UNIVERSITY of CALIFORNIA
SANTA CRUZ

PROTEIN SEQUENCE ALIGNMENT RELIABILITY: PREDICTION AND MEASUREMENT

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Melissa Suzanne Cline

June 2000

The dissertation of Melissa Suzanne Cline
is approved:

Associate Professor Kevin Karplus, Chair

Professor David Haussler

Assistant Professor Lydia Gregoret

Dean of Graduate Studies

Contents

List of Figures

List of Tables

Abstract

Protein Sequence Alignment Reliability: Prediction and Measurement

by

Melissa Suzanne Cline

Protein sequence alignments are one of the most fundamental sources of information in bioinformatics, with applications in structure prediction, homology modeling, prediction of functional residues, phylogenic analysis, secondary structure prediction, and gene prediction. In blind prediction experiments involving the best methods in the world, Hidden Markov Models (HMMs) have been shown to be very effective alignment methods. However, all alignment methods – including HMMs – make mistakes. Since mistakes in alignments yield mistakes in their application, prediction and removal of suspect alignment regions would improve the reliability of the many applications of alignments.

Useful prediction of reliable alignment regions requires optimized alignments, an objective function, and meaningful inputs. We have identified numerous factors that yield better alignments with HMMs, optimized our methodology, and benchmarked against the respected aligner CLUSTALW. We have developed an alignment quality measure that produces a score that is meaningful, comprehensive, and optimizable. Pairwise contact potentials, a popular source of information not used by HMMs, would seem to be useful indicators of HMM alignment reliability. However, our statistical analysis of pairwise contact potentials found them significant yet weak. Even β-strand contacts, the class we found most informative, did not yield enough information to merit application. Finally, we developed a system that examines alignments predicted by HMMs and estimates the reliability of each alignment column. When we trim our best alignments according to this reliability estimate, we remove 70% of the positions more than slightly misaligned and 58% of the over-aligned positions, while retaining 86% of those aligned accurately.

To Barbara Molyneux, my high school mathematics teacher, who worked hard to make a diligent student out of me even when I gave her little cause for hope.

Acknowledgements

This work was not performed in a vacuum. This thesis builds on the hard work put in over the years by the UCSC computational biology group. This group has had a number of very talented people during my time, including Christian Barrett, Nguyet Manh, Rachel Karchin, Spencer Tu, Leslie Grate, Michael Brown, David Kulp, Chuck Sugnet, and Terry Furey. I'd work with any one of them again in a heartbeat. Special thanks go to a few individuals. Albion Baucom and I worked together on the experiments described in Chapter 8. Albion brought to the project key observations, a lot of enthusiasm, and some neural network source code that is also used in Chapter 9. Richard Hughey has been invaluable for friendship, help with SAM, and the occasional dose of pure common sense. I could not ask for a better "big sister" than Kimmen Sjölander, who also lured me into the group years ago. To this day, she claims that no finder's fee was involved, but this author remains skeptical. Mark Diekhans has generously provided me with three ingredients vital to any research effort: useful code, moral support, and coffee.

All the my thesis committee has been important over the years in this work. Over the years, I've had not one advisor but three. All members of my thesis committee have enriched this work with distinct knowledge and expertise, and have played a major role in shaping me as a researcher. I've been very lucky to have three excellent mentors during the course of my Ph.D. work; many people never have one.

While the university and the federal financial aid administration are both vital institutions to graduate study and research, they're both driven by millions of picky details. Carol Mullane and Henry Rutland have protected me from a lot of those details so I could concentrate on my research instead of my administrative forms. I know ways in which Carol and Henry have both helped me, and I have no doubt they've both helped me with many other things I don't know about.

Everyone needs a cheerleader at times, and my friends and family have been wonderful. Special thanks go to my parents and my sister, Vicki Cline. Old friends deserve better than to be badgered into thesis-reading, but Cricket Grice bore the mistreatment bravely! Jorge García deserves thanks for many reasons: for keeping my computer running, for gamely attempting to cook dinner while I've been preoccupied, and for **volunteering** to review my manuscript.

Friends in the wine-making industry claim that it takes a lot of good beer to make good wine. Similar claims can be made for thesis-writing. I wish to thank the management of the Seabright Brewery and the Jahva House for years of good beer, good coffee, and resulting improvements in the quality of life.

The final revisions of this thesis could not have been completed in a timely manner without the aid of Source Naturals Wellness Formula.

Chapter 1

Introduction

In the words of protein structure prediction expert David Jones, “As the familiar joke goes, there are really only three things that govern the overall accuracy of comparative modeling: alignment quality, alignment quality, and... alignment quality” [85]. Protein sequence alignments are one of the most important sources of information in bioinformatics, with applications in tertiary structure prediction, homology modeling, secondary structure prediction, predition of functional residues, phylogenic analysis, and gene prediction. Any mistake made in the alignment step will yield mistakes in its application. Yet alignment quality has not been studied thoroughly, owing in part to historical difficulties in measuring overall alignment quality.

This dissertation focuses on alignments produced by hidden Markov models (HMMs). Given a sequence family alignment, typically consisting of a sequence of known structure plus its *homologs*, or evolutionary relatives, HMMs profile the patterns of conservation and variation that characterize the sequences in the family. Given a new sequence, and given a library of sequence family alignments, HMMs predict which family the new sequence is most closely related to, and produce an alignment of the sequence to this family. In blind experiments of the best methods currently available, HMMs have proven successful at protein structure prediction [128, 87, 109].

In this thesis, we introduce and validate a comprehensive measure of overall alignment quality, which we apply in detailing improvements to HMM alignment methods. We quantify the impact of various factors that influence the accuracy of HMM-generated alignments, and benchmark the best methods against the respected aligner CLUSTALW [180, 65, 82]. We then consider factors which might indicate reliable regions within an alignment. Pairwise contact potentials, which validate predicted substructures within threading algorithms, would appear to be useful for identifying suspect regions in alignments produced by a sequence-based method. However, we show that the information available through contact potentials is weak, and would probably not be useful for identifying questionable alignment regions. Yet there is information already encoded within an alignment that is useful for identifying such regions. We describe a method that predicts if a region within an alignment is reliable. When regions predicted as unreliable are removed, overall alignment quality is increased by 15%, where the greatest possible improvement is 50%.

1.1 Motivation

Predicting a protein's structure from its amino acid sequence is a problem of great scientific importance. As the major genome projects progress, and as protein sequencing methods continue to out-pace structure determination methods, it has become a problem of great practical importance. While this problem is not yet solved, we have a partial solution. If two protein sequences are at least 30% identical, they tend to have similar structures [160, 154]. Sensitive methods such as hidden Markov models (HMMs) can successfully predict strutural similarity can when the sequence similarity is less-pronounced [93]. The power of this partial solution should not be underestimated; a researcher is approximately 20 times more likely to have a novel sequence “under the microscope” than a sequence of novel structure [84].

The first and most fundamental step in analysis of a new protein sequence (*target sequence*) is to search the protein databases for homologs of the sequence and to align the sequence to these homologs [174]. If some homolog of known structure is found, then the structure of the target sequence can be predicted according to its alignment to this *template sequence*. Next, the target sequence is added to the multiple alignment of the template and its homologs. Such alignments have been used to predict functional residues [134], to locate genes in newly-sequenced genomes [102], to predict the secondary structure of the target sequence [152], and to detail the evolutionary relationships between the target sequence and homologs [167]. As all these analyses build off the alignment, they are dependent on alignment accuracy. Mistakes in the alignment are known to lead to mistakes in the secondary structure prediction [137] and suspicious predictions of evolutionary relationships [45]. If the goal of target sequence analysis is structure-based drug design, the next step is to build a *homology model*, a three-dimensional model of the structure of the target sequence. The homology model is computed from the template sequence structure and the alignment of the target and template sequences and its accuracy depends strongly on the accuracy of the alignment. Alignment errors are the most frequent and serious source of errors in homology modeling [173]. Sequence analysis begins with alignment, and success in all later steps depends on success in the alignment step.

Given the importance of the alignment step, care should be taken in selecting the alignment algorithm and its parameters. Factors that stand to improve the quality of HMM-generated alignments include the size and variation of the sequences used to generate the HMM, methods for aligning these input sequences, and weighting methods to factor out redundancy.

Unfortunately, no alignment method is free of mistakes. When the sequences to align are somewhat diverged, even alignments generated by the best methods tend to contain inaccurate regions [127]. While a partially correct alignment can yield useful information, a scientist using an alignment would like to know which regions can be trusted and which should be removed.

However, to what extent does removal of suspect regions improve an alignment? At the extreme, one could remove everything except regions of very strong conservation. The result would be that the portions of the target sequence still aligned are probably aligned correctly, but with so much of the target sequence not aligned, the alignment would be of little value. Before one can develop an effective system to remove questionable alignment regions, one must have a scoring system to indicate when the removal has yielded an overall improvement.

Once the means are in place to measure alignment quality improvement, factors to indicate questionable alignment regions must be identified. One traditional rule of thumb state that alignments are less likely to be accurate in long loop regions. Another states that protein families tend to have regions of strong conservation and regions of greater variation, and one should question the alignment in the less-conserved regions [35] or where the target sequence aligns with decreased probability [165]. A body of work in *sub-optimal alignments* suggests studying many different alignments of the same sequences; certain subsequences will tend to be aligned to the same subsequences in most cases, and those alignment regions should be trusted [188].

Many successful threading algorithms are based on the principle that certain types of amino acids are frequently found in proximity to certain other types of amino acids, and the quality of a predicted structure can be assessed in part by observing the types of amino acids found close to each other in the predicted structure, and assessing the statistical likelihood of these pairings [184, 126]. Particularly for alignments produced with no structural information, these *pairwise contact potentials* would seem to be useful for alignment validation. Yet the effectiveness of pairwise contact potentials has been debated [179], so they should be studied further before application.

1.2 Thesis

Almost any pairwise sequence alignment will contain regions of greater and lesser reliability. The reliability of a region can be estimated by studying the alignment. By using these signals, one can obtain a subalignment containing the reliable regions and omitting many of the suspect regions.

1.3 Contributions and organization

This dissertation presents the following contributions. Chapter 4 presents and validates an alignment scoring scheme which can be used to measure effectively improvements in alignment quality when suspect regions are removed. This measure is then used as a tool for improving alignments generated by SAM-T98 [93], an optimized hidden Markov model methodology.

When the ultimate goal is generating the most accurate alignment possible, little is to be gained by starting with alignments produced by anything but the best methods available. As such, in Chapter 5, we document contributions to the task of optimizing HMM alignments and identifying factors which yield better alignments. Specific factors explored include sequence weighting schemes [89], the choice between global and local alignment [9], and the effects of posterior decoding [70]. Where certain alignment errors are suspected to result from shortcomings in certain parameters, we document efforts in optimizing those parameters. Of particular interest here are fields in the *transition regularizers*, the HMM equivalent of *gap costs*, which define in part where an HMM will open or extend gaps [52].

Any thorough comparison of alignment methods should include some external and recognized method as a benchmark. In Chapter 6, we study the literature on sequence alignment methods, and compare the quality of the SAM alignments from Chapter 5 with those from the popular alignment program CLUSTALW [180, 65, 82]. We generate this comparison carefully, by comparing the alignments to structural alignments by three different structural aligners and by measuring the comparison over an impartial validation set.

A particular factor which could suggest alignment reliability is pairwise contact potentials, statistical propensities indicating if two amino acids would be stable in proximity to each other. Such potentials have a long history in validating predicted substructures in threading algorithms. When a source of information is not used to build alignments, it can be effective as an objective quality assurance measure. Thus, pairwise contact potentials should be effective for validating alignments built without structural information. However, as the information represented by pairwise contact potentials is a subject of controversy, we performed a statistical analysis on how much information they carry, and found that the information they carry is too weak to suggest that they would be effective. This analysis is summarized in Chapter 7. Yet one particular class of contacts appears to be more informative: contacts between interacting β-strand residues. Chapter 8 summarizes explorations by myself, Albion Baucom, and Lydia Gregoret in building a beta sheet contact potential function.

Chapter 9 describes a method to improve predicted alignments by identifying and removing suspect regions. Starting with alignments built with the best HMM methodology we have today, we estimate the reliability of alignment columns, remove any columns which are sufficiently suspect, and measure the resulting change in alignment accuracy. With a carefully-validated experimental process, we estimate prediction models on one dataset, search for effective prediction thresholds on a second dataset, and measure overall effectiveness of the method using a third dataset. We measure change in alignment quality by comparing the alignments before and after trimming to structural alignments. To reduce the chance that the results might somehow be dependent on one structural alignment method, we measure performance relative to three structural aligners. Our best methods yield a 4% improvement in the quality of alignments built with our best method, and a 15% improvement on alignments built by a more commonly-used method. In this latter case, the alignment trimming removes 70% of the positions more than slightly misaligned and 58% of the *over-aligned positions*, positions aligned in regions where the sequences have no actual structural similarity, while retaining close to 90% of the accurate positions – on pairs of structures representing a very challenging alignment test.

Chapter 2

Biological background

Bioinformatics is a field that bridges many disciplines including biology, chemistry, physics, mathematics, and computer science. The important background material is similarly broad, and much of it would not be known by an average computer scientist. This chapter is not a comprehensive introduction to molecular biology. Instead, this chapter was written to provide the average computer scientist with enough biochemical background to read and assess this thesis.

To augment this background chapter, Appendix A contains a glossary of biological terms relevant to this thesis.

2.1 What is a protein?

Proteins are basic building blocks of life. Each protein is a macromolecule built by linking together amino acids into contiguous chains. Proteins are produced in the cell according to a pattern specified by the organism's DNA, with RNA serving as an intermediate specification.

Amino acids are the subunits of proteins. There are twenty different types of amino acids found in proteins. All consist of a central carbon (the *alpha carbon* or C_α) bonded to a hydrogen, an amino group, a carboxyl group, and a side chain as shown in Figure 2.1. The *side chain* refers

Amino Acid	Three-letter code	One-letter code
Alanine	Ala	A
Arginine	Arg	R
Asparagine	Asn	N
Aspartic acid	Asp	D
Cysteine	Cys	C
Glutamic acid	Glu	E
Glutamine	Gln	Q
Glycine	Gly	G
Histidine	His	H
Isoleucine	Ile	I
Leucine	Leu	L
Lysine	Lys	K
Methionine	Met	M
Phenylalanine	Phe	F
Proline	Pro	P
Serine	Ser	S
Threonine	Thr	T
Tryptophan	Trp	W
Tyrosine	Tyr	Y
Valine	Val	V

Table 2.1: The twenty types of amino acids found in proteins

to a set of atoms connected to the alpha carbon.. Each type of amino acid has a different side chain, and the side chain gives each amino acid its distinguishing characteristics. Most amino acid side chains consist of a *beta carbon* or C_β connected to the alpha carbon, with additional atoms connected to the beta carbon. The twenty amino acids found in proteins are listed in Table 2.1, along with the three-letter and one-letter codes often used in place of their full names.

When two or more amino acids combine to form a protein, or more technically a polypeptide chain, a condensation reaction takes place. The net result is the production of a water molecule and a connection between the carbonyl carbon of the preceding amino acid and the amino nitrogen of the following amino acid, forming a *peptide bond.* The process is then repeated to extend the chain. A short polypeptide is illustrated in Figure 2.2. The start of a protein is referred to as the *N terminus*, and the end as the *C terminus.* The series of N, C_α, and C' atoms is referred to as the *backbone.* Once amino acids have joined to form a protein, they are technically no longer amino acids in the chemical sense, and are usually referred to as *residues.* The length of a protein

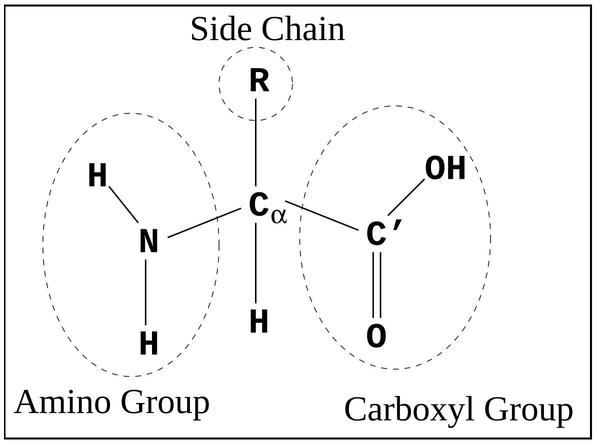


Figure 2.1: Diagram of an amino acid

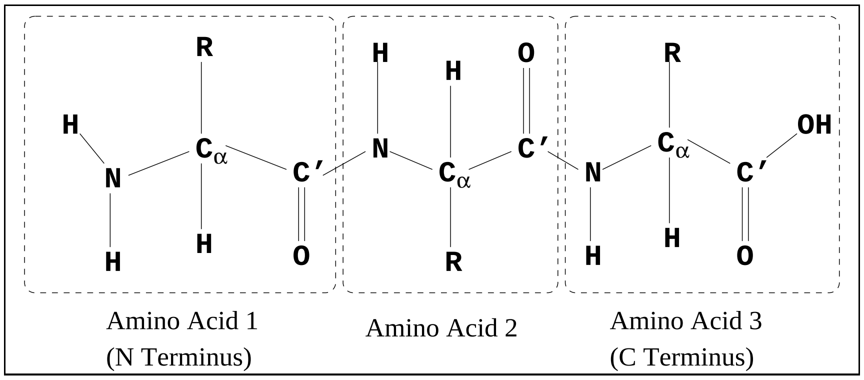


Figure 2.2: A simple polypeptide chain of three amino acids

is measured in residues, with lengths varying from approximately 50 to 2000 residues.

2.2 Protein structure

Proteins in nature assemble themselves into neat, compact structures. The structure of a protein is somehow specified by its amino acid sequence, in a manner only partly understood.

2.2.1 Classes of protein structure

The structure of a protein is defined at four levels: *primary, secondary, tertiary*, and *quaternary* structure.

Primary structure

The primary structure of a protein refers to its amino acid sequence. The one-letter code is normally used when describing the amino acid sequence, yielding a string built from an alphabet of twenty letters. It is always represented in a standard direction, from the amino (N) terminus to the carboxyl (C) terminus, as shown in Figure 2.2.

Secondary structure

In proteins, regions of amino acids within the protein chain will tend to arrange themselves into regular formations. These patterns are referred to as *secondary structure*, and can be recognized by the angles and hydrogen bond patterns between the backbone atoms. Secondary structure is usually divided into three classes: *α-helix, β-sheet,* and *loop.*

In an α-helix, the amino acids arrange into a tight spiral, making a complete turn every 3.6 residues. In a β-sheet structure, the amino acids assemble into *β-strands:* short, fairly flat segments. These β-strands form a β-sheet, a somewhat flat and even surface, by establishing regular patterns of hydrogen bonds between amino acids in adjacent β-strands. β-sheets can be either *parallel* or *antiparallel.* In antiparallel β-sheets, the protein chain doubles back on itself, often making a short turn called a *β-turn*, and forms an alternating pattern. If one thinks of a protein chain as pointing from N to C terminus, adjacent β-strands point in opposite directions. In parallel β-sheets, the β-strands all point in the same direction relative to the N and C termini. Figure 2.3 shows an example of an antiparallel β-sheet and an α-helix.

