	0.45–2.85	0.34–2.54	0.41–2.69	0.40–4.79
8	1.15–4.97	0.53–3.89	0.47–2.75	0.37–2.45	NA	0.57–4.74
9	1.51–4.98	0.76–3.44	0.75–2.68	0.82–2.62	NA	0.40–6.58
10	1.92–4.95	1.01–3.55	0.82–2.68	NA	NA	0.47–6.45
11	1.78–4.89	1.29–3.52	1.14–2.51	NA	NA	0.65–6.90
12	1.41–4.18	1.05–2.83	NA	NA	NA	1.14–8.33
13	2.17–4.72	1.04–3.50	NA	NA	NA	0.65–7.37
14	2.15–4.28	1.46–3.10	NA	NA	NA	1.32–7.66

Table 6.1: Range of cRMS (Å) for the best fragment-based loops

given length. Finally, Table 6.2 shows the average number of fragment–based loops generated within 3 Å and 4 Å of their target loops; the average is computed over all target loops of the same length.

Figure 6.7 shows that the quality of the fragment-based loops improves as the size s of the fragment library increases. This improvement however comes at a cost. For long loops, the use of large libraries results in significantly more candidate loops, since the total number generated by this procedure is a high order function of s (see Equation 6.1). Also, it is not possible to generate fitting loops using all combinations of library size and loop length. For short loops and small fragment libraries, most if not all candidate chains do not close, within the 1 Å tolerance we have set (see 6.2.1). For example, we can only build loops of 4 residues with libraries of 60 or more fragments.

For very short loops ($ll < 6$), the database approach described in 6.2.1, selects better loops than those built from the fragments (see Figure 6.7). This result is not surprising: Recall that fragment libraries are "compressed" versions of the database of structures, or equivalently, the database has a better sampling of these short loops. Conversely, the fragment-based approach to loop building performs much better than

the database approach for loops larger than 6 residues, and this occurs even for small fragment libraries. Table 1 shows also that systematic sampling provides more consistent quality of the best fitting loops between target loops of the same length. Table 6.2 illustrates that even with libraries of small size, the systematic search reliably builds candidate loops within 3 Å of their target, for loop length up to 14 residues.

Loop Length	Number of fragments in Library (L)				
	20	40	60	80	100
4	NA	NA	NA	40 (57)	66 (91)
5	NA	NA	4 (7)	12 (26)	36 (71)
6	NA	NA	1.7 (3.9)	5.6 (14)	14 (32)
7	0.3 (0.3)	2.5 (6.7)	18 (46)	56 (162)	123 (368)
8	1.3 (3.8)	27 (108)	258 (1016)	1354 (5355)	NA
9	0.8 (2.6)	14 (82)	163 (800)	845 (4358)	NA
10	0.3 (1.3)	4.2 (34)	49 (339)	NA	NA
11	0.5 (4.1)	26 (220)	203 (1640)	NA	NA
12	6.8 (53)	420 (4448)	NA	NA	NA
13	1.7 (20)	129 (2135)	NA	NA	NA
14	0.8 (13)	63 (1257)	NA	NA	NA

Table 6.2: Average number of fragment–based loops within 3 Å (4 Å) of their target loop

Running Times. Table 6.3 lists the average (over the test set) amount of central processor unit time (CPU minutes) needed to enumerate and evaluate all loops. All programs were run under the Linux operating system (RedHat 7.3) on a cluster of dual 2.8 GHz Intel Xeon processor machines, each with 1 Gigabyte of memory. As expected, it takes more time to enumerate all longer loops, and larger libraries offer more possibilities, again running longer.

6.2.3 Discussion

Loop closure is an essential element of many protein structure prediction problems. It is nearly always needed in homology modeling, where the framework for the structure of the target protein is derived from highly homologous regions in a template protein, leaving gaps between these regions that need to be filled in with protein segments

Loop Length	Number of fragments in Library (L)				
	20	40	60	80	100
4	NA	NA	NA	0.0303	0.0435
5	NA	NA	0.0202	0.0307	0.0448
6	NA	NA	0.0210	0.0320	0.0462
7	0.0182	0.0818	0.0973	0.1525	0.1847
8	0.0359	0.0896	0.1639	0.3534	NA
9	0.0382	0.1074	0.1850	0.4415	NA
10	0.0376	0.1022	0.1753	NA	NA
11	0.0778	0.3803	2.2904	NA	NA
12	0.1387	2.6720	NA	NA	NA
13	0.1439	3.5776	NA	NA	NA
14	0.1579	4.9761	NA	NA	NA

Table 6.3: Average CPU minutes per loop on a 2.8 GHz processor

(see Figure 6.5).

Searching the PDB for loops that satisfy a geometric fitness criteria [114] assumes that there exists a least one segment from a known protein structure that matches the target loop. In 1994, Fidelis *et al.* [29] showed that unfortunately this assumption is not valid for loops longer than 4 residues. We observe, similarly to vanVlijmen and Karplus [122] that the database approach performs well for all loops shorter than 9 residues. In a recent study, Du *et al.* [24] showed that for a protein chain of 15 residues, there is a 91% probability of finding a non homologous protein segment in the PDB within 2 Å cRMS. Based on these results, they concluded that the database approach to loop building should perform well up to 15 residues. We argue that, in fact, their results are similar to ours for loops of 9 residues. The search in the PDB of candidate segments that fit in a gap of a protein is successful when the segment found is structurally similar to the native conformation of the target loop, and if the flanking regions of the segment in the protein to which it belongs, matches the anchors on both sides of the gap. The geometric filter applied to the candidate segments enforces the second condition and defines the proper orientation of the segment in the framework. A successful segment for a loop of nine residues must therefore match the target loop and its stem regions, giving a total of 15 residues in our procedure.

Even with the steady increase in size of the PDB, it is not clear that one can expect

to find longer loop structures in the database in the near future. Specifically, our results use the current database, and are similar to those achieved in 1997 [122]. The number of conformations that a protein loop can adapt grows exponentially with its length. For long loops (i.e. longer than 9 residues), it may be too large to be properly sampled in the PDB, even if we was to multiply its size by an order of magnitude. Our results indicate that there is in fact no need to wait for much better statistics in the distribution of long protein segments. As previously reported [24], the current PDB provides a good sampling of the conformations of protein fragments of 5 residues, and we can cluster these fragments into libraries. Since these libraries contain only a small number of fragments, we can exhaustively enumerate all loop conformations obtained by concatenating these fragments. The size of the library controls the trade-off between feasibility and accuracy. Loop building based on libraries with a large number of fragments generates accurate loops of length up to 8 residues. For longer loops, the procedure is more costly in computing time, in which case the smaller libraries still provide satisfactory loops, at a much reasonable computing cost.

Our method is reminiscent of the PDB based loop prediction method of Sudarsanam *et al.* [113], which constructs a chain based on a ϕ_{i+1}, ψ_i dimer database. Their results on the construction of short target loops of 5 residues are comparable in quality to those reported here. As they did not cluster their large database of dimers, they could not perform exhaustive construction of candidate loops. Another difference is that unlike our sequence independent approach, Sudarsanam *et al.* binned the dimers into 400 categories, based on the amino acid pair. Lastly, their database of ϕ_{i+1}, ψ_i dimers was built selectively from residues belonging to loops regions in proteins, removing all residues that belong to an α-helix or β-sheet. We have tried a similar approach by designing loop-specific fragment libraries, which are used to generate candidate loops that can fit in a gap in a protein framework. These structure-specific libraries did not perform better than the general libraries used above (results not shown).

Our approach combines the advantages of *ab initio* and database search methods. Similarly to *ab initio*, we can enumerate all approximation structures that satisfy the end-to-end constraint of the target loop. Similarly to database searches, using libraries

of fragments guarantees that the backbone conformations are physically reasonable. By varying the size of the library of fragments used to build the loops, we control the trade-off between accuracy and feasibility in terms of computing time. Previous database approaches to the loop prediction problem were limited to loops of up to 9 residues. Here, we extend it to loops of 14 residues.

6.3 More applications of approximation nets

The approximations to protein structure that are generated by the fragment libraries are fully described by strings of fragment labels. Thus, we can describe sets of protein structures (e.g., the PDB) by their corresponding strings. This may allow importing tools that were developed for string analysis to structure analysis; a good review of string analysis tools can be found in [42].

String analysis tools help investigate properties of DNA sequences. It is possible, that similar properties of protein structure can be studied using our structural strings. For example, we can investigate structural entropy following studies of the entropy of DNA sequences [28, 102]. Alternatively, studies of the statistical composition of DNA, and in particular, of unusually common and uncommon sequences [94] suggest a method for analyzing structural composition. Another potentially useful tool is suffix trees [120], which may find structural motifs by identifying long common strings.

When using our string representation for protein structure, one must keep in mind a few additional details. There are many equally good approximating structures for a protein and different approximations are specified by different fragment labels. Namely, the string representation of structure is not unique. Furthermore, the string representation is found using a greedy heuristic. For almost identical structures, we may find different string representations because of the greedy search procedure. Also, the string representation of structure effectively encodes the angles along the backbone of the protein, implying that a small change in a single position may result in a large change in the Cartesian coordinates. Thus, a naive comparison of strings can identify between almost similar strings that define very different structures.