Loop, also known as *random coil*, is the *all other* category of secondary structure. In general, loops are not structured in the way that α-helices or β-sheets are; they are the portion of the protein that resembles "cooked spaghetti". They can be flexible or rigid, and usually serve as connectors between α-helices and β-strands.

Sometimes, the term "secondary structure" is used to refer to the portions of the protein for which the secondary structure is structured: α-helix and β-sheet. In particular, the term *secondary structural elements* refers to the non-loop regions of the protein.

Tertiary structure

Tertiary structure or *fold* refers to the three-dimensional structure of the protein, containing secondary structural elements as sub-units. Protein structures are generally compact and orderly. Proteins are said to *fold* into their tertiary structures, a verb which suggests order in the final structures, whether or not the process is orderly. The word *fold* is also used to refer to a protein's structural family: proteins with the same fold are proteins with similar structures.

Protein structures occasionally contain compact modules called *domains*. A domain can be thought of as an independently-folding section of the protein.

Quaternary structure

Certain protein chains form stable structures in isolation, and others exist only in a three-dimensional complex with other protein chains. The *quaternary structure* refers to the assembling of individual protein substructures into three-dimensional structures.

2.2.2 Methods of determining protein structure

Three-dimensional protein structures are determined by two methods: *x-ray crystallography* and *NMR spectroscopy*. Protein structures can be determined to a fine detail, describing the relative position of every single atom within the protein.

X-ray crystallography is the predominant method of protein structure determination. The process begins by growing crystals of a purified protein sample. Once the crystals have grown sufficiently large, X-ray beams are applied to the crystal, and the structure is determined by studying the diffraction pattern. While this process might sound simple in a brief summary, it is not. Growing protein crystals requires skill, can easily take months, and is not feasible for all proteins. Obtaining a structure from the diffraction pattern amounts to finding a model of the

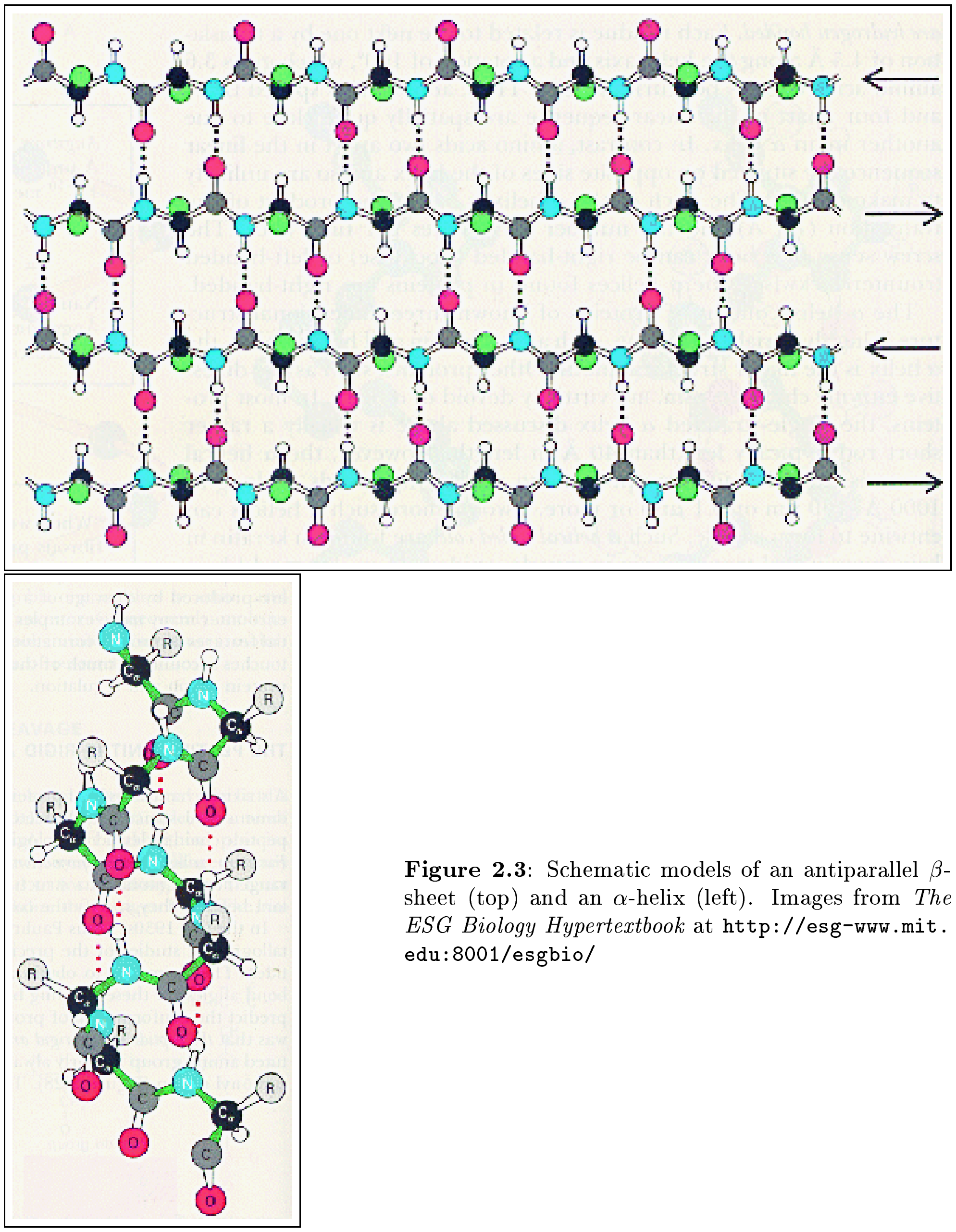


Figure 2.3: Schematic models of an antiparallel β-sheet (top) and an α-helix (left). Images from *The ESG Biology Hypertextbook* at `http://esg-www.mit.edu:8001/esgbio/`

structure which best explains the data, and is as complex a process as any method of protein structure prediction. A recent innovation known as *molecular replacement* uses the structure of a protein with a known structure and a similar sequence to reduce the number of free variables in the modeling process. While this procedure is only viable when such a related protein exists, the high rates growth of the protein structure databanks suggest that this technique will become more common over time.

Nuclear Magnetic Resonance (NMR) spectroscopy does not require protein crystals, but “merely” a highly concentrated and purified sample of the protein in question, at a slightly lowered pH. The protein is then put in a strong magnetic field, and subjected to radio frequency (RF) pulses. This puts the nuclei of certain atoms of the protein in an excited state, and as they return to equilibrium, they emit RF radiation. Structural information can then be inferred from the frequencies and intensities of the emitted radiation and from coupling between the frequencies of individual nuclei. Like crystallography, determining the structure from the observed data is a complex modeling process itself, and the technique is not viable on all proteins. Certain proteins are not stable in concentrated solutions at lowered pH. In general, NMR spectroscopy is not viable on larger proteins due to technical limitations.

Both structure determination methods require months or years of work, and are not viable for certain proteins. For this reason, predicted structures are often used as a substitute for determined structures, especially for time-sensitive work such as structure-based drug design.

The Protein Data Bank (PDB) is the world’s central repository of determined protein structures [13]. As of February 22, 2000, the PDB contains structural data for approximately 10,000 proteins. While this might sound like an impressive amount of data, the 10,000 structures represent less than 3,000 significantly-different structures.

2.2.3 Characteristics of protein structures

The first salient point about the structure of a protein is that it is somehow specified by the amino acid sequence. This has been shown in studies where proteins are *denatured*, or forced to lose their structure by application of strong chemical or physical forces such as high temperature or low pH. After the denaturing forces are removed, proteins tend to revert to their original structure. As the amino acid sequence is the only source of information to survive the denaturing process, the structural information must be somehow specified by the sequence. This discovery earned a Nobel prize for Christian Anfinsen in 1972 [4].

Protein sequences are comprised of amino acids, and amino acids are distinguished by their side chains. Most proteins exist in an aqueous solution within the cell, and certain amino acid side chains will tend to interact with the water molecules. Amino acids with such side chains are referred to as *hydrophilic* or *polar*. Their interaction with water often involves forming *hydrogen bonds*, a weak bond established when a partially negatively-charged atom in one molecule interacts with a partially positively-charged hydrogen of another molecule. *Hydrophobic* amino acids, in contrast, lack the atomic structure that would enable them make hydrogen bonds with water. Much as oil and water separate within a bottle, hydrophobic amino acids will tend to cluster together away from the water, while hydrophilic amino acids will tend to be located where they can interact with water. When hydrophobic molecules pack together, away from the water, they are said to share *hydrophobic bonds*. Thus, proteins will tend to organize themselves to bury the hydrophobic regions of the sequence within their interior, and to place the polar regions on the surface near the water or solvent. These *hydrophobic forces* are considered by most to be the dominant effect in protein folding [34], though some researchers consider hydrogen bonds to be just as important [139].

Another characteristic of protein structures is patterns of amino acid interaction that they exhibit. Amino acids are said to interact if they are close in the structure (within approximately eight angstroms) with no other molecules in between them. *Pairwise* interactions (involving two amino acids) have been studied at length, and show that there are definite patterns in pairwise preferences that amino acids exhibit. This suggests that protein structures might make small adjustments to accommodate the more favorable pairwise interactions. The most obvious pairwise preference is that cysteines tend to interact with other cysteines. Two cysteines can form *disulfide bonds*, strong bonds between two sulfur atoms. Disulfide bonds stabilize proteins structures significantly, and only cysteines are able to form them. Another obvious pairwise preference involves amino acid charge. Certain amino acids carry a formal unit charge: some positive, some negative. As one would expect, charged amino acids interact frequently with amino acids of opposite charges and infrequently with those of the same charge.

While certain forces stabilize protein structures, including those discussed above, certain other forces destabilize protein structures. When stabilizing forces are balanced against the destabilizing ones, proteins are only marginally stable. As most proteins are also short-lived within the cell, this marginal stability allows the cell to better adapt to changing conditions. One effect of this marginal stability is that when proteins are mutated, either in the laboratory or by evolutionary forces, the vast majority of mutants are not stable. Some positions are more tolerant to mutation, particularly those in loops on the surface of the protein. Yet sometimes, a change in just one critical residue can render a protein unstable.

2.3 Protein function

Most proteins are highly specialized. Each protein interacts with one particular compound within the cell, such as some other protein or a particular region of DNA. Proteins identify their target for interaction by their complementing shape, similar to three-dimensional jigsaw puzzle pieces, and by favorable interactions between residues on the interacting surfaces. Typically, only a small, localized portion of the protein is directly involved with the interaction. This portion is known as the *functional site*, and residues within the functional site are called *functional residues.*

The interaction between a protein and its target compound is also known as *docking*.

Protein structure and function are closely-related, since a protein's function is dependent on the three-dimensional structure of its functional site. Even though this might be a small portion of the protein, the rest of the protein plays an indirect role in its function by providing a structural framework for the active site. Mutants which alter the structure of a protein usually have an adverse effect on the protein's functionality.

2.4 Protein homology

Most proteins are not unique. Proteins in nature tend to fall within families, with the members of the family having similar amino acid sequences. Such families of similar proteins are said to be *homologs*.

First, proteins are often highly *conserved* or unchanged in evolution. A protein from a human might very closely resemble the corresponding protein in *Escherichia coli*, even though human and *E. coli* are quite different life forms. When two proteins are very closely related, and perform the same function, they are said to belong to the same *subfamily*.

Second, proteins performing similar functions are often found to be similar proteins. It is said that nature is a tinkerer, not an inventor [81]. Small changes in functionality are accommodated by adapting existing proteins. If two proteins have similar sequences and perform different functions, they are said to be of the same *family* or *superfamily*, depending on the level of similarity.

When one compares a protein to a homolog or family of homologs, differences that might appear include replacement of one or more residues (*substitution*), removal of one or more residues (*deletion*), or addition of one or more residues (*insertion*). *Sequence alignments*, discussed in depth later in this thesis, represent pairs or families of sequences to best show their similarity. Alignments are tables, with each row representing one sequence and with the columns representing regions of correspondence. Figure 2.4 shows an example of an alignment of four sequences.

```
XFTNVSCTTSKECWSVCQRLHNTSRGKCMNKKCRCYS.
VPINVSCTGSPQCIKPCKDAG-MRFGKCMNRKCHCTPk
--MCMPCFMAKKCRDCCG-----GNGKCFGPQCLCNR.
-------INHSACAAHCLLRG-NRGGYCNKGVCVCRN.
```

Figure 2.4: Example of a multiple alignment

When two homologous sequences are aligned, and a region of one sequence is deleted in the other, dashes in the relevant columns are used to indicate the deletions. Deletions are often called *gaps*. Figure 2.4, shows a long gap at the beginning of the fourth aligned sequence. Insertions are the analog to gaps: a region added to one sequence relative to the other. Figure 2.4 shows an insertion at the far right of the alignment, denoted by the *K* in lowercase, and the dots that serve as spacers in the other sequences. The term *indel* refers to either an insertion or deletion: something present in one sequence and absent in the other. When comparing two sequences, one can use deletions to represent any region contained in one sequence and not the other. However, when comparing a sequence to a family of homologs, it can be easier to think of a region as added to the sequence in question rather than deleted from all the other sequences. Sometimes, whether something is inserted by one sequence or deleted from the others is a question of what is the frame of reference. In Figure 2.4, there is a column containing and N in the first sequence and gaps in all other sequences. Turning this N into an insertion would be just as legitimate a representation. Indels and gaps can be used to indicate minor changes between two protein sequences, or more extensive changes when one protein contains a major region that corresponds to nothing in the other.

Pairwise alignments contain two sequences, and *multiple alignments* contain many sequences. Similarity between two proteins is generally measured in *percent identity*, the percentage of alignment columns in which the two proteins have the same residue.

2.4.1 Proteins with similar sequences tend to have similar structures

Given that protein structure is somehow specified by protein sequence, one might expect that proteins with similar sequences have similar structures. With a few exceptions [32], this is true. Sequences with as low as 30% identity usually have very similar three-dimensional structures [158, 154]. The opposite is not true. Two proteins can have very different sequences, but nearly identical structures [51].

2.4.2 Evolutionary changes between proteins

When evolution substitutes one amino acid within a protein for another, not all substitutions are equally likely. Certain pairs of amino acids tend to substitute for each other more frequently. For example, replacement of a hydrophobic residue by another hydrophobic residue is common; replacement of a hydrophobic residue by a polar residue is less common.

Within any protein structure, only certain changes might be permitted. In the tightly-packed interior of the protein, a residue might be replaced by only residues of similar size and shape. While insertions and deletions might be accommodated near the surface of the protein, indels in the core are far more rare.

Finally, certain positions are so critical to the structure or function of a protein that they cannot tolerate any mutation. Generally, if a certain residue does not change within a family of proteins, one can infer that it cannot be changed. Such a residue is said to be *conserved* within the family. If changes are permitted, but only changes that preserve some amino acid property, that property is said to be conserved. Residues important for stabilizing a protein structure tend to be conserved within the protein superfamily. Functional residues tend to be conserved within protein subfamilies, but vary between different subfamilies [134].

2.4.3 Remote homology and structural homology

While similar sequences tend to fold into similar structures, proteins with similar structures do not always have similar sequences. There are many examples of proteins with very similar structure with little or no sequence homology [51, 68, 73]. Such cases are referred to as *structural homologs*. In cases where the sequence similarity is weak but discernible, such proteins are referred to as *remote homologs*. The range of homology that typifies remote homologs is referred to as the *twilight zone*. This range is typically described in terms of *percent identity*, the number of alignment positions in which both sequences have residues of the same amino acid type, in proportion to the length of the sequences. By common estimates, two sequences are in the twilight zone when they are less than 25% identical [154, 182].

While protein structure prediction algorithms can accurately predict homology even in the twilight zone, structural homology is generally ascertained by direct comparison of protein structures. A family of algorithms called *structural aligners* compare the structures of two proteins by superimposing portions of the structures, and searching for a superposition that best highlights any similarity between the two structures. From this *structural superposition*, a *structural alignment* can be derived to show how the two sequences should be aligned according to the similarity of their structures. While some systems merely build pairwise structural alignments, others organize protein structures into families and hierarchies. Examples of the former include the Yale aligner [50] and CE [161]. Examples of the latter include FSSP [68], VAST [51], and CATH [136]. The SCOP database [73] also organizes proteins according to structural similarity, and is probably the most widely-respected structural classification system at this time. However, unlike the previous systems, which rely on an automated method for detecting structural similarity, SCOP relies on the encyclopedic protein knowledge and photographic memory of a single individual: Alexey Murzin. While structural classification by human inspection might sound eccentric, Murzin's method should not be taken lightly; in the CASP2 blind structural prediction experiment, Murzin's "knowledge-

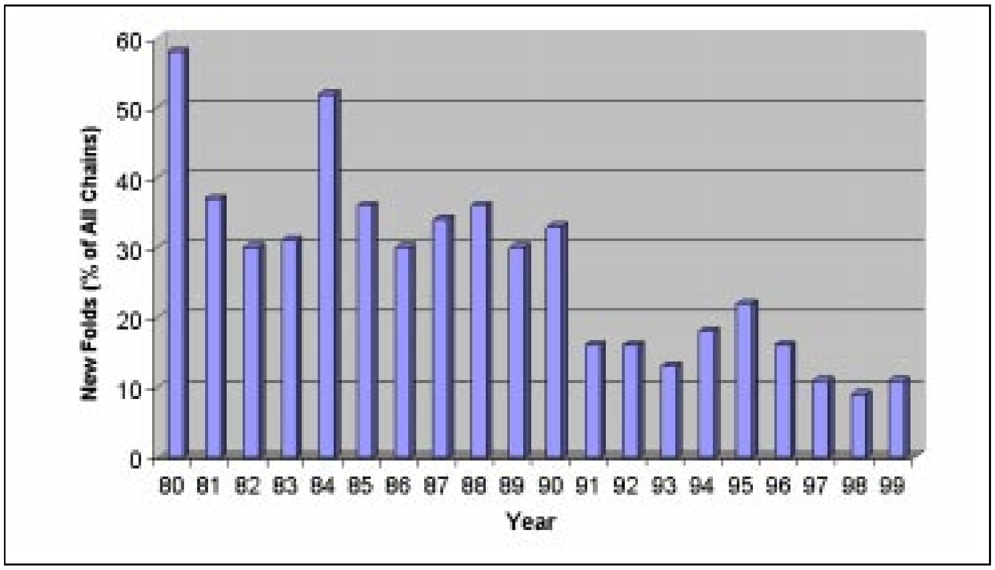


Figure 2.5: As new protein structures are added to the Protein Data Bank (PDB), the rate at which novel folds have been found has decreased steadily over recent years. Currently, approximately one tenth of new structures submitted to the PDB represent a novel fold. *Source: the Protein Data Bank (PDB)* at `http://www.rcsb.org/pdb/`.

based" method outperformed the best automated methods in the world [109].