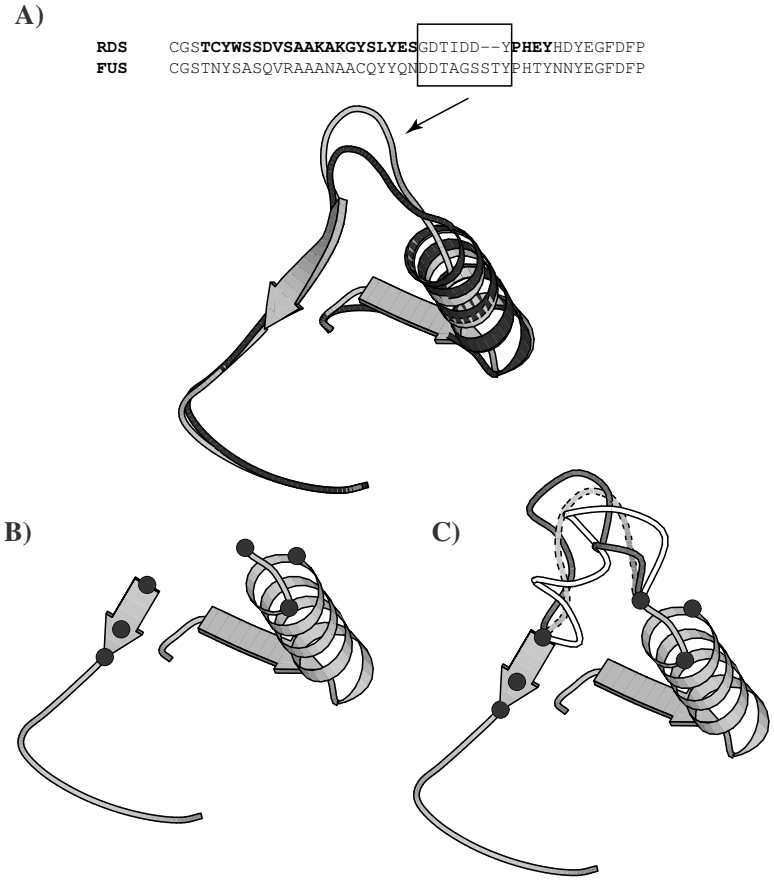

Figure 6.5: Homology modeling demonstrated on the toy experiment of predicting the structure of 1fus. Homology modeling has 3 steps: (A) *identifying a "template" structure*. 1rds is a close homolog of 1fu (their sequences are very similar; FASTA [90] E-value: 10^{-22}, and 57.8 % sequence identity in 102 amino acid overlap). For illustration, we use STRUCTAL [112] and structurally align the structures of 1fus and 1rds (in grey and black respectively); only residues $20-50$ shown. Indeed, conserved regions in the sequence correspond to conserved regions in the structural (cRMS = 0.8 Å over 102 residues). The two structures differ in the loop region between two secondary structures (highlighted in bold in the sequence alignment), and the loop in 1fus is two residues longer. (B) *Building the framework*. The framework for 1fus is the backbone regions of 1rds, that correspond to the conserved secondary structure. This framework has a 9 residues gap. The three residues preceding and following the gaps are the "anchors" of the constructed loops (marked with balls centered at their C^{α}) (C) *Loop building*. Fragments of 9 residues with appropriate end-to-end geometry are selected in a database of protein segments, or built using small libraries of protein fragments. The native loop is shown in a dashed line, the best loop found in the database is in grey (5.6 Å) and in white the best one using fragments (2.7 Å). This figure was generated using MOLSCRIPT [68].

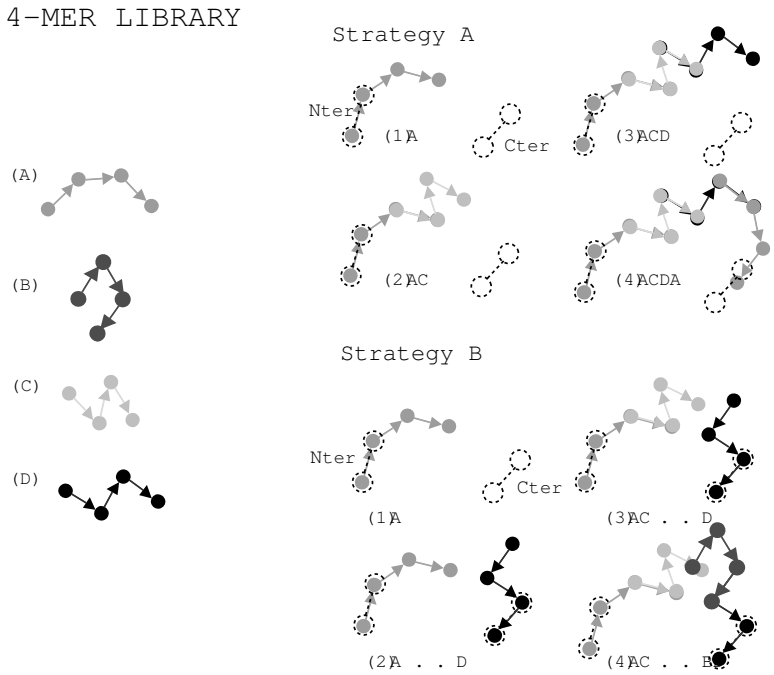

Figure 6.6: Loop construction using a library of fragments. We demonstrate building a two-dimensional loop of 7 residues using a library of four-residue long fragments. The anchors of the loop in the framework are shown with dashed lines (two anchor points on each side), and labels Nter and Cter, corresponding to the N terminal and C terminal ends of the loop. In *Strategy A* we construct the loop from the left anchor. In this example, the loop ACDA ends (approximately) on the first right-hand anchor point. In *Strategy B*, we build the loop from both ends of the gap. In this example, the last residue of extension AC and the first residue of extension BD (approximately) overlaps. Notice that library fragments can be re-used, and that the chain has a direction. Because this is a two-dimensional example, positioning a new fragment on an existing extension requires only 2 residues, and loop closure is checked over 1 residue. In three dimensions, positioning requires 3 residues, and loop closure is checked over 2 residues.

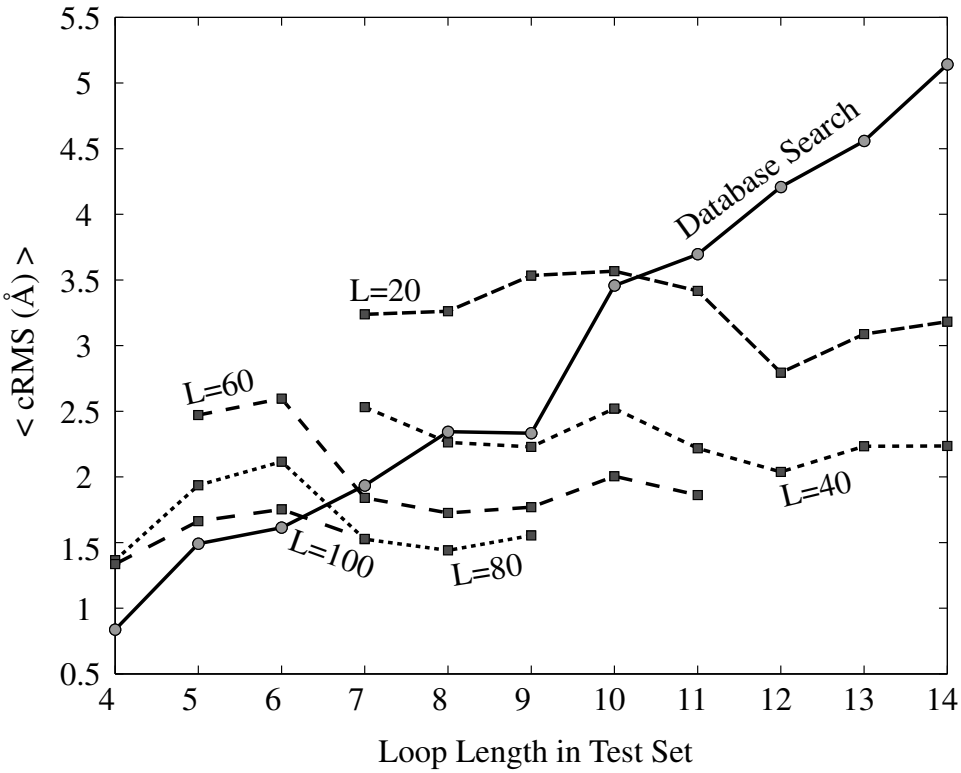

Figure 6.7: A fragment-based approach to the loop building problem. We build candidates for each of the 427 target loops in the set, using small libraries of protein fragments of five residues. The quality of these candidates is measured by the cRMS deviation from the target loop, computed over all loop C^{α} atoms, after superpositioning the flanking regions of the native and candidate loops. For each target loop, we consider only the cRMS of the best fitting loop, and average over all target loops of that length. The figure shows these average <cRMS> values as a function of loop length, for different library size L. For comparison, the solid line shows the average cRMS of the best loops found in the database of known structures.