One fact supporting structural homology is that the number of distinct protein structures seems to be limited in nature. For any given protein sequence, the number of conformations it can adopt is limited by biochemical and biophysical forces. While the number of physically-possible conformations increases exponentially with respect to sequence length, it is orders of magnitude smaller than the number of geometrically-feasible arrangements of the protein's atoms. As new protein structures are obtained, the frequency with which new structures are discovered has steadily decreased over the years, as seen in Figure 2.5. As seen in this figure, when a new protein structure is determined, the odds that it will be from a novel fold are approximately one in ten. Cyrus Chothia once projected that the total number of distinct folds in natures was only a few thousand [26]. This estimate has since gone up with discoveries of novel folds, but the number of distinct folds in nature is still believed to be very small compared to the number of proteins.

2.5 Overview of protein structure prediction

The fact that protein sequences and protein structures both occur in families makes protein structure prediction a viable pursuit. Depending on the protein in question (the *target* protein) and the extent of its relationship, if any, to proteins of known structure, structure prediction methods can yield structural models that fall within a few Angstroms root-mean-squared deviation of the actual structure. Recent growth in accurate, fast, automated structure prediction algorithms has made predicted structures viable alternatives to obtaining actual structures.

Protein structure prediction methods fall into two major categories: those which assume that the protein is similar in structure to some protein of known structure (*fold recognition* methods), and those which do not (*ab initio* methods). Ab initio methods estimate protein structure by searching for a conformation that minimizes biophysical and biochemical forces. A major drawback of these methods is computational complexity, with the huge number of potential conformations to explore. A second problem is the difficulty of accurately estimating the forces defining protein structure. In the most recent CASP contest, ab initio methods yielded viable predictions only for very short proteins [127]; when the protein had a structural homolog, fold recognition methods performed much better than ab initio methods [128].

Compared to ab initio methods, fold recognition makes structure prediction closer to a multiple-choice question than an essay question. Fold recognition methods predict the structure of a protein by searching the protein structure databases for a fold family that best fits the protein, and then describing which portions of the protein will adopt which portions of the fold. The drawback of fold recognition methods is that they are only viable for proteins with homologs of known structure, and their effectiveness is limited by the extent of the structural similarity. However, as “structure space” becomes more densely-populated, this limitation should become less stringent.

Fold-recognition methods fall into two major classes: *threading* and *homology recognition*. Threading predicts the structure of the target sequence by identifying a *template* sequence, a sequence predicted to be of similar structure, and finding some mapping of the target sequence onto the template structure that minimizes estimated biophysical energy. While this process can be augmented by information on homologs of the template sequence, the emphasis is on a favorable match between the target sequence and the template structure. For contrast, homology recognition methods predict structure on the basis of similarity of the target sequence to a template sequence and its homologs. The process might be augmented by structural information, but the emphasis is on sequence similarity. The drawback of such methods is that their effectiveness is limited by the extent of the similarity, though accurate predictions can be made well into the twilight zone [92]. A major drawback of threading methods is their computational complexity. As evidence of this point, threading groups do not tend to offer web server interfaces for their methods; no threading groups competed in the first CAFASP contest of protein structure prediction servers [46].

Still another distinction between protein structure prediction methods concerns the level of detail of the prediction. The methods discussed to this point generally yield low-resolution structure predictions. Homology recognition methods generally predict an alignment, detailing portions of the template structure and target structure predicted to be similar. This level of resolution is equivalent to predicting the location of the backbone atoms of the target protein. Threading algorithms rarely predict more than the approximate location of the center of the amino acid side chains. *Homology modeling*, in contrast, yields a very high-resolution model, detailing the position of every atom in the structure of the target protein. Homology modeling methods usually start with an alignment or other low-resolution model and refine the structure progressively. As such, the quality of a homology model depends strongly on the quality of the starting alignment; errors in the starting alignment are the most frequent and most serious source of errors in homology modeling [173]. For this reason, homology modeling methods are usually only applied to fairly close homologs, for which an accurate alignment can be predicted with high confidence.

A final class of structure prediction algorithms are *docking prediction* algorithms. Rather than predicting the structure of the target protein, they study the target protein and the compound, protein, or nucleic acid with which it interacts, predict the functional site of the target protein, and predict the nature and orientation of the interaction.

2.6 Structural genomics

Anyone who reads newspapers is probably aware that the human genome is expected to be fully sequenced quite soon. Yet the average newspaper reader might not be aware that this is merely one of many genome projects underway. According to The Institute for Genomic Research (TIGR) at **http://www.tigr.org/**, there are over thirty microbial organisms with fully-sequenced genomes, and genome projects are underway for over one hundred microbial organisms.

Once a genome is fully sequenced, the next step is to identify the genes and the proteins that they code for. Gene prediction systems can predict which sections of DNA code for genes with over 90% accuracy [103]. Once predicted genes have been identified, the next steps involve identifying and characterizing the proteins that might be *expressed*, or produced from these genes, and the circumstances through which each protein is expressed. The recently-coined term *proteomics* describes characterization of the protein output and expression patterns of a particular genome.

Most protein science research has followed a "top-down" approach, where proteins are chosen for study typically on the basis of their biochemical role. Therefore, some functional information on a protein has usually been available when protein scientists set out to analyze it. Genomic and proteomic efforts, in contrast, have given way to a "bottom-up" approach, whereby proteins are identified for investigation by the fact that they seem to be expressed, and thus probably have a biochemical function, but nothing more is known about them. As of April, 2000, of the 1644 genes identified in the *Haemophilus influenzae* genome, 691 code for "hypothetical proteins": proteins with no homologs of known structure or function [24]. These numbers are typical; as new genomes are analyzed, approximately one third of the proteins encoded by the DNA tend to be hypothetical [44]. Elucidating the function of these hypothetical proteins represents an exciting new scientific challenge, as knowledge of the full repertoire of the proteomic information of a functional life-form is quite likely to yield major breakthroughs in scientific and medical research. Cross-genomic studies reveal that hypothetical proteins are often conserved among several organisms. In other words, many life-forms have a portion of their genetic code that we cannot yet elucidate, and the portions we cannot yet elucidate is similar for many life-forms. This is a clear signal from nature on what gaps of information to fill in next.

The goal of *structural genomics* is to provide structural information for the complete protein repertoire of a genome, whether the information is experimentally-determined structures or plausible homology models. This structural information can aid in the determination of function where unknown, or can yield insights to biological mechanisms where the function is known. A major prerequisite for this goal is determination of the structure of one member of each sequence family. While structure determination remains no trivial task, determining the structure of a hypothetical protein can yield some of the greatest rewards for the effort, in terms of scientific progress. Databases such as PRESAGE [16] have been created to support this effort by identifying good candidates for structure determination and listing updates on experimental status.

The day might not be far off when every new protein has a homolog of known structure. Advent of that day represents "the slow death of the (natural) protein folding problem" [3], as ascertaining the structure of a novel protein will become a task of fold recognition and homology modeling. Meanwhile, this vision demands that the research community prepare for such a day by creating highly accurate and automated methods for fold recognition, sequence alignment, and homology modeling.

Chapter 3

Background on sequence alignment algorithms

The focus of this thesis is on improving alignments produced by hidden Markov models (HMMs). To assess the work in the later chapters, the reader should have some understanding of HMMs and other alignment algorithms. The necessary background is provided in this chapter. Section 3.1 gives a brief introduction to HMMs. An overview of alignment methods is given in Section 3.2. This overview describes various classes of alignment algorithms: pairwise methods (Section 3.2.1); profile methods (Section 3.2.2), and multiple alignment methods (Section 3.2.3). Finally, Section 3.2.4 reviews the literature comparing different alignment methods.

3.1 Brief primer on HMMs

Hidden Markov models (HMMs) are statistical models with a long history of successful application to speech recognition [149]. To an average person, a spoken word is a series of syllables. To a linguist, each syllable can be further subdivided into a series of *frames*, units of 10 to 20 milliseconds in length. When sequences of frames are assembled into words, certain characteristics of each sequence will characterize each word. Other characteristics will vary between speakers, particularly those related to accent and cadence of speech. HMMs construct a generalized profile of each word, in which the more salient or distinguishing characteristics are expected with high probability. Then, when a speaker utters a word, the word is recognized by comparing its sequence of frames against the HMMs for various words to see which it matches best.

The problem of homology detection compares favorably to that of speech recognition. Members of a protein family share certain characteristics, such as presence of conserved motifs. In other respects, such as precise location of the conserved motifs within the protein sequence, there can be marked differences between members of the same family. The goal of HMMs for sequence analysis is to model each protein family in such a way that the distinguishing characteristics are expected with high probability while variation is permitted in the more varying characteristics. Then, when a new and potentially homologous sequence is presented to the model, the model estimates the likelihood that the sequence is a new homolog, given the characteristics that most directly define the protein family.

The use of HMMs for sequence analysis was pioneered in 1993 at an esteemed institution within the University of California system [100], with related publications from other institutions following shortly afterwards [7, 43]. Since then, HMMs have proven effective at protein structure prediction in blind experiments in two CASP contests [94, 92]. HMMs have been applied to related sequence analysis problems including recognizing protein structure based on secondary sequence [48], gene prediction [103], motif detection [23], motif assembly [59], and have been the subject of a couple of reviews [42, 40].

Figure 3.1 depicts an HMM, along with the path taken through the model by two sequences and the alignment generated subsequently. There are three types of states in an HMM: *match, insert*, and *delete* states. The three types of states are represented with squares, diamonds, and circles respectively. There is a one-to-one correspondence between columns in an alignment and match and delete states in an HMM. When a sequence enters a match state, it aligns its next residue in the corresponding alignment column. It enters a delete state if it has no appropriate residue to align to the column; using the delete state is equivalent to opening or extending a gap in the alignment. At each column of the alignment, the sequence must enter either the match or delete state. In addition, between alignment columns, it can enter the insert state and emit one or more residue. Entering the insert state is equivalent to inserting residues between columns of the alignment.

The path that a sequence will take through the model is determined by a series of costs. At each match column, each amino acid can be emitted, or aligned to the column, according to a cost computed from the probability of aligning each amino acid at that column. Similarly, at each insert state, there is a cost table reflecting the probability of inserting each amino acid at the respective alignment position. Finally, the model can move between states according to a predetermined set of transitions, each with its own cost. At each match or delete state, the sequence can move, or transition, to the match or delete state of the next column, or to the insert state before the next column. At each insert state, the model can remain in the insert state and insert an additional residue, or can move to the next match or delete state.

The parameters in an HMM are estimated according to a *training alignment* or *seed alignment*. In the SAM [78] HMM software package, the `modelfromalign` module estimates an HMM according to a training alignment. HMM training typically involves various mechanisms for making a more general model. Such mechanisms include sequence weighting, column regularizers, and transition regularizers. Biological databases are notoriously redundant, and sequence weighting seeks to address this by giving each sequence a weight according to its distinctiveness in the alignment. Column regularizers compute an amino acid probability distribution for each column according to the data seen in the column and to patterns of amino acid conservation. Dirichlet mixtures [166, 91] are typically used for this purpose. Transition regularizers are systems of pseudocounts applied to the model such that no matter what appears in the alignment, all transitions have a cost that is finite, and hopefully sensible.

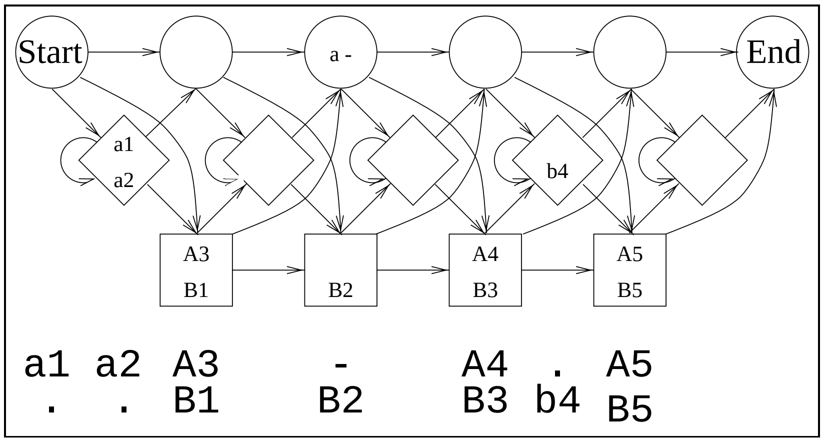


Figure 3.1: Example of a hidden Markov model, showing the path taken through the model to emit the `A` and `B` sequences, where `A` = `a1,a2,a3,a4,a5` and `B` = `b1,b2,b3,b4,b5`. Capitalization is used to distinguish letters generated from match states (squares) from those generated by insert states (diamonds). Circles represent delete states.

Other SAM modules of interest are summarized as follows. `Align2model` aligns a sequence to a model. By default, the alignment is estimated by the *Viterbi* algorithm, a dynamic programming algorithm for finding the lowest-cost path through the model that accounts for all residues in the sequence. Alternatively, the alignment can be determined by the *posterior decoding* algorithm. The two algorithms differ in how they estimate the cost of using each node in the model. While the Viterbi algorithm estimates the cost according to the single lowest-cost path to that node, posterior decoding estimates it according to all paths through the model. Posterior decoding requires more computing resources, but can yield more accurate alignments. When there are new homologs to align to a protein family, they can be aligned by first estimating a model from the existing protein family alignment, then using `align2model` to align these sequences plus the new homologs to the model. `Hmmscore` yields the overall cost of aligning a sequence to a model. This overall cost reflects the likelihood that the sequence is homologous to the training sequences. HMMs can also be used to align unaligned sequences, or to re-estimate an existing alignment. This is done with the `buildmodel` module, which uses the *forward-backward* algorithm.

3.2 Review of sequence alignment algorithms

The field of sequence alignment has a long history, but since there are a number of recent reviews [170, 5, 55, 39], I shall only touch on the major points.

Sequence alignment methods fall into two major categories: pairwise methods, which work with only two sequences, and multiple sequence methods, which can work with more than two sequences. Multiple sequences methods in turn fall into two categories: profile methods and multiple alignment estimation methods. Profile methods can add new sequences to an existing alignment, but do not otherwise modify the alignment. Multiple alignment estimation methods build or re-estimate a multiple alignment. Even when the goal is to merely add new sequences to an existing alignment, multiple alignment methods are worth considering for the following reason: every sequence added to the alignment represents added information. This added information might help to suggest a better overall alignment. While re-estimating a curated multiple alignment such as a Pfam [172] alignment might not be sensible, re-estimation of an automatically-generated alignment might be worthwhile when one adds new sequences.

3.2.1 Pairwise alignment methods

Of the many varieties of sequence alignment algorithms, pairwise algorithms are the simplest. The classic is the Needleman-Wunsch algorithm [130]. Given a system of substitution costs and gap costs, this algorithm uses dynamic programming to find the lowest-cost global alignment of the two sequences. The Smith-Waterman algorithm [171] finds the optimal local alignment of two sequences by a variation of this method. The alignment is permitted to start and end in the middle of the sequences by omitting low-scoring regions.

Heuristic algorithms, another class of pairwise alignment algorithms, do not promise optimal pairwise alignments but offer greater efficiency. Examples of such algorithms are FASTA [148] and BLAST [2]. Central to both of them is the idea that most alignments contain short regions of high identity or similarity, that such regions can be identified quickly, and that a larger alignment can be assembled from these short matches. FASTA divides the target sequence into small *k-tuples*, segments of one or two residues in length. It compares each k-tuple to equal-sized stretches in the template sequence, looking for identically-matching segments. Next, it derives ungapped a set of local alignments from the list of k-tuple matches by looking for diagonals with non-overlapping k-tuple matches. Finally, it combines various ungapped local alignments into a gapped global alignment by applying a system of gap penalties. BLAST divides the target sequence into short "neighborhood words", three residues in length by default. It then scans the template sequence for matches scoring above some threshold value, using a substitution matrix to score potential matches. When an acceptable match is found, the algorithm seeks to extend it in both directions, stopping according to a scoring heuristic. The most familiar version of BLAST generates only ungapped alignments, but can report multiple matches of the same sequence. Updated versions can generate gapped alignments, but retain the focus on generating short matches with strong signals. Because of their efficiency, heuristic algorithms are used widely as filters for database searches: given a search sequence, they scan the protein databases for potential matches. These potential matches are then analyzed by a more sensitive method, or by a human.

3.2.2 Profile alignment methods

Profile methods represent the next level of complexity in alignment algorithms. Given an alignment assumed to be reasonably accurate, the algorithm constructs a profile of the alignment. In general, this profile consists of a system of gap costs and a set of costs for aligning each of the twenty amino acids to each alignment column. These latter costs are derived from each column's amino acid probability distribution. The probabilities are derived from a system of *sequence weighting*, *column regularizing*, and the amino acids observed in each column. Sequence weighting accounts for the fact that biological databases are skewed toward the proteins most heavily studied, and that this skew tends to be reflected in protein family alignments. To tone down this skew, a weight is assigned to each aligned sequence according to its uniqueness within the alignment. Sequence weights generally range between 0 and 1, with weights close to 0 assigned to the sequences that more closely resemble others in the alignment. When the amino acid counts are then gathered at each column, they are scaled according to the sequence weights. Column regularizers then work to derive meaningful amino acid probability distributions from the weighted counts given amino acid substitution patterns and conservation of residues or biochemical properties. Some examples include Dirichlet mixtures [166] and data-dependent pseudocounts [176].

Profile methods were pioneered by Gribskov et. al [57]. Contemporary examples of systems that use profile information include TOPITS [151], GenThreader [86], PSI-BLAST [1], SAM-T98 [93] and CLUSTALW [82, 180]. This thesis emphasizes SAM-T98 and CLUSTALW. SAM-T98 is a fold recognition method built on iterative searches with the SAM hidden Markov model software. SAM and hidden Markov models are described in Chapter 5. The profile alignment functionality of CLUSTALW is described below.

The most familiar functionality in CLUSTALW is multiple alignment estimation, but it does provide functionality to align sequences to an existing profile. When used in this mode, it begins by calculating a phylogenic tree for the aligned sequences, using the neighbor-joining method [157]. The weight of each sequence is calculated according to the length of the branches in the tree from the root to the sequence. When a branch leads to more than one sequence, its weight is divided evenly between them. All sequence weights are normalized so that the largest sequence weight is 1. CLUSTALW has been described as "shamelessly ad-hoc" [38]; one can appreciate this description when looking at the intricate rules that CLUSTALW uses to set gap penalties. The rules are summarized as follows. [65].

1. Initial gap opening and gap extension penalties are set by the user or from default values.

2. If an alignment column contains gaps, its gap opening penalty is scaled down by 0.3 times the fraction of sequences with gaps in that column and its gap extension penalty is divided

by two.