Appendix A

Fragment clustering

A.1 Simulated annealing k-means

The *simulated annealing k-means* clustering heuristic (abbreviated SA k-means) incorporates the "probabilistic rejection" principle of simulated annealing into the k-means algorithm. This heuristic better avoids entrapment in unfavorable local minima. SA k-means repeatedly executes the following three-step procedure: (1) run k-means [25] until convergence (2) modify the clustering into a potentially better one. (3) accept or reject the transformation probabilistically. Figure A.1 lists the pseudo code.

The modification step merges two clusters and splits a cluster, maintaining the total number of clusters. It is designed to escape a local minima with two very close clusters. Both the merged cluster pair and the split cluster are selected at random, where pairs of closer clusters are more likely to be merged and a larger cluster is more likely to be split. The distance between a pair of clusters is measured by the cRMS deviation of their centroids, where a centroid is the element with minimal sum of distances to all other elements in the cluster. We use the density function $1/x^2$ (normalized), where x is the distance between clusters. The size of a cluster is the maximal cRMS deviation of two elements in the cluster. Here, we use a linear density function.

The total variance serves as the clustering score. We accept the modifications

```
temperature = INITIAL_TEMP_VALUE;
initialize_clustering_at_random();

while (temperature > TEMPERATURE_LOWER_BOUND){
        run_k_means();
        old_score = calculate_clustering_score();

        (m1, m2) =  pick_clusters_to_merge_at_random();
        m1 = merge(m1,m2);          //merges m1 and m2 into m1
        s1 = pick_cluster_to_split_at_random();
        (m2, s1) = split(s1);       //splits s1 into m2 and s1
        new_score = calculate_clustering_score();
        pr_accept = exp((old_score - new_score)/temperature);
        r =  rand();                //r in [0,1]
        if (pr_accept < r) then {
                undo_merge_and_split(m1,m2,s1);
        }

        temperature = temperature * COOLING_FACTOR;
}
```

Figure A.1: Pseudo code for simulated annealing k-means algorithm

in a Monte Carlo fashion [76]: if the score improves, it is accepted. Otherwise, it is accepted with some probability. An unfavorable modification is more likely to be accepted if it worsens the score only slightly. Also, the system has a temperature, and unfavorable modifications are more likely to be accepted in higher temperatures. The system cools as the algorithm progresses, making it more conservative (i.e., it rejects more unfavorable steps).

A.1.1 Comparison to other clustering techniques

We compare the quality of the clusters found using SA k-means and other clustering methods: (1) k-means (2) agglomerative hierarchical (3) top-down (random and deterministic flavors). In all cases, we assume that the number of clusters k is given as input. We survey the compared clustering methods here briefly for completeness; a detailed description can be found in reference [25].

$k-$means clustering

Initially, k random elements are selected as cluster centers. The algorithm then iterates between two steps: (1) associate each element with the cluster with the closest center, and (2) re-evaluate the cluster centers based on the elements associated with them in the last phase. Here, the center of each cluster, is the centroid. K-means is guaranteed to converge [4].

Agglomerative hierarchical clustering

The algorithm repeatedly merges the two closest clusters, decreasing the total number of clusters by one. In the initial state, each element is a cluster of its own (n clusters); the algorithm halts when there are k clusters. The inter-cluster distance used to determine the closest cluster pair is either (1) the minimum (single-link clustering), (2) the maximum (complete-link clustering) or (3) the average (average-link) distance between the elements of two clusters. Here, we compare to complete-link clustering.

Top down clustering

The algorithm repeatedly splits clusters into two, increasing the total number of clusters by one. Initially, all n data set points are in one cluster, and the algorithm stops after $k-1$ rounds, with k clusters. The algorithm can be deterministic or random:

1. Deterministic top down always splits the cluster with the maximal internal distance. The cluster points are split to those closer to e_1 versus those closer to e_2, where e_1, e_2 are the two most distant points in the cluster.

2. Random top down selects the cluster for splitting randomly, such that clusters with greater maximal internal distance are more likely to be split. The split consists of clustering the elements of this cluster into two new clusters using k-means algorithm.

A.2 Results

To evaluate SA k-means, we cluster the set of fragments of length 5 into 20 clusters using different clustering methods and compare the results. We consider the methods SA k-means, k-means, Agglomerative hierarchical, and Deterministic top-down (random and deterministic variants). For each of the non-deterministic clustering methods, we initialize the random seed and execute the procedure 50 times. Figure A.2 shows the quality of the clusters found by the different methods. On the x-axis we plot the total variance; this is the score that the procedure is optimizing. On the y axis we plot the average local-fit cRMS when fitting the Park and Levitt [89] test set; this validates externally the quality of the clusters found. Along both axis, better results have lower values. We see that SA k-means best partitions this data set. Also, SA k-means is robust, that is almost all random seeds result in good clusters.

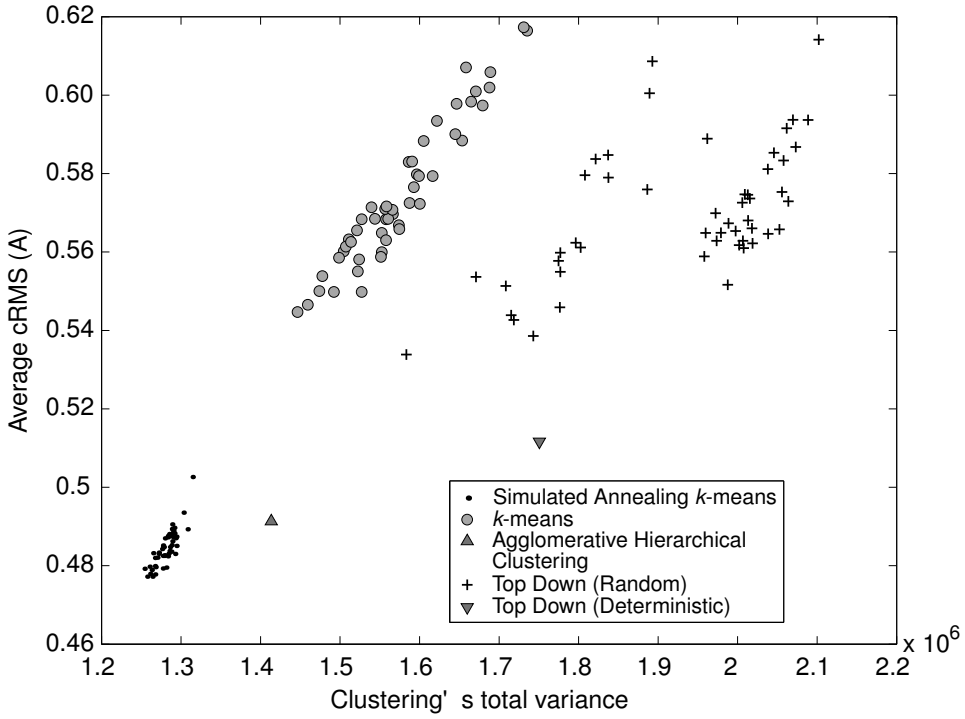

Figure A.2: The performance of clustering methods for clustering the data set of 5 residue fragments into 20 clusters. The value along the x-axis is the total variance, and along the y-axis the average local-fit cRMS on the Park & Levitt set. SA k-means finds the best clusters.

A.3 Discussion

Clustering the data set of fragments is tricky because of dense regions in the data. Specifically, approximately 50% of the fragments originate in α-helical regions. Since helices are well defined, their fragments are almost identical to each other. When the data has dense regions, the initial choice of the cluster centers for k-means clustering is critical. A dense region typically corresponds to a single cluster. However, if several centers are initially selected in this region, k-means will maintain the number of clusters in the that region, optimizing the partitioning of the points in this region into clusters. In this data set a random selection of initial cluster centers will include approximately $k/2$ helical fragments, implying that k-means will find $k/2$ clusters that are the same one. Moreover, the rest of the fragments will be described, less accurately, using only $k/2$ clusters instead of $k - 1$.

Simulated annealing k-means is a heuristic that finds better clusters of the fragments data, by using k-means and optimizing the choice of fragments used for the initial cluster centers. This simple optimization can be modified to use any score. Jane & Dubes [54] and Duda & Hart [25] list many possible scores. Alternatively, the optimization can use an external criterion, e.g. the performance on a test set. Rose [98] did a rigorous analysis of a similar (deterministic annealing) heuristic.

Bibliography

[1] A. Akutsu. Protein structure alignment using dynamic programming and iterative improvement. *IEICE Transactions on Communications/Electronics/Information and Systems*, E78-D(0), 1996.

[2] B. Al-Lazikani, J. Jung, Z. Xiang, and B. Honig. Protein structure prediction. *Curr. Opin. Chem. Bio.*, 5:51–56, 2001.

[3] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, (215):403–410, 1990.

[4] Dempster A.P., Laird N.M., and Rubin D.B. Maximum-likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, B39:1–38, 1977.

[5] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, and H. et al. Weissig. The protein data bank. *Nucl. Acids Res.*, (28):235242, 2000.

[6] C. Brandon and J. Tooze. *Introduction to Protein Structure*. Garland, second edition edition, 1998.

[7] S.E. Brenner, C. Chothia, and T. J. Hubbard. Assessing sequence comparison methods with reliable structurally-identified distant evolutionary relationships. *Proc. Natl. Acad. Sci. USA.*, (95):6073–6078, 1998.

[8] S.E. Brenner, P. Koehl, and M. Levitt. The astral compendium for protein structure and sequence analysis. *Nucleic Acids Research*, (28):254–256, 2000.

[9] W.J. Browne, A.C.T. North, D.C. Phillips, K. Brew, T.C. Vanman, and R.L. Hill. A possible three-dimensional structure of bovine alpha-lactalbumin based on that of hen's egg-white lysozyme. *J. Mol. Biol.*, 42:65–86, 1969.

[10] R.E. Bruccoleri and M. Karplus. Chain closure with bond angle variations. *Macromolecules*, 18:2767–2773, 1985.

[11] A.A. Canutescu and R.L. Dunbrack. Cyclic coordinate descent: A robotics algorithm for protein loop closure. *Protein Sci.*, 12:963 – 972, 2003.

[12] G. Chelvanayagam, G. Roy, and P. Argos. Easy adaptation of protein structure to sequence. *Protein Eng*, (7):173–184, 1994.

[13] I. Choi, J. Kwon, and S.H. Kim. Local feature frequency profile: A method to measure structural similarity in proteins. *Proc. Natl. Acad. Sci. USA.*, (101(11)):3797–3802, 2004.

[14] C. Chothia and A.M. Lesk. The relation between the divergence of sequence and structure in proteins. *EMBO J*, (5):823–826, 1986.

[15] R.B. Corey and L. Pauling. Fundamental dimensions of polypeptide chains. *Proc. R. Soc.*, (141):10, 1953.