3. If a column contains no gaps, but is within 8 residues (by default) of a column that contains gaps, its gap opening penalty is modified as follows: $gopen = gopen \times \left(2 + \frac{8 - D \times 2}{8}\right)$, where D is the distance to the gap.

4. If the column is amid a stretch of five or more hydrophilic residues according to any of the aligned sequences, its gap opening penalty is scaled down by 0.3.

5. Internal to the program is a set of amino acid gap priors, representing the likelihood with which each of the twenty amino acids is adjacent to a gap. If the column does not contain a gap or a stretch of five or more hydrophilic residues, its gap opening penalty is scaled according to the sequence-weighted average of the gap priors of the residues aligned.

Additionally, the user can specify a secondary structure mask, indicating the secondary structure of one of the aligned sequences. If a secondary structure mask is provided, the gap opening penalties are scaled according to a constant value for each secondary structure class, with special constant scale values for positions near the end of an α-helix or β-strand. This complicated-sounding set of rules has four aims: to decrease gap penalties where gaps already occur; to increase gap penalties near existing gaps; to decrease gap penalties near stretches of hydrophilic residues; and to scale gap penalties according to the amino acids aligned.

3.2.3 Multiple alignment estimation methods

Given a pool of sequences, multiple alignment estimation methods search for an alignment to maximize the overall homology. This is no simple task, and it has been addressed by many different types of methods.

Two-dimensional dynamic programming algorithms were described in Section 3.2.1. *N-dimensional dynamic programming* algorithms seek to simultaneously align N sequences. The computational complexity of this task is proportional to $N\,(2L)^N$, where N is the number of sequences to align and L is the length of the alignment. Given the exponential complexity, this is not done for more than 3 or 4 sequences [55]. There are methods that use approximations to reduce the complexity, the best-known of which is MSA [111]. However, even with approximations, MSA cannot align more than 10 sequences.

Progressive methods estimate the alignment in an agglomerative manner, starting by aligning two sequences. Next, either profile methods are used to align a third sequence to the pair, or two other sequences are aligned. The process is repeated until all sequences are aligned. After each step, the resulting alignment is fixed: once two sequences are aligned to each other, their alignment is not altered. The order in which sequences are aligned is typically decided on the basis of pairwise similarity. The multiple alignment estimation algorithm of CLUSTALW is a progressive method [82].

The advantage of progressive methods is that they generally yield sensible alignments while making efficient use of computing resources. The drawback is that they cannot recover from mistakes made early in the process. This is the motivation behind *iterative refinement* methods. Iterative refinement methods are essentially extensions of progressive methods. The difference is that they re-estimate the alignment as it is built, typically by removing certain sequences from the alignment, and then aligning them back in. PRRP is an example of such a method. It aligns sequences progressively according to a predicted evolutionary tree, and periodically reassesses both the evolutionary tree and the alignment under construction. Iterative refinement methods can generate excellent alignments, but they require more computing resources than progressive methods.

Finally, *stochastic* alignment methods modify parts of the alignment according to some probability function, assessing the value of the modifications according to some objective function. As is true with most applications of stochastic methodology, the drawback of stochastic alignment methods is they do not guarantee an optimal solution – even according to their own objective functions. However, they can build excellent alignments. One example of such a method is SAGA-COFFEE [133], a genetic algorithm for estimating multiple alignments. The objective function for

SAGA-COFFEE is weighted similarity between the multiple alignment and a library of pairwise alignments of the same sequences, with the weights assigned according to pairwise sequence similarity. As a genetic algorithm, it starts with an initial population of alignments, and the population is changed gradually according to a rough simulation of evolution. In this simulation, each member of the population is measured according to its fitness, or by the objective function. Death occurs at random, and with a greater frequency among the less-fit members. In contrast, more fit members have a greater probability of "breeding", creating new alignments that share most of their characteristics. Mutation occurs at random during breeding, with mutation events including movement of a gap or a block. While this method is not without computational overhead, one advantage is that it lends itself well to parallelization.

Hidden Markov models (HMMs) for multiple alignment estimation are another stochastic methods. To estimate a multiple alignment, HMMs start with an initial alignment and refine it over a number of iterations. In each iteration, for each sequence in the alignment, they first estimate the probability with which each residue might align to each column given all possible alignments of the sequence to the model. Then, they determine an alignment, or a path through the model, which aligns the residues of the sequence with greatest probability. *Simulated annealing* is used to avoid local minima. Random noise is injected into the model according to some probability. This amount of noise is high early in training. As training progresses and the model converges, the amount of noise injected is decreased.

3.2.4 Comparison of alignment methods

Given various options of alignment methods, we return to the central question: once a fold is identified, what is the best way to align the target and template sequences? To answer this question, one might be tempted to turn to the results of the CASP contests. However, human intervention plays a significant role in many contest teams. Some of the teams that achieved the best alignment results stated openly that their methods involved manual refinement [11, 99, 92], though one particular HMM team was forced to admit that their human intervention had not improved their alignments in general [92].

To separate the effects of human intervention from automated methods, the CASP3 contest included CAFASP1, a contest on automated servers [46]. However, the CAFASP1 contest did not address alignment quality.

When reading the literature comparing various sequence alignment methods, the unequivocal rule is that if the comparison involves a method developed by the authors, their method will be among the best. Exceptions to this rule are as follows. First, if there are various tests with no overall winner, the tests on which the authors' method performed best will later be deemed the most significant. Second, if the comparison involves two or more methods by the authors, their early methods are permitted to fail on the condition that their refined methods are successful. This situation cries out for a blind prediction experiment on alignment quality. Unfortunately, there is no such experiment at this time, so we must content ourselves with papers in which the authors' method will invariably be described as the best.

Gotoh [54] compared the performance of three alignment methods on their ability to align two sequences consistently with their structural alignments. The test set consisted of approximately 50 protein families, each consisting of from 2 to 10 sequences, each with at least two members of known structure, and covering the range from 10% to 50% average identity. The three methods tested were Needleman-Wunsch pairwise alignment, CLUSTALW multiple alignment, and PRRP multiple alignment. Both CLUSTALW and PRRP are described in Section 3.2.3. For all methods, he experimented with various gap parameters and substitution matrices. Accuracy of the alignments produced were evaluated in a global manner: the number of pairs of residues aligned correctly as a proportion of the number of residues in the two sequences. PRRP performed better than CLUSTALW, which in turn performed better than Needleman-Wunsch. As sequence identity decreased, the gap in performance increased in both cases. Gotoh interpreted these results to say that while iterative methods are not as simple as progressive alignment methods, the extra complexity is justified. The accuracy of the multiple alignment methods increased as the number of sequences increased. All methods were very sensitive to the choice of gap parameters, although none were especially sensitive to the choice of substitution matrices.

Two years later, Notredame et al [133] compared CLUSTALW and PRRP to a number of other methods for multiple alignment estimation. Other methods evaluated were SAM, described earlier; PILEUP [64], an early variation on the CLUSTALW algorithm; SAGA-COFFEE, the authors' genetic algorithm for multiple alignment estimation, described in Section 3.2.3; and SAGA-MSA, a related genetic algorithm that uses as an objective function similarity with alignments produced by the N-dimensional dynamic programming algorithm MSA. All methods were run with default parameters, with the exception that a Dirichlet mixture was used as a column regularizer for SAM. Nonetheless, SAM would probably not fare well in such an analysis, as its performance is quite sensitive to parameter settings and the default values are often not the best.

The test set for this study was eleven alignments from 3d_ali [147], a database of multiple alignments combining structural alignments with sequence-based alignments of close homologs. The test cases were selected such that each had at least five sequences, with a consensus length of 50 or greater. In addition, the authors removed from consideration four alignments which CLUSTALW could reproduce with 95% or greater identity and three alignments which neither CLUSTALW nor SAGA-COFFEE could reproduce with any accuracy. The remaining eleven alignments had average identities ranging from 17 to 61%. Methods were not evaluated relative to the entire 3d_ali alignment; rather, they were evaluated on the positions for which the secondary structure was conserved within the 3d_ali alignment. All methods were scored according to the proportion of residue pairs in such columns that they aligned accurately.

As might be expected, SAM fared the worst in this analysis. It had the worst performance in 10 out of 11 cases; in the eleventh case, its score was the second-best, but all methods were close in score. Its performance was less horrible in cases where the alignments had the greatest number of sequences, probably reflecting that the version of SAM used had no default sequence weighting.

The methods that performed the best were PRRP and SAGA-COFFEE, with SAGA-COFFEE performing better in cases with lower average identity. CLUSTALW, SAGA-MSA, and PILEUP yielded similar performance measures in most cases. The authors interpreted these results as support for the complexity of stochastic methods for multiple alignment.

Most recently, Thompson et. al [182] compared the performance of PRRP, CLUSTALX [82], PILEUP, and SAGA-COFFEE with six other methods: MULTALIGN [10], MULTAL [178], PIMA [169], DIALIGN [125] and HMMT [41]. CLUSTALX is simply the CLUSTALW program with an X-windows interface. MULTALIGN, MULTAL, PILEUP, and CLUSTALW are progressive global aligners; their main differences concern the order in which they add sequences to the alignment. HMMT is part of the HMMER HMM package. PIMA is a local aligner with Smith-Waterman pairwise alignment, secondary structure-dependent gap parameters, and progressive alignment construction. DIALIGN constructs multiple sequence alignments from local alignments based on comparisons between *diagonals*, subsequences of potential alignment, rather than individual residues. The authors do not state what parameter settings were used, but they probably used default parameters. The authors of this paper have worked with the authors of the previous paper, and one might guess that their methods were similar to those of their colleagues.

All methods were tested on the BAliBASE alignment benchmark set [181]. BAliBASE is a set of 142 multiple alignments, curated according to the literature on the respective protein families and to various structural aligners. The alignments are annotated with *core regions*, regions over which the alignment is judged most reliable by the curators. Most results in this study were measured over the core regions only; I shall indicate the exceptions. Two measures of alignment accuracy were collected: the number of pairs of residues aligned accurately, and the number of columns in which all residues were aligned accurately. BAliBASE is divided into five subsets, with each subset representing a distinct class of alignment test.

The first test involved alignment of families of sequences of equidistant homology: less than 25%, between 20 and 40%, and greater than 30%. All methods achieved close to 100% accuracy in the two latter cases, with one exception. That exception was HMMT. The most likely explanation is the same as that for the poor performance of SAM in the previous study: default parameter settings. On the first set, the best methods dropped to just below 70% accuracy while HMMT achieved approximately 0% accuracy. The best results were achieved by PRRP, CLUSTALW, and SAGA-COFFEE, with global methods generally performing better than local methods.

The second test involved alignments in which there was one *orphan*, divergent family member. Alignment accuracy was measured over all columns rather than just the core regions, and was measured for the overall alignment and for alignment or the orphan only. In general, the presence of the orphan had little negative impact on overall alignment quality. In terms of accurate alignment of the orphan, the poorest performance was seen by HMMT, and global methods generally performed better than local ones. What methods performed best overall is not entirely clear; the text states that PRRP was outperformed by CLUSTALW and SAGA-COFFEE, while the accompanying figure shows SAGA-COFFEE outperformed by CLUSTALW and PRRP.

The third test involved alignment of equidistant subfamilies of closely-related sequences. The results of this test were consistent with those of the first test.

While the first three tests involved alignment of sequences of approximately the same length, the fourth test involved alignment with large N-terminal and C-terminal extensions. The ranking of methods on this test was rather different from those for the previous tests: local methods performed best by far, with DIALIGN performing the best overall. CLUSTALW, SAGA-COFFEE, and PRRP showed equally-poor performance, and no results were shown for HMMT.

The final test involved alignments with large internal insertions. The best performance was achieved by the local method DIALIGN, with CLUSTALW, SAGA-COFFEE, and PRRP also generating strong results.

In summary, the authors observed that best performance was achieved by global methods, though DIALIGN seemed to offer some promising innovations. Iterative and stochastic refinement methods yielded better performance than most progressive alignment methods, though the authors noted the extra computational overhead required by these methods. No particular explanation was given for the success of CLUSTALW over other progressive aligners. Finally, the authors noted that while many methods could align moderate homologs very accurately, no methods achieved the same accuracy on remote homologs, with a dividing line of approximately 20% identity. The authors pointed that this is the area where the next generation of improvements must be made.

Briffeuil et al [19] compared the performance of MATCH-BOX, their local multiple alignment method, with six other multiple alignment methods for which there are web servers. All methods were run with default parameters, although MATCH-BOX was tested at three different reliability thresholds. The methods tested were CLUSTALW, MSA [111], and PIMA [169], described earlier; MAP [72], a global aligner that estimates multiple alignments through an iterative pairwise method; Block Maker [61], a local aligner that uses the Motifj algorithm to identify and merge conserved motifs; and MEME [6], a motif-based local multiple aligner. Their own method, MATCH-BOX [33], generates local multiple alignments by a method of iterative scanning for significant motif matches. A match is considered significant according to statistics estimating the probability that the score would be achieved by random chance, and an alignment region is regarded as significant if all sequences contain significant matches in the same region.

All methods were tested on 20 families, each of which included at least three sequences of known structure. Reference alignments were produced by Insight ©and Homology ©, commercial products distributed by Molecular Simulations of San Diego, CA. The methods were tested on their accuracy on recovering *structurally-conserved regions* (SCRs), regions where the structural alignments superimposed the C_α residues with an RMS deviation of 1.8 angstroms or less. Predicted SCRs were defined as ungapped regions in the multiple alignment, as estimated by each method. All methods were scored according to the *specificity* and *sensitivity* with which they reproduced the SCRs. Specificity is the number of correctly predicted residue pairs as a fraction of the number predicted, and sensitivity is the number of correctly predicted residue pairs as a fraction of the number of correct pairs.

In their results, the global alignment methods were characterized by an inverse relationship between specificity and sensitivity. On families for which the average identity was 10% or better, the global methods all achieved strong sensitivity and respectable specificity, indicating that they aligned the SCRs correctly but tended to align too much. Below 10% average identity, all exhibited a marked decrease in both sensitivity and specificity. Of all the global methods, MAP showed least degradation in performance. In contrast, performance of the local aligners was not greatly affected by the similarity of the sequences aligned. In general, these methods did not have the sensitivity of the global methods – not surprisingly. Their specificity results were scattered across a wide range, with the exception of those for MATCH-BOX. At the most stringent reliability level, the MATCH-BOX alignments achieved very high specificity and low sensitivity. This indicates that what was aligned tended to be correct, but that not much was aligned. As they dropped the reliability threshold, the results exhibited an increase in sensitivity with a decrease in specificity. At the lowest level of reliability, the results appeared almost but not quite as good as those for MAP.

In a second test, the authors tripled the size of the alignments by adding two very close homologs for each original sequence. When their tests were repeated with the additional homologs, MEME showed a significant increase in sensitivity, MATCH-BOX showed a significant increase in specificity, and the remaining methods were not greatly affected. Therefore, the authors cautioned that adding additional homologs does not necessarily improve alignment accuracy, and can unfairly overload someone else's server.

More recently, MATCH-BOX was compared to other methods by Hudak and McClure [77]. This paper evaluates methods on their accuracy at aligning conserved motifs. These motifs are identified by a combination of expert analysis and laboratory work to identify the structurally or functionally-important residues. All methods are carefully optimized prior to testing. In previous studies, McClure and co-authors had found that contrary to expectation, global methods often perform better than local methods [119] and SAM performed well [118]. In this most recent study, the authors compared seven different multiple alignment methods on their alignment of conserved motifs in the reverse transcriptase family.

The methods tested by Hudak and McClure were SAM [78], MATCH-BOX [33], PIMA [169], Block Maker [61], and MEME [6], described earlier; ITERALIGN [20], an iterative motif detection and alignment method; PROBE [132], a local aligner that identifies matches with Smith-Waterman alignments and assembles them into a multiple alignment with Gibbs sampling. In contrast to the authors of the previous study, Hudak and McClure did not observe strong performance with Match-Box. Perhaps this is due to a slightly different definition of the portions of the sequence of interest, or perhaps it is due somehow to the reverse transcriptase family. They found that while all methods were able to detect the conserved Motif IV, only ITERALIGN, MEME, SAM, and PROBE were able to detect the entire series of motifs, with PROBE achieving the best performance overall.

While there have been some studies on the accuracy of multiple alignment estimation methods, few authors have touched on profile alignment methods. One exception is the work of Sauder, Arthur, and Dunbrack [159]. This team is best known for their work on homology modeling [37] and their goal was to evaluate methods according the utility of their alignments for homology modeling. Homology modeling places two contradictory demands on alignment methods. First, alignments should be accurate, as alignment errors represent the most serious class of errors in homology modeling [85]. Second, alignments should be long, because any region that is not aligned represents a region that is not modeled. Therefore, they evaluated predicted alignments according to three criteria: specificity, sensitivity, and length.

Their assessment was far larger than any of the related work. Their test set involved all pairs of structures related by SCOP [73] at the family or superfamily level. In later analysis, they excluded immunoglobulins, which dominate all of the protein databases. Even focusing on non-immunglobulin pairs, they had a test set of 10,665 data points, nearly all of which had a sequence identity of 25% or less. Structural alignments for all pairs were obtained from the CE structural aligner [161], and were compared to structural alignments from FSSP where available. Results of this comparison were used as an accuracy target, to determine if there is still room for improvement in the methods.

The sequence alignments methods tested were BLAST, PSI-BLAST, CLUSTALW, and PSI-BLAST filtered by ISS [145]. Alignments were generated with each method as follows. With CLUSTALW, the authors generated a multiple alignment of the template sequence, target sequence, and all members of the template sequence's family according to SCOP. The pairwise alignment of the template and target sequence was extracted and scored. BLAST alignments were obtained by searching a database of SCOP sequences with the target sequence, and extracting the alignment of the template sequence. PSI-BLAST alignments were generated by seeding a PSI-BLAST query with the target sequence, and gathering homologs through four iterations of searches against the non-redundant database. A matrix of posteriors for each column was then extracted, and used with BLAST to search the database of SCOP sequences. PSI-BLAST with ISS performed two such PSI-BLAST searches: one seeded with the target sequence, and one with the template sequence. All intermediate sequences common to both resulting alignments were identified. A set of template-target alignments were obtained from the template-intermediate and intermediate-target alignments by using the intermediate sequence to infer a template-target alignment. The longest such alignment was used as the final alignment. All methods were optimized on a subset of the data.