[16] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*, chapter 16.2, pages 309–310. MIT Press, 1990.

[17] J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison Wesley, Reading, Massachusetts, 1986.

[18] T.E. Creighton. *Proteins Structures and Molecular Properties*. Freeman, European Molecular Biology Laboratory, Heidelberg, Germany, second edition edition.

[19] M.O. Dayhoff. *Atlas of protein sequence and structure*, volume 5. National Biomedical Research Foundation, 1978.

[20] A.G. de Braven, C. Etchebest, and S. Hazout. Bayesian probabilistic approach for prediction backbone structures in terms of protein blocks. *PROTEINS: Structure, Function, and Genetics*, (41(3)):271–287, 2000.

[21] K. Diederichs. Structural superposition of proteins with unknown alignment and detection of topological similarity using a six-dimensional search algorithm. *Proteins*, 23:187–195, 1995.

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

[23] F. Domingues, P. Lackner, A. Andreeva, and M. Sippl. Structure-based evaluation of sequence comparison and fold recognition alignment accuracy. *J. Mol. Biol.*, (297):1003–1013, 2000.

[24] P.C. Du, M. Andrec, and R.M Levy. Have we seen all structures corresponding to short protein fragments in the protein data bank? an update. *Protein Eng.*, 16:407 – 414, 2003.

[25] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley and Sons, New York, NY, U.S.A., 1973.

[26] I. Eidhammer, I. Jonassen, and W.R. Taylor. Structure comparison and structure patterns. *J. Comp. Biol.*, 7:685–716, 2000.

[27] B. Fain and M. Levitt. A novel method for sampling alpha-helical protein backbones. *J. Mol. Biol.*, (305):191–201, 2001.

[28] M. Farach, Savari S. Noordewier, M., L. Shepp, A. Wyner, and J. Ziv. On the entropy of dna: algorithms and measurements based on memory and rapid convergence. *Proceedings of SODA*, 1995.

[29] K. Fidelis, P.S. Stern, D. Bacon, and J. Moult. Comparison of systematic search and database methods for constructing segments of protein-structure. *Protein Eng.*, 7:953 – 960, 1994.

[30] Do R.K.G. Fiser, A. and A. Sali.

[31] K.W. Foreman, A.T. Phillips, J.B. Rosen, and K.A. Dill. Comparing search strategies for finding global optima on energy landscapes. *J. Comp. Chem.*, (20(14)):1527–1532, 1999.

[32] I. Friedberg, T. Kaplan, and H. Margalit. Evaluation of psi-blast alignment accuracy in comparison to structural alignments. *Protein Sci.*, (9(11)):2278–84, 2000.

[33] D. Frishman and Argos P. Knowledge-based protein secondary structure assignment. *Proteins: Struct. Funct. Genet.*, (23):566–579, 1995.

[34] M. Gerstein and M. Levitt. Using iterative dynamic programming to obtain accurate pairwise and multiple alignments of protein structures. *Proceedings of the Fourth International Conference on Intelligent Systems in Molecular Biology*, pages 59–67, 1996.

[35] M. Gerstein and M. Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins. *Protein Sci.*, (7):445–456, 1998.

[36] J.F. Gibrat, T. Madej, and S.H. Bryant. Surprising similarities in structure comparison. *Curr Opin Struct Biol*, 6:377–385, 1996.

[37] K. Ginalski, A. Elofsson, D. Fischer, and L. Rychlewski. 3d-jury: a simple approach to improve protein structure predictions. *Bioinformatics*, (19(8)):1015–8, 2003.

[38] N. Go and H.A. Scheraga. Ring closure and local conformational deformation of chain molecules. *Macromolecules*, 3:178 – 186, 1970.

[39] A. Godzik. The structural alignment between two proteins: Is there a unique answer? *Protein Science*, (5):1325–1338, 1996.

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

[41] M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (roc) analysis to evaluate sequence matching. *Comput. Chem.*, (20):25–33, 1996.

[42] D. Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology.* Cambridge University Press, New York, NY, 1997.

[43] M.H. Hao, S. Rackovsky, A. Liwo, M.R. Pincus, and H.A. Scheraga. Effects of compact volume and chain stiffness on the conformations of native proteins. *Proc. Natl. Acad. Sci. (USA)*, 89:6614–6618, 1992.

[44] A. Harrison, F. Pearl, R. Mott, J. Thornton, and C. Orengo. Quantifying the similarities within fold space. *J. Mol. Biol.*, (323):909–926, 2002.

[45] T.F. Havel, I.D. Kuntz, and G.M. Crippen. The theory and practice of distance geometry. *Bull Math Biol*, (45):665–720, 1983.

[46] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA.*, (89):100915–100919, 1992.

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

[48] L. Holm and Sander C. Protein structure comparison by alignment of distance matrices. *J. Mol. Biol.*, 233:123–138, 1993.

[49] L. Holm and J. Park. Dalilite workbench for protein structure comparison. *Bionformatics*, (16):566–567, 2000.

[50] L. Holm and C. Sander. The fssp database of structurally aligned protein fold families. *Nucleic Acids Res*, (22):3600–3609, 1994.

[51] L. Holm and C. Sander. Mapping the Protein Universe. *Science*, 273(5275):595–602, 1996.

[52] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4, 1987.

[53] E.S. Huang, R. Samudrala, and J.W. Ponder. Ab initio fold prediction of small helical proteins using distance geometry and knowledge-based scoring functions. *J. Mol. Biol.*, (290):267–281, 1999.

[54] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[55] A.T. Jones and S. Thirup. Using known substructures in protein model building and crystallography. *EMBO J.*, (5):819–822, 1986.

[56] D.T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.*, (292(2)):195–202, 1999.

[57] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallog.sect.*, A(34):827–828, 1978.

[58] M. Karplus and G.A. Petsko. Molecular dynamics simulations in biology. *Nature*, 347(6294):631–9, 1990.