Because the focus of the article was alignment quality and not fold recognition, the authors did not address fold recognition accuracy; with the BLAST methods, they ignored cases in which the desired pair was not reported because the hit was not significant. At the 20-25% identity level, nearly all SCOP structure pairs were reported, even by BLAST. At 10-15%, BLAST aligned 8% of the structure pairs, PSI-BLAST 17%, and PSI-BLAST with ISS 38%. As CLUSTALW has no fold recognition component, it aligned 100% of all structure pairs. This fact must be remembered later when the authors report that the results obtained with CLUSTALW in this range were not very good. Better-performing methods were BLAST with 28% sensitivity, PSI-BLAST with 40%, and PSI-BLAST with ISS at 46%. Nothing was reported regarding the relative accuracy of the methods for the subset of the structure pairs that they all aligned. However, in regards to both fold recognition and alignment accuracy, PSI-BLAST with ISS achieved better performance than PSI-BLAST alone, which in turn achieved better performance than BLAST. The lengths of the alignments produced followed the same rank order, indicating that this rank order reflects the value of the alignment method for homology modeling.

In summary, there have not been many statistical analyses comparing the quality of alignments produced by various methods. This body of literature is not only small, it is contradictory. However, there seem to be a few consistent lessons. First, methods that estimate or use multiple alignments generate more accurate pairwise alignments than pairwise methods do. Second, many methods will generate accurate alignments when the level of homology is 20% or better; the real challenge is accuracy below this threshold. Third, if the objective is to estimate a multiple alignment, iterative refinement or stochastic methods often require more computational resources than progressive methods, but their overhead seems to be worthwhile. Fourth, global methods can work as well as local methods, even if the scoring system favors short, accurate alignments.

Chapter 4

Measurement of alignment quality

This chapter describes the *shift score*, a score developed to measure the quality of a predicted alignment in relation to a structural alignment of the same sequences. Section 4.1 provides background information and introduces the terminology used later in this chapter. Related work is described in Section 4.2. Section 4.3 describes the calculation of the shift score, and Section 4.4 validates it by comparing it with other popular alignment quality measures. Section 4.5 provides examples produced by a companion visualization program. Certain issues should be considered whenever one uses structural alignments to measure alignment quality; Section 4.6 discusses those issues and how we have addressed them.

Closely related to the shift score is the *optimal subalignment score*, the score of the best portion of the alignment. Optimal subalignments are calculated from predicted alignments using a simple greedy algorithm, as discussed in Section 4.3.

4.1 Background and terminology

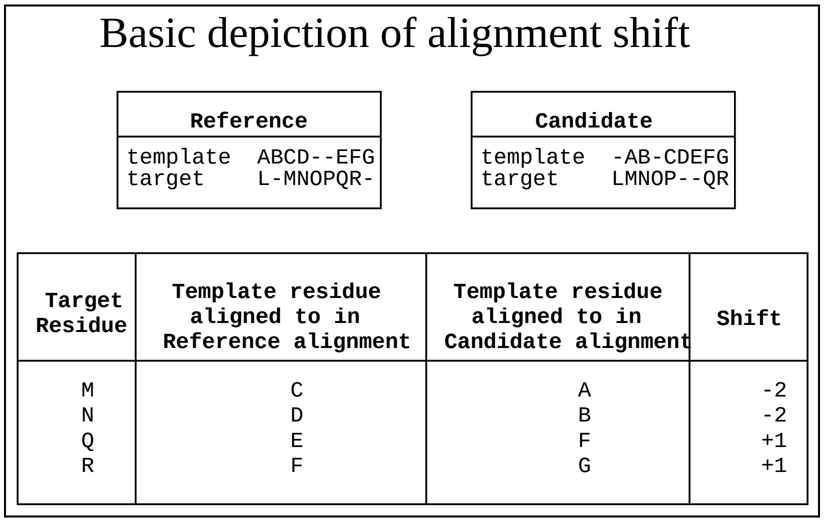
Sequence alignments fall into two major classes depending on the number of sequences aligned. *Pairwise alignments* contain two sequences while *multiple alignments* contain more than two. Similarly, alignment quality measures can involve all sequences in the alignment, or can focus on two sequences in particular. Such *pairwise* alignment quality measures can be made on multiple alignments, with sequences other than the two sequences of interest ignored. The measure discussed in this chapter is a pairwise quality measure.

In pairwise alignment quality measures, the two sequences studied are referred to as the *template* and *target* sequences. In a structure prediction scenario, the template sequence would have a known structure, and the target would be a sequence of unknown structure aligned to the template. More generally, for the sake of testing an alignment program, the algorithm cannot use information pertaining to the structure of the target sequence, even if known. Structure of the template sequence, in contrast, can figure into the alignment algorithm.

The most common application for alignment quality assessment is in experiments where alignment methods are being refined, tested, or optimized. Such experiments typically involve comparing one alignment with another which is trusted or considered correct. This second alignment is referred to as the *reference alignment.* Frequently, reference alignments are *structural alignments*, alignments produced by tools that study the template and target sequence structures to identify regions of correspondence. The alignment under test is referred to as the *candidate alignment.* The candidate alignment is sometimes also referred to as the *predicted alignment*, because it represents a prediction of the structural similarity between the template and target sequences.

Consider Figure 4.1 and the pair of residues C and M aligned in the reference alignment. In the candidate alignment, target residue M is aligned to template residue A rather than C. The *shift* of M is defined as -2, the number of positions between A and C in the template sequence. Note that the shift can be positive or negative, depending on the direction of the shift. A positive shift moves residue M closer to the C-terminus of the sequences, or to the right in Figure 4.1. If a residue is not aligned in either alignment, its shift is undefined. Hence, in Figure 4.1, no shift is listed for target residues L, O, or P.

Because shift is measured in sequence positions rather than alignment columns, the shift



Basic depiction of alignment shift

```
Reference
template  ABCD--EFG
target    L-MNOPQR-
```

```
Candidate
template  -AB-CDEFG
target    LMNOP--QR
```

Target Residue	Template residue aligned to in Reference alignment	Template residue aligned to in Candidate alignment	Shift
M	C	A	-2
N	D	B	-2
Q	E	F	+1
R	F	G	+1

Figure 4.1: Illustration of the shift of a single residue. Shift is measured for target residues aligned in both alignments, and refers to the number of template residues between its position in the two alignments.

of a residue is not affected by lengths of gaps between its two positions. For example, residue *M* in Figure 4.1 has shifted by two sequence positions and three alignment columns, yet we consider its shift to be two sequence positions.

If the target and template sequences are swapped, the shift measures can change radically due to differences in the lengths of any unaligned segments. If the reference and candidate alignments are swapped, shift measures will merely change in sign.

4.2 Related work

There are a number of scenarios in which a quantifier of alignment quality is vital. One is a CASP contest [127], in which there are many different alignments of the same sequence and the best must be chosen objectively. Another scenario is in refining methods for generating alignments; over a test set of hundreds of alignments, one must determine which methods produce the best alignments consistently. The first scenario requires that alignments that are better from a biological standpoint receive better scores. The second requires that the score be uniformly interpretable over a wide range of alignment length and quality.

In recent years, much work has gone into the assessment of predictive alignment methods [117, 54, 21, 119, 19, 118] and many measures have been proposed [133, 115, 153, 120, 189, 143], but no single measure has emerged as a complete measure of alignment quality.

One of the most intuitive and popular measure is *the number of residues aligned correctly.* Yet by itself this measure has no meaning; to know if it reflects a good alignment, one must know the number of residues aligned or the length of the structural alignment. In the CASP3 assessment, S_0, the number of residues aligned correctly was reported together with both lengths: structural alignment length L_1 and predicted alignment length L_x [104]. Alignment quality assessment requires mentally absorbing all three numbers simultaneously, which might be acceptable for selecting the better predictions from a small pool of candidates but is not viable for characterizing the overall performance of an alignment method.

A related quantity is the *fraction of residues aligned correctly.* This quantity is easier to interpret, as meaningful information can be gleaned without comparing two numbers, and it has been used extensively in the literature [117, 19, 133, 182, 17, 159]. However, one must decide which alignment length to use in the denominator: the length of the predicted alignment or the length of the structural alignment. Many authors address this by reporting two values: the fractions relative to the lengths of the predicted alignment and structural alignment respectively. Fraction relative to predicted alignment and fraction relative to structural alignment have appeared with various labels: alignment specificity and alignment sensitivity [117], confidence and power [19], and F_m and F_d [159]. Yet as often noted, there tends to be a tradeoff between the two measures, with neither capturing all of the relevant information. A very short but accurate alignment will have excellent alignment specificity but poor alignment sensitivity. An alignment that aligns the correct regions accurately but aligns too much will have excellent alignment specificity but poor alignment sensitivity.

Many popular measures involve similar tradeoffs. Measures which reward short, accurate alignments include the CASP2 measures [115] *alignment specificity*, *mean RMS deviation*, and *mean* *shift error*. Measures which reward alignments that contain accurate alignments but align too much include the CASP2 measures *alignment sensitivity*, *alignment length*, and *coverage*, the number of target sequence residues aligned in the candidate alignment divided by the number aligned in the reference alignment. Alignment quality assessment often involves regarding one measure in the context of another, a process that is controversial at best [109].

Hubbard sought to diminish this controversy during CASP3 assessment by providing a visualization for RMS deviation and coverage [74]. In his analysis, he started with a very low threshold for RMS deviation, generated a superposition between portions of the predicted and actual structures which fell below the RMS deviation threshold, and measured coverage. He then increased the threshold slightly, generated a new substructure superposition, and measured coverage. This process was repeated for RMS deviation thresholds of up to 10 angstroms. This yielded a curve identifying what proportion of the prediction fell within certain accuracy tolerances. For instance, a prediction that was correct for one section and incorrect for another would show a gradual climb followed by a distinct bend and a far steeper climb. To put the curve into perspective, graphs were assembled showing the curve for all predictions on the same target, with the curve for the prediction in question highlighted. These graphs avoid the philosophical pitfall of which method to reward: one which knows its weaknesses and does not predict unless it can do so with confidence, or one which attempts the difficult portions of the problem. However, as stressed by Hubbard himself, the graphs serve as a qualitative tool for prediction assessment and do not generate a comprehensive measure of prediction ranking.

A number of studies [177, 119, 118, 19] score alignments by their accuracy on conserved motifs in the relevant protein families. These motifs contain the residues most important for the structure or function of the protein. While one can argue that these regions are the most important portion of the structure prediction, they are simply a portion. Other regions, including the more variable regions important for evolutionary analysis, are not considered. The value of such analysis depends on the objective. If the goal is to select an alignment method for prediction of functional residues, this might be the appropriate type of test. If the goal is to refine an existing alignment method, other tests would be more appropriate.

Another family of measures [120, 189] measure the importance of each position in the alignment by comparing the alignment score with the score of the best alignment that does not include that position. While this measure is excellent for reporting the importance of a position within the prediction, it does not report the position's accuracy. Further, it requires access to the alignment tool and its intermediate results.

Another factor that can cloud alignment quality measurement is the choice of parameters such as distinguished alignment or sequence. For instance, *mean shift error* is sensitive to the choice of sequence over which shift is measured. Because shift is defined for residues which are aligned to other residues in both alignments, the number of positions for which a shift measure is available will depend on the choice of distinguished sequence.

We propose a measure that includes penalties for aligning too much, aligning too little, and aligning inaccurately. This measure, referred to as the *shift score*, is based on shift error. It requires no structural information, and is easy to compute. It compares two alignments in a symmetric fashion, so no distinguished alignment or sequence is required. With a simple greedy algorithm, it can be optimized to produce an *optimal subalignment*, the subalignment that would be produced if the alignment method did not align too much.

We have applied the shift score in a number of investigations. First, we have used it to refine our methods for producing alignments from hidden Markov models [93]. This application requires a single measure that is meaningful over a wide spectrum of alignment quality and penalizes all types of alignment errors we wish to minimize. Second, we have developed a tool to predict when a method has aligned too much and to generate a trimmed alignment. Here, we have used the optimal subalignment as our prediction target.

Section 4.3 describes the calculation of the shift score, and Section 4.4 presents results describing the correlation of the shift score to the CASP2 alignment quality measures. Finally, in

Section 4.5, we display some examples of the output of our alignment comparison tool: an accompanying visualization tool that reports the shift score and depicts the shift of each residue. This tool, and the source code, are available on the World Wide Web at **http://www.cse.ucsc.edu/research/compbio/HMM-apps/compare-align.html**.

4.3 Calculation of the shift score

For some pair of sequences A and B aligned in alignments X and Y, the *shift score* is computed as follows.

$$\begin{aligned}
\epsilon &= \text{small-valued algorithmic parameter, typically set to 0.2} \\
|X| &= \text{Number of aligned residue pairs in alignment } X \\
X_i &= \text{Aligned residue pair } i \text{ in alignment } X \\
s(r_i) &= \text{Subscore for residue } r_i \\
&= \left\{ \begin{array}{ll} \frac{1+\epsilon}{1+|\text{shift}(r_i)|} - \epsilon & \text{if shift}(r_i) \text{ is defined} \\ 0 & \text{otherwise} \end{array} \right\} \\
X_i(A) &= \text{The sequence } A \text{ residue aligned in column } X_i \\
cs(X_i) &= \text{Column score for column } i \text{ in alignment } X \\
&= \left\{ \begin{array}{l} s(X_i(A)) + s(X_i(B)) \\ \text{if column } X_i \text{ aligns some residues } X_i(A) \text{ and } X_i(B) \\ \ 0 \text{ otherwise} \end{array} \right\} \\
\text{shift\_score} &= \frac{\sum_{i=1}^{|X|} cs(X_i)}{|X| + |Y|}
\end{aligned}$$

While this definition emphasizes alignment X, the score is symmetric with respect to the alignment emphasized. To see this, recall that for any residue in either sequence, shift is defined if and only if the residue aligns to some residue in both alignments. Whether or not shift is defined for any given residue is thus independent of the choice of alignment emphasized. Shift is symmetric with respect to labelling of reference and candidate alignment. If some residue is shifted by k positions under one alignment labelling, it will be shifted by $-k$ positions under the other labelling. As this measure uses only the magnitude of shift, each column score $cs(X_i)$ will be consistent across both labellings.

The $s(x_i)$ terms range from $-\epsilon$ to 1.0. They are positive for small shifts, or shifts of zero (where $s(x_i) == 1.0$). For large shifts, they approach $-\epsilon$. Table 4.1 shows the relation between $|\text{shift}(x_i)|$ and $s(x_i)$ as a function of ϵ and for $\epsilon = 0.2$. Residues shifted by four or fewer residues (approximately one turn of a helix or less) contribute to the score, while residues shifted by more than five residues decrease the score. In general, $s(x_i)$ is positive for shifts of less that $\frac{1}{\epsilon}$ and negative for shifts of greater than $\frac{1}{\epsilon}$.

The shift score is also a number between $-\epsilon$ and 1.0. If the two alignments are identical, their shift score is 1.0. If one alignment is a subalignment of the other, then all the $s(i_x)$ terms are 1 for the residue pairs aligned in the subalignment and 0 for the residue pairs excluded from the subalignment. Coverage, defined as the number of target residues aligned in the candidate alignment divided by the number aligned in the reference alignment, is related to the shift score as shown with B representing a subalignment of A:

$$\text{shift\_score} = \frac{2\,|B|}{|A| + |B|} \tag{4.1}$$

$$= \frac{2cov}{1 + cov} \tag{4.2}$$

The shift score does not depend on the assignment of target and template sequence, or reference and candidate alignment. Because the shift score includes alignment length, shift error, and coverage information, it does not need to be viewed in the context of other alignment statistics.

4.3.1 Optimizing the shift score

The shift score can be optimized by the following greedy algorithm:

$\lvert\mathrm{shift}(x_i)\rvert$	$s(x_i)$	$s(x_i)$ with $\epsilon = 0.2$
0	1	1.00
1	$\frac{1}{2} - \frac{\epsilon}{2}$	0.40
3	$\frac{1}{4} - \frac{3\epsilon}{4}$	0.10
5	$\frac{1}{6} - \frac{5\epsilon}{6}$	0.00
7	$\frac{1}{8} - \frac{7\epsilon}{8}$	-0.05
∞	$0 - \epsilon$	-0.20

Table 4.1: Illustration of the relation between absolute shift and the shift score term $s(i_x)$ as a function of ϵ and for $\epsilon = 0.2$.

```
repeat {
    old_score = shift_score(reference,candidate)
    remove from the candidate the column with the largest column score
    new_score = shift_score(reference,candidate)
} while (new_score > old_score)
reinstate the last column removed from candidate
```

This algorithm produces a version of the candidate alignment with the most inaccurate regions removed. Columns will be removed or retained according to the following points:

- Columns that are aligned accurately or with small shifts of 1 to 2 residues will always be retained.
- Columns containing residues with shifts of five or greater, or residues not aligned in the reference alignment will always be removed.
- Other columns might be removed or retained according to the overall alignment accuracy. In an accurate alignment, positions with moderate shifts of 3 or 4 residues will be removed.In a mostly inaccurate alignment, such positions will be retained.

4.4 Comparison with the CASP2 assessment measures

We compared the shift score to the following alignment quality measures used in CASP2 fold recognition assessment [117]: alignment specificity (ASpc), alignment sensitivity (ASns), alignment specificity ± two residues (ASp2), alignment specificity ± four residues (ASp4), alignment contact specificity (ACSpc), mean RMS deviation (ARms), mean shift error (Shft), and coverage (Covr). Alignment sensitivity (ASns) is defined as the number of residues aligned correctly divided by the length of the reference alignment. Alignment specificity (Aspc) is the number of residues aligned correctly divided by the length of the candidate alignment. Asp2 and Asp4 extend alignment specificity by computing the number of residues with no more than small shifts of two and four residues, respectively divided by the length of the candidate alignment. Alignment contact specificity and sensitivity are similar to alignment specificity and sensitivity, but use the number of contacts correctly predicted rather than the number of residues correctly aligned. Further detail can be found on the CASP2 automatic evaluation data web site [116].

For our comparison, we downloaded from the CASP2 web sites [116] alignments that had been submitted to the fold recognition section of the CASP2 contest. We scored these candidate alignments relative to those produced by the three structural aligners used in CASP2 assessment: DALI [69], SSAP [135], and VAST [51]. We then compared the shift score to the results in the alignment model table in the automatic evaluation data web site. When CASP2 predictors submitted more than one different alignment of some target-template prediction, the alignment model table reports one set of scores weighted by the probability the predictors assigned to each alignment. To simplify our analysis, we limited ourselves to cases where the predictors had submitted only one alignment of the target and template. We investigated 156 alignments compared to the three structural aligners for a total of 325 data points.

For the CASP2 contest, fold recognition assessment featured a myriad of alignment quality measures. Fewer evaluation measures were used in assessment of the CASP3 predictions. Most assessment focused on RMS deviation, percentage of residues aligned correctly, and the percentage aligned within four positions of correct /citeCASP3-assess-livermore. No new measures were introduced, aside from the visualization quantities LCS and GDT. The remaining measures are all very similar to measures used in the CASP3 assessment. Therefore, we did not repeat this analysis

Measure	Description	Correlation coefficient r
ASpc	Alignment specificity	0.974
Asp2	ASpc $\pm$ two residues	0.957
ASns	Alignment sensitivity	0.947
Asp4	ASpc $\pm$ four residues	0.936
ACSpc	Alignment contact specificity	0.854
ARms	Mean RMS deviation	-0.758
ACSns	Alignment contact sensitivity	0.737
Covr	% Coverage	0.397
Shft	Mean shift error	-0.346

Table 4.2: Correlation between the shift score and other alignment measures, as seen on 156 fold recognition alignments submitted to the CASP2 contest.

with the CASP3 measures.

Table 4.2 shows the correlation of the shift score to the CASP2 assessment measures. The shift score correlates very well to alignment specificity and sensitivity and correlates well to alignment contact specificity and sensitivity and to RMS deviation. Its weak correlation to coverage and mean shift error is not surprising. If an alignment is very short but accurate, its mean shift error will be low but its shift score will be small. Coverage reflects alignment length but not accuracy.

Alignment sensitivity (ASns) and alignment specificity (ASpc) are of special interest because of their importance in the CASP2 assessment [115]. Figure 4.2 details their comparison to the shift score. These scatterplots show strong diagonal lines with some outliers. One outlier circled on both graphs has values of 0.447 for shift score, 11.3 for ASns, and 19.1 for ASpc. Its values for ASp2 and ASp4 of 36.0 and 39.4 suggest that while the alignment did not align many residues correctly, many were aligned with only a small error. This accounts for the high shift score relative to both ASpc and ASns. The other circled outlier has values of 72.6 for ASns, 30.6 for ASpc, and 0.144 for a shift score. It aligned many residues correctly, but aligned far too much; the length of the candidate alignment was more than twice the length of the reference alignment. Its mean shift error of 0.144 suggests that though many residues had shifts of zero, some had very large shifts. Its low shift score results from penalties incurred by these greatly-misaligned positions, plus the large

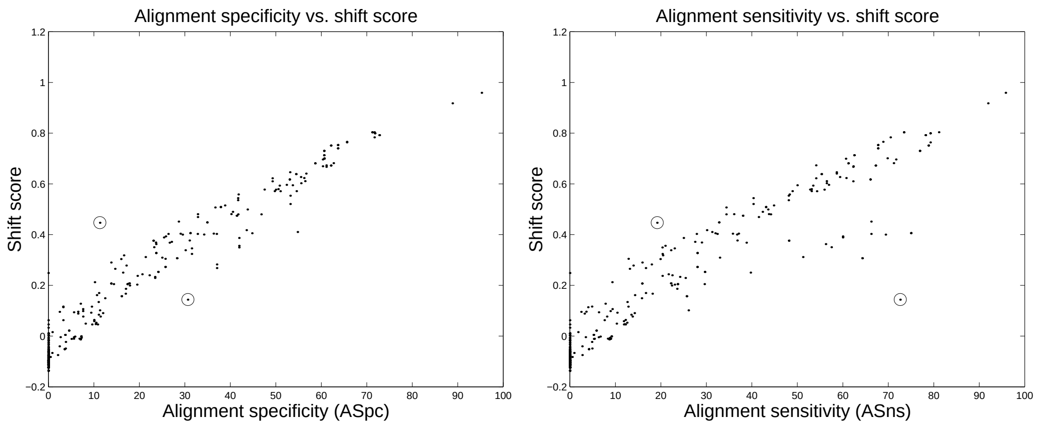


Figure 4.2: Comparison between shift score, alignment specificity (left), and alignment sensitivity (right), calculated on 156 fold recognition alignments submitted to the CASP2 contest. The points discussed in Section 4.4 are circled. Alignment sensitivity is the number of correctly aligned residues divided by the length of the reference alignment, and alignment specificity is the number of correctly aligned residues divided by the candidate alignment length.

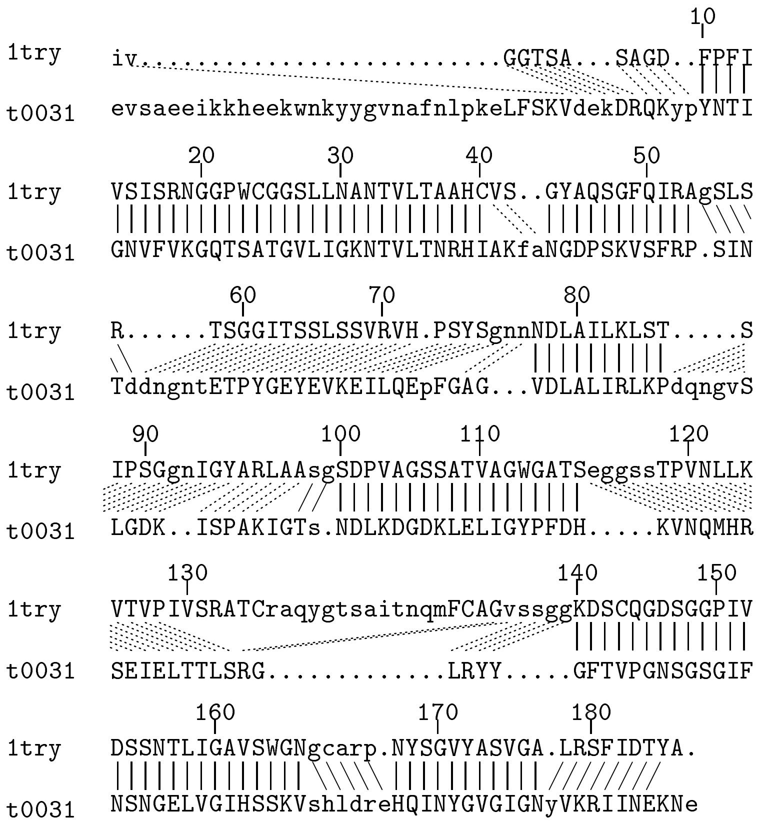
amount of over-alignment. Other outliers with lower shift scores than expected from ASns are also a result of overaligning. To sum up, for most alignments tested, ASpc, ASns, and the shift score were consistent with each other. Where they disagreed, the shift score seems to provide the better indicator of alignment quality.

4.5 Alignment comparison visualization

We conclude by showing some sample alignments, their shift scores, and their *optimal subalignments*. This alignment comparison visualization tool and its software is available on the World Wide Web [113] at **http://www.cse.ucsc.edu/research/compbio/HMM-apps/compare-align.html**.

Figures 4.3 and 4.4 compare FSSP structural alignments [69] with predicted alignments. The FSSP alignments are shown, and used as the reference alignments. The candidate alignments are not shown, but are reflected in the shift lines. All shift scores shown below are computed with $\epsilon = 0.2$.

In Figure 4.3, the CASP2 target T0031 is aligned to the structure 1try. Much of the



```
1try   iv..........................GGTSA...SAGD..FPFI
t0031  evsaeeikkheekwnkyygvnafnlpkeLFSKVdekDRQKypYNTI

1try   VSISRNGGPWCGGSLLNANTVLTAAHCVS..GYAQSGFQIRAgSLS
t0031  GNVFVKGQTSATGVLIGKNTVLTNRHIAKfaNGDPSKVSFRP.SIN

1try   R......TSGGITSSLSSVRVH.PSYSgnnNDLAILKLST.....S
t0031  TddngntETPYGEYEVKEILQEpFGAG...VDLALIRLKPdqngvS

1try   IPSGgnIGYARLAAsgSDPVAGSSATVAGWGATSeggssTPVNLLK
t0031  LGDK..ISPAKIGTs.NDLKDGDKLELIGYPFDH.....KVNQMHR

1try   VTVPIVSRATCraqygtsaitnqmFCAGvssggKDSCQGDSGGPIV
t0031  SEIELTTLSRG............LRYY.....GFTVPGNSGSGIF

1try   DSSNTLIGAVSWGNgcarp.NYSGVYASVGA.LRSFIDTYA.
t0031  NSNGELVGIHSSKVshldreHQINYGVGIGNyVKRIINEKNe
```

Figure 4.3: FSSP alignment of T0031 with 1try compared to a predicted alignment. This alignment achieves a shift score of 0.594 and obtains an optimum shift score of 0.718 by removing the 71 columns indicated with dotted lines. The large difference between the these scores is from a few positions shifted by a large number of residues.

alignment is good, as represented by the large number of shifts of zero, yet there are a few positions with shifts as large as 18. When these positions are omitted, the shift score improves substantially.

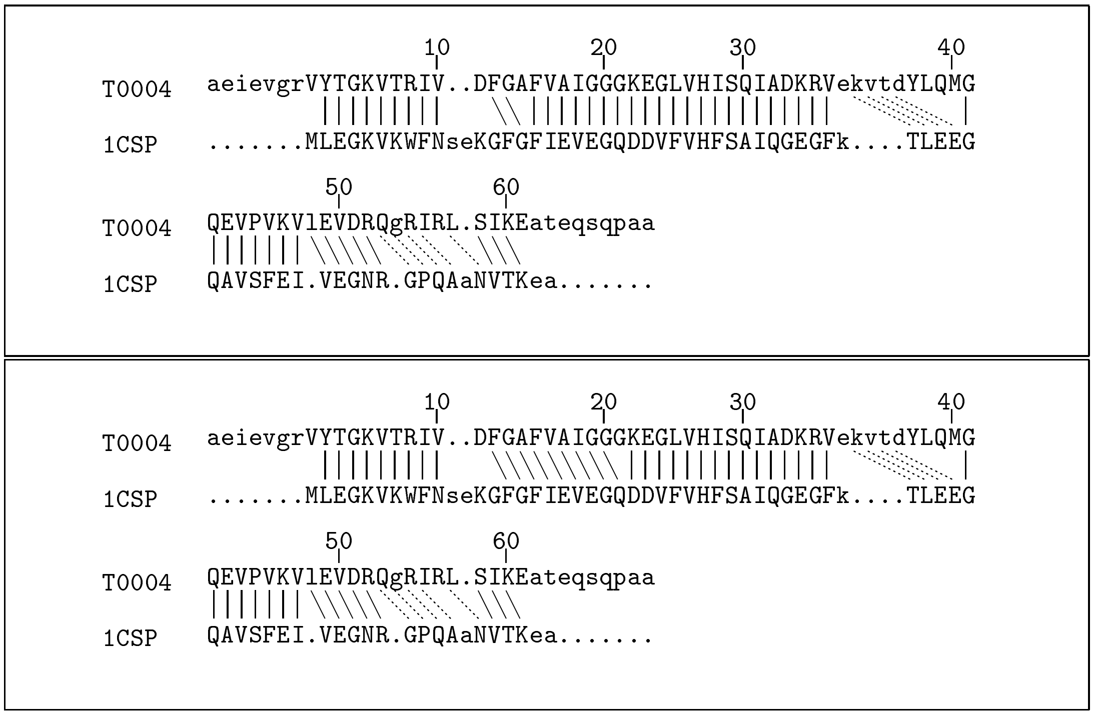

10 20 30 40
T0004 aeievgrVYTGKVTRIV..DFGAFVAIGGGKEGLVHISQIADKRVekvtdYLQMG
1CSP ....MLEGKVKWFNseKGFGFIEVEGQDDVFVHFSAIQGEGFk....TLEEG
50 60
T0004 QEVPVKVlEVDRQgRIRL.SIKEateqsqpaa
1CSP QAVSFEI.VEGNR.GPQAaNVTKea.......
10 20 30 40
T0004 aeievgrVYTGKVTRIV..DFGAFVAIGGGKEGLVHISQIADKRVekvtdYLQMG
1CSP ....MLEGKVKWFNseKGFGFIEVEGQDDVFVHFSAIQGEGFk....TLEEG
50 60
T0004 QEVPVKVlEVDRQgRIRL.SIKEateqsqpaa
1CSP QAVSFEI.VEGNR.GPQAaNVTKea.......

Figure 4.4: FSSP alignment of T0004 compared with two predicted alignments. The upper alignment achieves a shift score of 0.716, and optimum shift score of 0.761 with 11 fewer columns. The lower alignment achieves a shift score of 0.648, and an optimum shift score of 0.687 with 11 fewer columns. Dotted lines indicate the columns removed to obtain optimal subalignments.

Figure 4.4 compares two different candidate alignments. The upper alignment is slightly better and receives the higher shift score.

4.6 Issues concerning structural alignments as alignment standards

The shift score measures the quality of a predicted alignment by comparing it with a "gold standard". In some cases, curated alignments such as the BAliBASE alignment test set [181] serve as the gold standard. Sadly, curation remains a labor-intensive task, and only a small proportion of the remote homology pairs known have curated alignments available. For this reason, the more common alternative is to use structural alignments as a gold standard, as they are derived by automated methods and available for more remote homology pairs. Structural alignments are derived by comparing the three-dimensional structures of two protein sequences, and by determining an *optimal superposition*, a superposition of the two structures which minimizes their inter-structural distance by some objective function such as the atomic distance between the backbone atoms.

However, for each "best" superposition, there are typically several other superpositions which are reasonable, and score only slightly lower than the one chosen [199]. At times, two superpositions can yield very different alignments, but be very close in score. Therefore, one can say that consistency between a predicted and a structural alignment indicates a good prediction, but one cannot say that lack of consistency indicates a poor prediction.

Furthermore, the "optimal" superposition chosen by each structural alignment will depend on its objective function, and different structural aligners will tend to find different optimal superpositions [53]. For example, VAST [51] looks for a superposition that minimizes RMS deviation between superimposed segments of at least a certain length, while DALI [69] looks for long regions of no more than a certain RMS deviation. As such, DALI alignments tend to be large compared to VAST alignments, even though the algorithms are similar in other respects. In general, when two structures are very similar, most structural aligners will yield consistent alignments. However, when the structures are not so close, each different structural aligner might yield a different structural alignment [114]. This problem can be addressed by comparing each predicted alignment to structural alignments from a number of different structural aligners, as was done in CASP2 [117]. When various structural alignments differ, one should judge a predicted alignment according to the structural alignment to which it bears most similarity. Similarity to one structural alignment indicates that the predicted alignment shares some features with a significant structural superposition, according to some definition of "significant".

The first problem, that the “best” superposition is not uniquely defined, is somewhat more difficult to address. For CASP3 assessment [104], the Sippl group generated a number of different structural superpositions for each target and template structure. Each superposition produced was the most optimal of a cluster of related superpositions. Each prediction was judged according to its similarity to the best superposition, and to whichever superposition it resembled most. This solution addresses the problem effectively, but requires access to a structural aligner that will produce alternative superpositions. The Sippl aligner, in its publicly-available form, does not appear to do this. An alternative is to use a jury of structural alignments from different aligners, assume they will cover the space of reasonable superpositions, and accept a small amount of experimental error where they will not.

Still another alternative is to emphasize a single structural aligner, and accept the experimental error due to the factors described here. This can be a reasonable choice for tasks such as parameter optimization, where one is comparing two sets of similar predicted alignments. In such a case, a better score for one method indicates that the method produces alignments more similar to those produced by some structural aligner. While those structural alignments are not the only “gold standard”, they are a reasonable one. Therefore, when we have applied the shift score to optimization of HMM parameters, we have focused on comparison to alignments produced by DALI [69].

When one is comparing alignments produced by two or more distinct methods, accepting this experimental error is less reasonable. When there is less similarity between the alignment methods, one can expect less similarity between the alignments produced. Structural alignments from one tool might not be sufficient as a standard for correctness, because one algorithm might tend to produce alignments more similar to one structural aligner. For example, consider an alignment method that emphasizes short regions that can be aligned with high confidence. It would be no surprise if its predicted alignments show more similarity to structural alignments from a method such as VAST that emphasizes accuracy at the expense of length. As such, for certain investigations, we have used a small jury of structural alignments and judged each alignment according to its best score from the jury. We have used three aligners: DALI [69], VAST [51], and the Yale aligner [50]. All three aligners met the following criteria:

Availability: Either the tool itself or an extensive alignment database is available to the public free of charge.

Reputation: If some method is not accepted by the research community, results obtained with that method will not be accepted. The aligners chosen are published aligners from reputable labs.

Performance: Early structural aligners could only find similarity in structures that were very close [66], and hence would not provide useful structural superpositions when similarity is not immediately apparent. The aligners chosen can find structural similarity in distant structural homologs, and provide meaningful numbers reflecting on the significance of the structural superpositions found.

Another application for which additional sets of structural alignments are useful is identifying suspect alignment regions. If removing suspect alignment regions yields an improved shift score relative to one structural aligner, critics might say that the alignments are not improved so much as made more similar to structural alignments produced by one method. If the shift score increases relative to a variety of structural aligners, then improvement is demonstrated more convincingly.

Chapter 5

Producing better alignments with SAM and SAM-T98

Chapter 4 described the shift score, a score which represents many facets of alignment quality with a single number. This chapter describes our use of the shift score in optimizing methods for producing alignments with the SAM HMM software package [78]. Optimization is a thankless task: journal papers are not written on optimization efforts. However, if one's ultimate goal is to improve alignments, one should start with the best alignments possible.

5.1 Experimental framework

All investigations described in this chapter were designed to follow a framework of alignment for fold recognition. We assume that we are given a target sequence of unknown structure, and some template sequence thought to be homologous. Our task is then to align the template and target sequences, using some variation of the approach listed below.

1. Select some seed alignment containing the template sequence and some set of homologs.

2. Build a weighted model of the seed alignment.

3. Align the template and target sequences to the model.

The predicted alignment is then judged against a structural alignment of the same sequences.

The data used for these investigations is a set of 170 pairs of remote homologs, identified by Spencer Tu. These remote homolog pairs were chosen by the fact that a simple method such as BLAST [2] cannot recognize them as homologs, but a more sophisticated method can. The remote homolog pairs are listed in Table 5.1, along with their Z-score and the percent identity of their FSSP structural alignment [69].

For each investigation and each pair of remote homologs, two alignments were built: using the first sequence as the template and the second as the target, and using the second as the template and first as the target. When the alignment method used the structure of the template sequence in some way, we used the mean of the two shift scores as the data point for the pair. When the alignment method used no structural information, we had various choices on how to interpret the results for each pair.

1. One can randomly choose one of the two sequences as the seed. The data representing this choice is the average of the two shift scores.

2. If one has prior knowledge of which seed will work better, the corresponding data is the better of the two shift scores.

3. Most often, one bases the decision on external factors, such as identifying the assignment of target and template for which the target sequence aligns to the template model with a higher hmmscore. We simulated this process by identifying the assignment that yielded the better hmmscore, and using the shift score corresponding to that assignment.