[59] K. Kedem, L.P. Chew, and R. Elber. Unit-vector rms (urms) as a tool to analyze molecular dynamics trajectories. *Proteins: Struct. Funct. Genet.*, (37):554–564, 1999.

[60] D. Kihara and J. Skolnick. The pdb is a covering set of small protein structures. *J. Mol. Biol.*, (334):793–802, 2003.

[61] G.J. Kleywegt. Use of non-crystallographic symmetry in protein structure refinement. *Acta Cryst. D.*, (52):842–857, 1996.

[62] G.J. Kleywegt and A. Jones. Superposition. *CCP4/ESF-EACBM Newsletter on Protein Crystallography*, (31):9–14, 1994.

[63] P. Koehl. Protein structure similarities. *Curr. Opin. Struct. Biol.*, (11):348–353, 2001.

[64] R. Kolodny, P. Koehl, L. Guibas, and M. Levitt. Small libraries of protein fragments model native protein structures accurately. *Journal of Molecular Biology*, 323:297–307, 2002.

[65] R. Kolodny, P. Koehl, L. Guibas, and M. Levitt. Inverse kinematics in biology: the protein loop closure problem. *International Journal of Robotics Research*, 2004.

[66] R. Kolodny and M. Levitt. Protein decoy assembly using short fragments under geometric constraints. *Biopolymers*, 68:278–285, 2003.

[67] R. Kolodny and N. Linial. Approximate protein structural alignment in polynomial time. *Proc. Natl. Acad. Sci. (USA)*, 101:12201 – 12206, 2004.

[68] J. Kraulis. Molscript: A program to produce both detailed and schematic plots of protein structures. *Journal of Applied Crystallography*, (24):946–950, 1991.

[69] E. Krissinel and K. Henrick. Protein structure comparison in 3d based on secondary structure matching (ssm) followed by c-alpha alignment, scored by a new structural similarity function. In A.J. Kungl and P.J. Kungl, editors, *Proceedings of the 5th international conference on Molecular Structural Biology*, page 88, Vienna, September 2003.

[70] C. Lemmen and T. Lengauer. Computational methods for the structural alignment of molecules. *Jour Computer Aided Molecular Design*, (14):215–232, 2000.

[71] R. Leplae and T. J. P. Hubbard. Maxbench: evaluation of sequence and structure comparison methods. *Bioinformatics*, (18):494–495, 2002.

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

[73] T. Madej, J.F. Gibrat, and S.H. Bryant. Threading a database of protein cores. *Proteins*, (23):356–369, 1995.

[74] Jiri Matousek. *Lectures on Discrete Geometry*. Springer-Verlag, New York, 2002.

[75] A.C. May and M.S. Johnson. Improved genetic algorithm based protein structure comparisons: pairwise and multiple superpositions. *Protein Eng*, 8:873–882, 1995.

[76] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6), 1953.

[77] C. Micheletti, F. Seno, and A. Maritan. Best representative oligomers (oligons). online http://www.sissa.it/ michelet/prot/repset/index.html, 2000.

[78] C. Micheletti, F. Seno, and A. Maritan. Recurrent oligomers in proteins: an optimal scheme reconciling accurate and concise backbone representations in automated folding and design studies. *PROTEINS: Structure, Function, and Genetics*, (40):662–674, 2000.

[79] K. Mizuguchi and N. Go. Comparison of spatial arrangements of secondary structural elements in proteins. *Protein Eng*, (8):353–362, 1995.

[80] J. Moult and M.N.G. James. An algorithm which predicts the conformation of short lengths of chain in proteins. *J. Mol. Graph.*, 4:180 – 180, 1986.

[81] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, (247):536–540, 1995.

[82] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, (48):443–453, 1970.

[83] M. Novotny, D. Madsen, and G. J. Kleywegt. Evaluation of protein-fold-comparison servers. *Proteins*, (54):260–270, 2004.

[84] R. Nussinov and H.J. Wolfson. Efficient detection of three-dimensional structural motifs in biological macromolecules by computer vision techniques. *Proc. Natl. Acad. Sci. USA*, (15):287–298, 1989.

[85] B. Oliva and P.A. Bates. An automated classification of the structure of protein loops. *J. Mol. Biol.*, (266):814–830, 1997.

[86] C. Orengo. Classification of protein folds. *Curr Opin Struct Biol*, (4):429–440, 1994.

[87] C.A. Orengo, A.D. Michie, S. Jones, D.T. Jones, M.B. Swindells, and J.M. Thornton. Cath- a hierarchic classification of protein domain structures. *Structure*, (5):1093–1108, 1997.

[88] C. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

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

[90] W. Pearson and D. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA.*, (85):2444–2448, 1988.

[91] J.T. Pederson and J. Moult. Protein folding simulations with genetic algorithms and a detailed molecular description. *J. Mol. Biol.*, (269):240–259, 1997.

[92] M.F. Perutz, M.G. Rossmann, A.F. Cullis, H. Muirhead, G. Will, and A.C.T. North. Structure of myoglobin: A three-dimensional fourier synthesis at 5.5 angstrom resolution, obtained by x-ray analysis. *Nature*, (185):416–422, 1960.

[93] G.N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Conformation of polypeptides and proteins. *J. Mol. Biol.*, (7):95–99, 1963.

[94] G. Reinert, S. Schbath, and M. Waterman. Probabilistic and statistical properties of words. *J. Comp. Biol.*, 7:1–46, 2000.

[95] B. Reva, A.V. Finkelstein, and J. Skolnick. What is the probability of a chance prediction of a protein structure with an rmsd of 6 a? *Folding & Design*, (3):141–147, 1998.

[96] P. Rogen and B. Fain. Automatic classification of protein structure by using gauss integrals. *Proc. Natl. Acad. Sci. USA.*, (100(1)):119–24, 2003.