Unless otherwise noted, we used the first approach. Here, the score reported for each remote homology pair was the average of the shift scores resulting from the two assignments of

Structure 1	Structure 2	DALI Zscore	% Identity	Structure 1	Structure 2	DALI Zscore	% Identity
1aac	2azaA	7.1	23	1aba	1grx	7.2	25
1aba	1kte	8.5	27	1aozA	1rcy	8.8	17
1arb	1svpA	8.7	10	1arb	1try	19.2	16
1arb	2sga	12.7	8	1bbt1	1pov1	10.5	18
1bbt1	1pvc1	7.1	16	1bcfA	1dat	18.9	17
1bcfA	1ryt	17.8	16	1bfg	2i1b	14.6	14
1bjmA	1hnf	9.0	17	1bjmA	1tit	7.7	11
1bjmA	1tlk	8.9	20	1cd8	1vcaA	14.7	17
1cdy	1cid	11.4	17	1cdy	1hilB	10.4	23
1cdy	1hnf	12.5	16	1cdy	1mlcB	11.5	19
1cdy	1neu	9.6	27	1cdy	1ospH	9.7	17
1cdy	1wit	7.0	14	1cdy	8fabB	12.0	21
1cnv	1ctn	14.4	14	1cnv	1nar	20.0	13
1cnv	2ebn	12.5	14	1cvl	1broA	12.1	16
1cvl	1din	8.7	16	1cvl	1ede	9.9	11
1cvl	1gpl	6.9	12	1cvl	1tca	14.9	14
1cvl	1yasA	13.7	16	1drw	1bmdA	6.7	13
1drw	1gypA	6.0	13	1drw	1xel	11.2	18
1drw	2cmd	8.6	13	1eaf	3cla	16.3	20
1ede	1din	11.8	13	1ede	1gpl	8.4	10
1ede	1tca	12.8	11	1ede	1thtA	14.9	11
1efm	1hurA	9.8	17	1fc1A	1hnf	6.1	18
1fc1A	1tit	6.0	9	1fc1A	1tlk	7.4	14
1fc1A	1vcaA	7.8	12	1forL	1hnf	8.8	11
1forL	1tlk	9.3	21	1forL	1wit	6.2	8
1frpA	1imbA	19.4	12	1gal	1aa8A	12.1	10
1gal	1fcdA	9.4	13	1gal	1gnd	7.0	8
1gal	1nhp	9.2	12	1gal	1pbe	11.8	8
1gal	1trb	10.4	10	1gal	2tmdA	7.7	16
1gal	3grs	11.4	22	1gal	3ladA	6.9	14
1gky	1kinA	7.2	16	1gky	1ukz	11.0	20
1gky	1zin	10.3	16	1gpl	1broA	9.5	11
1gpl	1yasA	8.1	9	1hbq	1bebA	13.6	19
1hbq	1epaA	14.9	22	1hbq	1mup	13.8	15
1hdcA	1lehA	6.5	13	1hdcA	1qorA	8.4	12
1hdcA	1xel	19.8	16	1hdcA	2cmd	9.9	12
1hdcA	2ohxA	8.3	19	1kay	1glcG	12.3	14
1kay	2btfA	25.2	13	1lfb	1octC	7.6	22
1lfb	1yrnB	6.3	19	1mfa	1hnf	10.3	13
1mfa	1tlk	9.7	15	1mtyB	1mhyD	24.9	11
1mucA	4enl	23.3	16	1mup	1bbpA	12	10
1nar	1ctn	19.4	16	1nar	2ebn	15.5	12
1nfkA	1nfa	6.7	22	1nhp	1aa8A	9.1	17
1nhp	1gnd	7.7	15	1nhp	1pbe	10.5	13
1nhp	3cox	8.5	17	1omp	1pot	15.7	17
1omp	1sbp	14.0	12	1prs	1amm	10.4	15
1prs	2bb2	10.6	26	1prs	4gcr	9.8	20
1psdA	1hrdA	8.3	9	1psdA	1hyhA	6.3	9
1psdA	1lehA	12.9	12	1psdA	2ohxA	9.1	11
1psdA	2pgd	8.6	13	1psdA	2tmdA	6.2	13
1ptvA	1vhrA	12.6	14	1pvc1	1bbt3	10.3	17
1pvc1	1pvc3	9	9	1sbp	1pot	22.5	14
1tca	1broA	16.3	14	1ten	1cfb	11.2	15
1ten	3hhrB	11.5	20	1thx	1kte	6.8	14
1tlk	1bec	10.6	11	1tlk	1cdy	9.9	16
1tlk	1hilB	9.4	16	1tlk	1hnf	10.0	16
1tlk	1ieaA	6.6	10	1tlk	1mlbA	11.4	16
1tlk	1mlcB	10.0	15	1tlk	1neu	10.3	19
1tlk	1ospH	8.4	15	1tlk	1ospL	8.7	17
1tlk	1tcrA	11.1	17	1tlk	1tetL	9.6	19
1tlk	8fabB	10.1	21	1ukz	1kinA	9.2	13
1xel	1aa8A	6.7	17	1xel	1bmdA	8.6	10
1xel	1cydA	17.9	13	1xel	1dhr	15.9	14
1xel	1enp	16.1	10	1xel	1eny	17.0	9
1xel	1fds	20.0	19	1xel	2cmd	9.6	14
1xyzA	1eceA	14.8	12	1xyzA	1pbgA	12.6	9
1xyzA	2myr	12.7	10	1ycc	1cyj	6.8	17
1ycc	2mtaC	6.2	14	1ytw	1vhrA	12.9	7
256bA	1cpq	6.3	6	2aaa	1eceA	10.7	9
2azaA	1plc	7.1	17	2cmd	1aa8A	6.2	21
2cmd	1cydA	8.5	12	2cmd	1dhr	8.2	8
2cmd	1fds	8.2	11	2gdm	1ash	11.5	13
2hbg	1ash	14.4	13	2mnr	4enl	21.5	16
2pgd	1dxy	7.0	17	2pgd	1gdhA	8.0	10
2phlA	1pmi	13.4	7	2por	2omf	21.5	15
2rn2	1rthA	8.6	22	2sas	2scpA	14.8	17
2sga	4ptp	12.3	19	3cox	1aa8A	10.8	12
3cox	1fcdA	7.9	12	3cox	1gal	25.6	17
3cox	1gnd	7.2	8	3cox	1pbe	11.6	11
3cox	1trb	12.7	12	3cox	2tmdA	8.0	18
3cox	3grs	8.8	17	3cox	3ladA	9.5	21
3grs	1aa8A	8.5	12	3grs	1fcdA	20.4	16
3grs	1pbe	11.4	11	3grs	2tmdA	12.2	20
3ladA	1aa8A	8.7	17	3ladA	1pbe	6.6	15
4ptp	1arb	15.9	15	8fabB	1hnf	8.0	14

Table 5.1: Pairs of remote homologs used to refine SAM and SAM-T98 alignments

template and target sequence. The shift score reported for each experiment was the average of the 170 mean values. In a few cases, we chose one of the two shift scores to represent the pair, typically the shift score of the alignment with the better hmmscore. In those cases, the second alignment was ignored, and the score reported was an average of the 170 selected scores. These cases are all carefully noted in the text.

Unless otherwise noted, all results were obtained with SAM version 3.1B, and all local alignments reflect the optional parameter setting *fimtrans = 1.0*.

5.1.1 What is a good shift score?

To put the results shown here into perspective, we should discuss the range of shift scores expected for a hard fold-recognition alignment. This is illustrated in Figure 5.1, which shows a histogram of the shift scores of fold-recognition alignments submitted to the CASP2 contest. The histogram reflects 77 alignments, most of the fold-recognition alignments for which some correct structure was predicted. These predicted alignments were compared to structural alignments produced by DALI [69], SSAP [135], and VAST [51]. Some structural alignments were not available for all targets, but I used all structural alignments available. This yielded a total of 325 data points.

As is evident in Figure 5.1, a large fraction of the alignments had a shift score of zero or less, indicating that the sequences were entirely misaligned. On the other extreme, there was one alignment that received three scores of 0.9 or greater. This alignment was actually for a homology-modeling target that proved hard to align, and fold-recognition teams were invited to submit alignments. Overall, the 325 data points had a mean shift score of 0.187. In general, for alignment problems of comparable difficulty to a CASP2 fold-recognition target, a shift score of 0.7 or greater reflects an excellent alignment, and a shift score of 0.4 or greater is quite respectable.

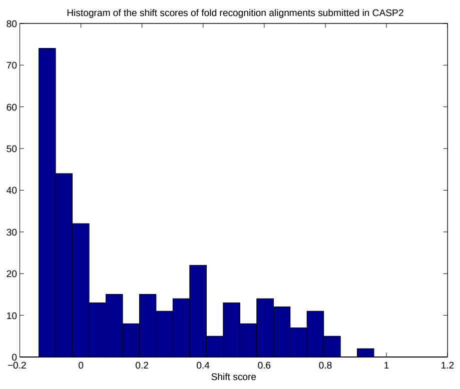


Figure 5.1: Histogram of the shift scores of 156 alignments submitted to the fold recognition section of the CASP2 contest, representing most of the alignments for which a reasonable structure was predicted. These 156 alignments were compared to up to three structural alignments for a total of 325 data points.

5.2 Selection of building method

Our first question concerns the choice of weighted build method. Sequence weighting is a vital component of any profile-based method, as sequence databases exhibit quite a bit of redundancy. In fold recognition, we have found our best results empirically by weighting according to a specific target savings over the background entropy [93]. The magnitude of the target savings indicates how much of the signal in the model is derived from the training data, rather than the priors. At the extremes, a savings of zero indicates that all of the information is derived from the priors, and a very large savings indicates that almost all of the information is derived from the training alignment. Here, we explore what target savings achieves the best alignment quality. The families of build methods tested are listed below.

Henikoff weighting: The relative weights for each sequence are set according to the Henikoff weighting scheme [62]. The sequence weights are scaled up or down to achieve a target savings. The target savings is expressed as the average number of bits saved over the columns of the alignment, relative to the entropy of the background distribution. Examples include *fh0.5*, which saves an average of half a bit relative to background, and *fh0.3*, which saves an average of three-tenths of one bit relative to background.

	Target99 seed alignment		FSSP seed alignment	
Method	Global	Local	Global	Local
fh0.3	0.323	0.364	0.398	0.394
fh0.5	0.335	0.364	0.439	0.414
fw0.3	0.321	0.361	0.421	0.396
fw0.5	0.343	0.368	**0.463**	0.409
fw0.7	0.352	0.356	0.462	0.403
fw1.0	0.354	0.337	0.458	0.399
w0.4	0.349	**0.372**	0.434	**0.419**
w0.5	0.356	0.368	0.447	0.415
w0.6	0.359	0.364	0.454	0.411
w0.8	**0.361**	0.346	0.449	0.402
w1.0	0.351	0.334	0.447	0.397

Table 5.2: Average shift score as a function of weighted build method, for both FSSP and target99 alignments, and for both global and local alignment. Within each column, the build method achieving the best score is shown in bold.

Entropy weighting: The relative sequence weights are set according to the columns of the seed alignment. The sequence weights are then scaled to achieve a fixed target savings. An example of this family is $w0.5$, which attempts to save half a bit relative to the background entropy. $Fw0.5$ is also an entropy-weighting build, and differs from $w0.5$ in its choice of column regularizer. Note that the fw builds and the fh builds use the same column regularizer.

To determine what building methods perform best, we aligned the remote homology pairs listed in Table 5.1 using eleven weighted building methods: the Henikoff weighting methods $fh0.3$ and $fh0.5$, and the entropy weighting methods $fw0.3$, $fw0.5$, $fw0.7$, $fw1.0$, $w0.4$, $w0.5$, $w0.6$, $w0.8$, and $w1.0$. Table 5.2 summarizes the results of these building methods, two choices of seed alignment, and both global and local alignment. While there is no one build that scores best in all cases, $w0.5$ performs well most consistently. Beyond that, two trends are visible. First, entropic weighting appears to perform better than Henikoff weighting. Second, the best overall setting for target savings seems to be at just under half a bit for local alignments and half a bit or slightly higher for global alignments.

		Global alignment		Local alignment	
		Shift	Optimal	Shift	Optimal
Seed	Method	Score	Subalignment	Score	Subalignment
Target99	fh0.3	0.323	0.506	0.364	0.448
Target99	fh0.5	0.335	0.505	0.364	0.444
Target99	fw0.3	0.321	0.503	0.361	0.447
Target99	fw0.5	0.343	0.514	0.368	0.451
Target99	fw0.7	0.352	0.504	0.356	0.434
Target99	fw1.0	0.354	0.482	0.337	0.407
Target99	w0.4	0.349	0.529	**0.372**	**0.463**
Target99	w0.5	0.356	**0.530**	0.368	0.457
Target99	w0.6	0.359	0.523	0.364	0.450
Target99	w0.8	**0.361**	0.506	0.346	0.421
Target99	w1.0	0.351	0.480	0.334	0.402
FSSP	fh0.3	0.398	0.574	0.394	0.473
FSSP	fh0.5	0.439	0.598	0.414	0.480
FSSP	fw0.3	0.421	0.591	0.396	0.476
FSSP	fw0.5	**0.463**	**0.614**	0.409	0.480
FSSP	fw0.7	0.462	0.608	0.403	0.467
FSSP	fw1.0	0.458	0.602	0.399	0.459
FSSP	w0.4	0.434	0.595	**0.419**	**0.495**
FSSP	w0.5	0.447	0.602	0.415	0.483
FSSP	w0.6	0.454	0.601	0.411	0.475
FSSP	w0.8	0.449	0.593	0.402	0.462
FSSP	w1.0	0.447	0.588	0.397	0.455

Table 5.3: Comparison of global and local alignment for the indicated build methods and seed alignments. Two sets of results are shown for each combination: the alignment shift score, and the shift score of the optimal subalignment. The best build for each category is shown in boldface.

5.3 Global versus local alignment

Table 5.3 summarizes results comparing global with local alignment. The table lists both the mean shift shift score and optimal subalignment score, so that we can better judge the behavior of the alignment method.

Here, the choice of alignment algorithm depends on the full experimental framework. In terms of shift score, local alignment generally performs better. However, global alignments tend to have a significantly better optimal shift score. This tells us that global alignment is more prone to overalignment – a result that makes intuitive sense – but that they contain more accurate regions than local alignments do. So, global alignment is the better choice if the objective is to accurately align as many positions as possible, or if there is a good system on hand to remove overaligned regions. But, if overall alignment quality is the objective, or if the alignment must have a minimum of inaccurate regions, local alignment is typically the better choice.

5.4 Selection of the seed alignment

Previously, I have listed results with two different seed alignments: FSSP and Target99. Here, I address the choice of seed alignment more thoroughly.

The seed alignment serves as the basis for the alignment modeling process. The target and template sequence are aligned by building a model from the seed alignment, and aligning the two sequences to the model. It would be far simpler to align the two sequences using simple dynamic programming, but we would not expect the good results because simple dynamic programing uses less information. The missing information is the homologs of the template sequence. The seed alignment, the alignment of the template sequence and its homologs, provide a much richer characterization of the sequence family than the template sequence can alone. So, the seed alignment is truly at the heart of this process. Therefore we experimented with each of the seed alignments described below.

FSSP: The homologs are the structural homologs of the template sequence as identified by DALI, excluding the target sequence. The FSSP alignment is built as a concatenation of pairwise alignments; each homolog is aligned to the template sequence by DALI Thus, the columns of an FSSP alignment represent corresponding positions in the proteins' three-dimensional structures. However, the homologs are frequently distant sequence homologs, and the sequence homology signal in FSSP alignments is weak. Compounding this issue, the FSSP database uses a somewhat generous cutoff for structural similarity. As a result, FSSP alignments include sequences that are weak homologs at best. To reduce this noise, we removed from the alignments any sequences with a DALI *Zscore* of less than 7.0.

FSSP-cheat: A FSSP-cheat alignment differs from the FSSP alignment in that it includes the target sequence. Since FSSP-cheat alignments contain the "correct answer", structural alignments of the template and target sequences, FSSP-cheat results represent an upper bound, the best one could expect to do with a set of informed homologs.

Target98: These alignments are generated by running SAM-T98 with the template sequence as a seed. Compared to FSSP alignments, these alignments tend to be larger and the homologs tend to be closer sequence homologs.

Target98-FSSP: These alignments are generated by running SAM-T98 with the template's FSSP alignment (excluding the target sequence) as a seed.

Target99: These alignments were generated by running SAM-T99, an improved version of the SAM-T98 process with the template sequence as a seed.

Template sequence: The alignment consists of the template sequence with no homologs, and reflect the performance that might be obtained with a simple pairwise method. When results obtained with these seed alignments are contrasted with others, the contrast indicates how much signal was derived from the homologs in the other seed alignment.

Table 5.4 shows results obtained with each of the seed sequences for three selected build methods. Not surprisingly, the strongest results are those obtained with the FSSP-cheat seed alignments, and the weakest are those obtained with the template sequence only. Results obtained with the template sequence only are the worst by a wide margin, indicating how little information is shared between the template and target sequences. What is more surprising is the consistency of the results. In all categories and for all weighted builds, the ranking of seed alignment according to score is the same. FSSP alignments perform best of the non-cheating methods, with Target99 seed alignments yielding the next-best results. This suggests the value of the structural information encoded in the FSSP alignment.

The take-home message here is that once the fold has been identified, the best alignments can usually be obtained by aligning the target sequence to the template's FSSP alignment.

Seed	Method	Global alignment		Local alignment	
		Shift Score	Optimal Subalignment	Shift Score	Optimal Subalignment
Template sequence	w0.4	0.144	0.323	0.141	0.160
Target98	w0.4	0.309	0.483	0.351	0.406
Target98-FSSP	w0.4	0.337	0.500	0.367	0.428
Target99	w0.4	0.349	0.529	0.372	0.463
FSSP	w0.4	**0.434**	**0.595**	**0.419**	**0.495**
FSSP-cheat	w0.4	0.698	0.823	0.706	0.801
Template sequence	w0.5	0.154	0.333	0.139	0.155
Target98	w0.5	0.315	0.478	0.347	0.401
Target98-FSSP	w0.5	0.342	0.494	0.360	0.417
Target99	w0.5	0.356	0.530	0.368	0.457
FSSP	w0.5	**0.447**	**0.602**	**0.415**	**0.483**
FSSP-cheat	w0.5	0.725	0.837	0.729	0.814
Template sequence	w0.8	0.163	0.344	0.138	0.151
Target98	w0.8	0.325	0.450	0.313	0.355
Target98-FSSP	w0.8	0.331	0.439	0.336	0.382
Target99	w0.8	0.361	0.506	0.346	0.421
FSSP	w0.8	**0.449**	**0.593**	**0.402**	**0.462**
FSSP-cheat	w0.8	0.746	0.843	0.746	0.819
Template sequence	fw0.5	0.121	0.293	0.123	0.143
Target98	fw0.5	0.307	0.467	0.339	0.389
Target98-FSSP	fw0.5	0.320	0.471	0.348	0.405
Target99	fw0.5	0.343	0.514	0.368	0.451
FSSP	fw0.5	**0.463**	**0.614**	**0.409**	**0.480**
FSSP-cheat	fw0.5	0.717	0.830	0.715	0.801

Table 5.4: Shown are results comparing the shift score and the score of the optimal subalignment for various seed alignments. For each weighted build and column, the best score of the non-cheating seeds is shown in boldface.