[97] M.J. Rooman, J.I. Kocher, and S.J. Wodak. Extracting information on folding from the amino acid sequence: accurate predictions for protein regions with preferred conformation in the absence of tertiary interactions. *Biochemistry*, (31):10226–10238, 1992.

[98] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 80:2210–2239, 1998.

[99] M.N Rosenbluth and A.W. Rosenbluth. Monte-carlo calculation of the average extension of molecular chains. *J. Chem. Phys.*, (23):356–359, 1955.

[100] E.B. Saff and A.B.J. Kuijlaars. Distributing many points on a sphere. *Mathematical Intelligencer*, 19:5–11, 1997.

[101] J.M. Sauder, J.W. Arthur, and R.L. Dunbrack. Large scale comparison of protein sequence alignment algorithms with structure alignments. *Proteins: Struct. Funct. Genet*, (40):6–22, 2000.

[102] A.O. Schmitt and H. Herzel. Estimating the entropy of dna sequences. *J. Theor. Biol.*, 188(3):369–77, 1997.

[103] J. Shapiro and D. Brutlag. Foldminer: structural motif discovery using an improved superposition algorithm. *Protein Sci.*, (13):278–94, 2004.

[104] I.N. Shindyalov and P.E. Bourne. Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. *Protein Eng*, 11:739–747, 1998.

[105] I.N. Shindyalov and P.E. Bourne. An alternative view of protein fold space. *Proteins: Struct. Funct. Genet.*, (38):247–260, 2000.

[106] K. Shoemaker. Animating rotation with quaternion curves. *Computer Graphics*, 19:245–254, 1985.

[107] M.L. Sierk and W.R. Pearson. Sensitivity and selectivity in protein structure comparison. *Protein Sci.*, (13):773–785, 2004.

[108] I. Simon, L. Glasser, and H.A. Scheraga. Calculation of protein conformation as an assembly of stale overlapping segments: Application to bovine pancreatic typsin inhibitor. *Proc. Natl. Acad. Sci. USA*, (88):3661–3665, 1991.

[109] K.T. Simons, R. Bonneau, I. Ruczinski, and D. Baker. Ab initio protein structure prediction of casp iii targets using rosetta. *Proteins: structure function and genetics*, (S3):171–176, 1999.

[110] K.T. Simons, C. Kooperberg, E. Huang, and D. Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *J. Mol. Biol.*, (268):209–255, 1997.

[111] A. Stark, S. Sunyaev, and R. B. Russel. A model for statistical significance of local similarities in structure. *J. Mol. Biol.*, (326):1307–1316, 2003.

[112] S. Subbiah, D. V. Laurents, and M. Levitt. Structural similarity of dna-binding domains of bacteriophage repressors and the globin core. *Current Biol.*, (3):141–148, 1993.

[113] S. Sudarsanam, R.F. Dubose, C.J. March, and S. Srinivasan. Modeling protein loops using a ϕ-i+1, ψ-i dimer database. *Protein Sci.*, 4:1412 – 1420, 1995.

[114] N.L. Summers and M. Karplus. Modeling of globular-proteins - a distance-based data search procedure for the construction of insertion deletion regions and pro reversible non-pro mutations. *J. Mol. Biol.*, 216:991–1016, 1990.

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

[116] J.D. Szustakowski and Weng Z. Protein structure alignment using a genetic algorithm. *Proteins: Struct. Funct. Genet.*, 38:428–440, 2000.

[117] W.R. Taylor and C.A. Orengo. Protein structure alignment. *J. Mol. Biol.*, 208:1–22, 1989.

[118] W.R. Taylor and C.A. Orengo. Protein structure alignment. *J. Mol. Biol.*, (208):1–22, 1989.

[119] J.D. Thompson, F. Plewniak, and O. Poch. Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, (15):87–88, 1999.

[120] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[121] R. Unger, D. Harel, S. Wherland, and J.L. Sussman. A 3d building blocks approach to analyzing and predicting structure of proteins. *PROTEINS: Structure, Function, and Genetics*.

[122] H.W.T. vanVlijmen and M. Karplus. Pdb-based protein loop prediction: Parameters for selection and methods for optimization. *J. Mol. Biol.*, 267:975 – 1001, 1997.

[123] M. Vasquez and H.A. Scheraga. Use of buildup and energy-minimization procedures to compute low energy structures of the backbone of enkaphilin. *Biopolymers*, (24):1437–1447, 1985.

[124] M. Vazquez and H. A. Scheraga. Calculation of protein conformation by the build-up procedure. application to bovine pancreatic trypsin inhibitor using limited simulated nuclear magnetic resonance data. *J. Biomol. Struct. Dynam.*, (5):705–755, 1988.

[125] G. Vriend and C. Sander. Detection of common three-dimensional substructures in proteins. *Proteins*, 11:52–58, 1991.

[126] W.J. Wedemeyer and H.A. Scheraga. Exact analytical loop closure in proteins using polynomial equations. *J. Comp. Chem.*, 20:819 – 844, 1999.

[127] R.T. Wintjens, M.J. Rooman, and S.J. Wodak. Automatic classification and analysis of alpha-alpha-turn motifs in proteins. *J. Mol. Biol.*, (255(1)):235–253, 1996.

[128] T.D. Wu, S.C. Schmidler, T. Hastie, and D.L Brutlag. Regression analysis of multiple protein structures. *J Comput Biol*, (5):585–595, 1998.

[129] A.S. Yang and B. Honig. An integrated approach to the analysis and modeling of protein sequences and structures i. protein structure alignment and quantitative measure for protein structural distance. *J. Mol. Biol.*, (301):665–678, 2000.

[130] D.P. Yee and K.A. Dill. Families and the structural relatedness among globular proteins. *Protein Sci*, (2):884–899, 1993.

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

[132] F. Zu-Kang and M.J. Sippl. Optimum superimposition of protein structures: ambiguities and implications. *Folding & Design*, (1):123–132, 1996.