Direction chosen	Build	Global	Local
Random	w0.4	0.349	0.372
Better standard HMMscore	w0.4	0.357	0.408
Better reversed-sequence HMMscore	w0.4	**0.380**	**0.410**
Best of the two	w0.4	0.417	0.450
Random	w0.5	0.356	0.368
Better standard HMMscore	w0.5	0.370	**0.403**
Better reversed-sequence HMMscore	w0.5	**0.388**	0.399
Best of the two	w0.5	0.430	0.445
Random	w0.8	0.361	0.346
Better standard HMMscore	w0.8	0.391	**0.391**
Better reversed-sequence HMMscore	w0.8	**0.393**	0.390
Best of the two	w0.8	0.436	0.427
Random	fw0.5	0.343	0.368
Better standard HMMscore	fw0.5	**0.409**	0.371
Better reversed-sequence HMMscore	fw0.5	0.371	**0.410**
Best of the two	fw0.5	0.445	0.430

Table 5.5: Shift score shown as a function of method for choosing the alignment direction: choosing one direction at random, using whichever direction achieves the better standard or reversed-sequence HMM-score, and selecting the direction with the better alignment — assuming prior knowledge of that direction. For each category, the best score that does not require prior knowledge of the best direction is shown in boldface. All results were obtained with Target99 seed alignments.

When the FSSP alignment is very small, and the model might not be sufficiently general, the best alternative might be the next-best method: using a Target99 seed alignment.

5.5 Selecting alignment direction with HMMscore

For each target-template pair, their alignment can be approached in two directions: aligning the target sequence to the template sequence and homologs, or aligning the template sequence to the target sequence and homologs. This process is not symmetric, and one direction will produce a more accurate alignment. One approach is to use the alignment from whichever direction yields the better, or lower, hmmscore. Here, we study if that approach is effective.

Table 5.5 lists the results that could be expected for two building methods and four approaches for choosing the better alignment direction.

Random: This approach corresponds to selecting one direction at random. The results reflect the average of the shift scores of both directions.

Better standard HMMscore: This approach consists of comparing the score with which the target aligns to the template model to the score with which the score with which the template aligns to the target model, and selecting whichever alignment yields the higher score. The scoring method used is simple null scores adjusted by sequence length [8].

Better reversed-sequence HMMscore: This approach is the same as the last, except that rather than using the standard HMMscore, it uses the reversed-sequence HMMscore [93]. This score consists of the simple null score minus the score achieved by reversing the sequence and aligning it to the model.

Best of the two: This approach assumes prior knowledge of which direction will yield a better alignment, and selects that alignment. Results obtained with this approach represent an upper bound, the best performance that could be obtained with any selection method.

Looking at Table 5.5, we see that a significant gain in performance could be realized by selecting the better direction. Selecting a direction using either HMMscore achieves much of that gain. The choice between standard HMMscore and reversed-sequence HMMscore is less clear, as neither one yields better performance with consistency. However, either of these options generate better results than selecting a direction at random.

Note that the Table 5.5 shows results obtained for Target99 seed alignments only, and not for FSSP seed alignments. Target99 alignments require no structural information, while all FSSP alignments require that the structure of the seed sequence is known. As the structure of the target sequence is not known, it has no FSSP alignment. Hence, choosing the direction is not an option with FSSP seed alignments.

5.6 Posterior decoding

When aligning a sequence to a HMM, the standard approach is to use the *Viterbi algorithm.* In short, the Viterbi algorithm finds the best path through a model given a table representing

Seed	Build	Align.	Algorithm	Shift Score	Optimal Subalign.
Target99	w0.4	Global	Viterbi	0.349	0.529
Target99	w0.4	Local	Viterbi	0.372	0.463
Target99	w0.4	Global	`-adp4` Posterior Decoding	0.359	**0.542**
Target99	w0.4	Global	`-adp5` Posterior Decoding	0.370	0.541
Target99	w0.4	Local	`-adp5` Posterior Decoding	**0.398**	0.525
Target99	w0.5	Global	Viterbi	0.356	0.530
Target99	w0.5	Local	Viterbi	0.368	0.457
Target99	w0.5	Global	`-adp4` Posterior Decoding	0.366	**0.545**
Target99	w0.5	Global	`-adp5` Posterior Decoding	0.377	0.543
Target99	w0.5	Local	`-adp5` Posterior Decoding	**0.395**	0.513
Target99	w0.8	Global	Viterbi	0.361	0.506
Target99	w0.8	Local	Viterbi	0.346	0.421
Target99	w0.8	Global	`-adp4` Posterior Decoding	0.363	0.520
Target99	w0.8	Global	`-adp5` Posterior Decoding	0.378	**0.521**
Target99	w0.8	Local	`-adp5` Posterior Decoding	**0.386**	0.478
Target99	fw0.5	Global	Viterbi	0.343	0.514
Target99	fw0.5	Local	Viterbi	0.368	0.451
Target99	fw0.5	Global	`-adp4` Posterior Decoding	0.347	0.524
Target99	fw0.5	Global	`-adp5` Posterior Decoding	0.363	**0.525**
Target99	fw0.5	Local	`-adp5` Posterior Decoding	**0.388**	0.506
FSSP	w0.4	Global	Viterbi	0.434	0.595
FSSP	w0.4	Local	Viterbi	0.419	0.495
FSSP	w0.4	Global	`-adp4` Posterior Decoding	0.461	**0.628**
FSSP	w0.4	Global	`-adp5` Posterior Decoding	0.483	0.624
FSSP	w0.4	Local	`-adp5` Posterior Decoding	**0.486**	0.602
FSSP	w0.5	Global	Viterbi	0.447	0.602
FSSP	w0.5	Local	Viterbi	0.415	0.483
FSSP	w0.5	Global	`-adp4` Posterior Decoding	0.465	**0.624**
FSSP	w0.5	Global	`-adp5`Posterior Decoding	0.486	0.620
FSSP	w0.5	Local	`-adp5` Posterior Decoding	0.**491**	0.598
FSSP	w0.8	Global	Viterbi	0.449	0.593
FSSP	w0.8	Local	Viterbi	0.402	0.462
FSSP	w0.8	Global	`-adp4` Posterior Decoding	0.469	**0.620**
FSSP	w0.8	Global	`-adp5` Posterior Decoding	**0.488**	0.613
FSSP	w0.8	Local	`-adp5` Posterior Decoding	0.484	0.579
FSSP	fw0.5	Global	Viterbi	0.463	0.614
FSSP	fw0.5	Local	Viterbi	0.409	0.480
FSSP	fw0.5	Global	`-adp4` Posterior Decoding	0.470	**0.627**
FSSP	fw0.5	Global	`-adp5` Posterior Decoding	**0.491**	0.623
FSSP	fw0.5	Local	`-adp5` Posterior Decoding	0.487	0.592

Table 5.6: Posterior decoding on alignment accuracy: `-adp4` is transition-based posterior decoding, and `-adp5` is character-based. Best results for each category are shown in boldface.

the cost of aligning each residue in the sequence to each state in the model. The cost of aligning a residue to a state is a function of the probability of emitting the residue in that state, plus the probability of entering the state given the best path from the start of the model to that state. In its entirety, the Viterbi algorithm yields the most likely path through the model that emits all residues, whether in insert, delete, or match states.

An alternative to the Viterbi algorithm is *posterior decoding* [70]. Posterior decoding is similar to the Viterbi algorithm, except that the cost of aligning a residue to some state is a function not of the most likely path through that state, but of all paths through that state. These probabilities are computed by the *Forward-backward algorithm*.

SAM 3.1 offers two varieties of posterior decoding: *transition-based* `(-adp4)` and *character-based* `(-adp5)`. Their difference lies in how they calculate the posterior probability of each residue aligning to each node in the model. Transition-based posterior decoding reflects the alignment with the greatest probability of entering each match state, or aligning to each column. given all paths through the model. It is not defined for local alignment. Character-based posterior decoding is perhaps the more traditional. It selects each position according to the posterior probability of emitting each residue at each alignment column, or match state in the model. Both forms of posterior decoding select each position according to the posterior probability of the position, given all paths through the model. For contrast, the Viterbi algorithm selects each position according to the single most-probable path through the model.

Intuitively, posterior decoding might yield better alignments, as it reflects more information: the probability of all paths through a node, rather than the probability of one path (albeit the best path). Research in *near-optimal sequence alignment* indicates that one should not merely pay attention to the top-scoring alignment, but should also observe alternate, high-scoring alignments to find the commonalities [188]. In Table 5.6, we see that posterior decoding does improve alignment accuracy. Both character-based and transition-based posterior decoding improve the shift score and the optimal subalignment score, with the latter indicating less misalignment. However, posterior-decoded alignments are not immune to overalignment: posterior-decoded global alignments do not score as well as Viterbi local alignments.

In Table 5.6, we see that character-based posterior decoding with local alignment yields the best shift score fairly consistently, and often by a wide margin. Further, its optimal subalignment scores are far better than those for Viterbi local alignment. The gap between its shift score and optimal subalignment score is almost as large as that for a global alignment method; this method is not immune to overaligning. However, in terms of overall alignment quality, local alignment with character-based posterior decoding is the method of choice.

Global posterior-decoded alignments have slightly higher optimal subalignment scores than the other methods, indicating that they have a slightly higher number of accurate positions. The highest optimal subalignment scores are obtained with transition-based posterior decoding. However, the optimal subalignment scores for character-based posterior decoding are almost as good, and the shift scores are far better. Therefore, character-based posterior decoding is probably the best of the global methods.

5.7 Thinning the seed alignment

We have examined the choice of seed alignment, but have not further addressed the question of alignment composition. For an alignment that characterizes the family of the seed sequence, what variety of sequences makes for a useful alignment? One might think that "more is better", that adding more sequences to the alignment always makes it more informative. However, as is shown in Table 5.7, this is not exactly the case.

The results shown in Table 5.7 were obtained by thinning each seed alignment to a maximal percent homology, such that no two sequences in the alignment are identical in more than the specified number of residues. There was no preliminary sorting of the sequences in the alignment; each sequence was kept or discarded based solely on its similarity to the sequences in

Maximum %	Shift Score	
Identity	Global Alignment	Local Alignment
90%	0.320	0.326
80%	0.323	0.326
70%	0.324	0.330
60%	0.319	0.328
50%	0.312	0.322
40%	0.303	0.321
30%	0.288	0.302
20%	0.213	0.225
10%	0.160	0.142

Table 5.7: Shift score as a function of the maximum percent identity. Seed alignments were thinned to the specified maximum % identity by starting with no sequences selected, and then reading through the alignment one sequence at a time. Each sequence is compared to the sequences already selected. When some sequence compares to all sequences selected with less than the threshold percent identity, then the sequence is added to the thinned alignment. The results shown reflect Target98-HSSP alignments, alignments built by SAM-T98 and seeded with the HSSP version of the sequence. The results were obtained using SAM Version 2.2, the 1998 vintage of the `w.05` weighted build method, and the parameter setting *fimtrans = 0.0*

the thinned alignment. The target and template sequences were then aligned using a model derived from the thinned alignment.

If it were true that more sequences in an alignment always yields a more useful alignment, we would expect to see the shift score deteriorating steadily as maximum percent homology is decreased. Instead, the score peaks at approximately 70% identity. As maximum percent identity decreases from 100% (not shown), the score increases gradually to its peak value. Below that value, it decreases gradually until approximately 30%. Below 30%, it decreases rapidly.

The pattern described is evident for each of the build methods shown, and for both global and local alignment. The same pattern is also evident when the two flavors of hmmscore are used to select the alignment direction and for the optimal subalignments. These results are not shown for the sake of brevity. The consistency of this pattern suggests that the result is not an artifact of the experimental setup, but is more characteristic of alignment behavior. As for why this might be the case, one explanation is that the seed alignment should characterize the breadth of the family, and that too many close homologs skew the alignment. While sequence weighting schemes should address this skew, it appears that they cannot completely remove it.

Delete-Delete Pseudocount	Shift Score	Optimal Subalignment Score
0.4	0.350	0.396
0.5	0.349	0.395
0.6	0.349	0.396
0.7	0.349	0.396
0.8	**0.350**	**0.397**
$\boxed{0.9}$	$\boxed{0.343}$	$\boxed{0.392}$

Table 5.8: Effects on alignment accuracy of the delete-delete pseudocount in the transition regularizer. For comparison, the delete-match pseudocount was 0.0685456 and the delete-insert pseudocount was 0.391659. The values currently used are shown boxed, and the best results are shown in boldface. All results were obtained with SAM version 2.2, Target98-HSSP seed alignments, local alignment, the 1998 version of the w0.5 building method.

Match-Match Pseudocount	Shift Score	Optimal Subalignment Score
1.6	**0.346**	**0.397**
1.9	0.345	0.395
2.2	0.345	0.394
2.5	0.343	0.392
$\boxed{2.8}$	$\boxed{0.343}$	$\boxed{0.392}$
3.1	0.341	0.389
3.4	0.340	0.387
3.7	0.338	0.384
4.0	0.338	0.384

Table 5.9: Effects on alignment accuracy of the match-match pseudocount in the transition regularizer. For comparison, the match-delete pseudocount was 0.145193 and the match-insert pseudocount was 0.422453. The values currently used are shown boxed, and the best results observed are shown in boldface. Better results might have been obtained by testing lower pseudocounts, but this data suggests that the results would not improve by very much. All results were obtained with SAM version 2.2, Target98-HSSP seed alignments, local alignment, the 1998 version of the w0.5 building method.

5.8 Searching for better transition costs

No exploration of building better alignments with SAM would be complete without considering transition costs. For most systems, meaningful transition costs are notoriously hard to compute; SAM is no exception. Transition costs in SAM are calculated according to the transitions observed in the alignment plus pseudocounts given in the transition regularizer. When one sets out to modify transition costs in SAM, one modifies the transition regularizer.

All experiments in this section use a transition regularizer derived from FSSP alignments. Preliminary experiments with other regularizers showed that this one worked considerably better than most. However, the alignments produced showed a pattern of mistakes; they were overly biased

Transition Cost Scale Factor	Shift Score	Optimal Subalignment Score
x0.2	0.340	**0.395**
x0.4	0.343	0.393
x0.5	0.344	0.394
x0.6	0.343	0.392
x0.7	**0.344**	0.393
$\boxed{\text{x1.0}}$	$\boxed{0.342}$	$\boxed{0.390}$
x5.0	0.331	0.374

Table 5.10: Effects on alignment accuracy of scaling down the transition pseudocounts. The values currently used are shown in boldface. The values currently used are shown boxed, and the best results are shown in boldface. All results were obtained with SAM version 2.2, Target98-HSSP seed alignments, local alignment, the 1998 version of the w0.5 building method.

toward long sequences of match states. Starting with the FSSP-trained regularizer, I explored regularizer optimization in three directions:

1. Decreasing the cost of extending gaps by decreasing the Delete-Delete pseudocount (results shown in Table 5.8),

2. Modifying the cost of staying in match states by modifying the Match-Match pseudocount (results shown in Table 5.9),

3. Modifying the impact of the regularizer by scaling all of the pseudocounts by some constant (results shown in Table 5.9).

In all three cases, decreasing the pseudocount or pseudocounts improved alignment quality slightly. The optimal subalignment score improved as well as the shift score, indicating that the change had not made the process more prone to misalignment. However, in all three cases, the change was quite modest. The change was more pronounced whenever a different regularizer was used; in such cases, the change was always negative. For the match-match pseudocount, additional tests might have yielded better results. However, the changes observed were subtle enough that the results would probably not yield a dramatic change. Overall, these results suggest that small changes in the regularizer will probably not yield much improvement in alignment quality.

5.9 Consensus builds

All results shown up to this point have involved build methods that generate one single alignment. This section concerns *consensus* methods, methods which generate two alignments, and compare the first alignment to the second to obtain a consensus alignment. This consensus alignment is obtained by removing from the first alignment any regions that are not aligned in the second, or for which the second alignment places the residues more than a couple of positions away.

The intuition behind consensus builds is the same as that behind posterior decoding and other forms of sub-optimal alignment study [188]. It states that the alignment of some target residue is trustworthy if various alignments of the same sequences always tend to align the residue in about the same position. If the residue is not consistently aligned, or if it tends to be aligned to very different columns in different alignments, then it belongs to an alignment region that should be treated with more skepticism.

5.9.1 Methods

The consensus results shown here reflect a combination of two different alignments. The alignments are combined by using the shift score as a measure of comparison between two predicted alignments. In brief, the shift score is used to compare the two alignments, and the optimal subalignment algorithm is used to obtain the subalignment of the first that maximizes its shift score relative to the second. The precise steps are detailed as follows.

1. A first alignment is built, aligning the target sequence to the template family.

2. A second alignment is built with a slightly different method, aligning the target sequence to the template family.

3. A third alignment is generated as follows. The program `a2mtrim` is used to obtain the subalignment of the first alignment that maximizes its shift score relative to the second.

4. The program `measure_shift` computes the shift score and optimal subalignment score of this third alignment to the FSSP structural alignment of the template and target sequences.

To best understand the changes made to the first alignment to produce the third, consider the columns that are removed from an alignment to generate its optimal subalignment.

- Columns with a subscore of 0.8 or better are always retained. If a column has a subscore of 0.8 or better, the template and target residues are shifted by at most one position.
- Columns with a subscore of 0 or less are always removed. If a column has a subscore of 0 or less, either its residues are not aligned in the reference alignment or they are aligned with a shift of five or more positions.
- Columns containing residues shifted by two to four positions might be deleted if the alignments are similar, or might be retained if the alignments are very dissimilar.

For a consensus method to yield a good alignment, then it must clearly start with two alignments that are good, and at least somewhat different. We have explored the following combinations of alignment algorithms:

- Global versus local,
- Different weighted build methods,
- Posterior decoding versus Viterbi alignment algorithm, and
- For cases when the seed alignment is not built with knowledge of the seed structure, swapping the choice of homolog set. First, the target sequence is aligned to the template sequence and its homologs. Then, the template sequence is aligned to the target sequence and its homologs. A third alignment is obtained by starting with the first and removing positions not aligned consistently in the second.

With four different factors to vary, care must be taken that the results presented do not overwhelm the reader. Therefore, this section focuses on the following parameters.

5.9.2 Consensus between global and local alignment

The first consensus method to consider is the consensus between global and local alignment. Table 5.11 shows the results of such methods. Unfortunately, such a consensus does not appear to be of benefit to either global or local alignment. This is true for both Viterbi and character-based posterior decoding alignments. Apparently, global and local alignments are so dissimilar that when one limits the alignment to the positions of agreement, too much is removed.

5.9.3 Consensus between template and target family alignments

The next consensus method tested involves alignment direction: whether to align the template sequence to the target homologs or the target sequence to the template homologs. Intuitively, the consensus between the two alignments should include alignment of those conserved domains or motifs common to both families. In terms of structural homology, this consensus alignment should align those regions of common structure, and should therefore reflect the more interesting or meaningful portions of the alignment.

One alternative to taking the consensus between the template-target and target-template alignments is to take whichever alignment yields a better HMMscore. Certain protein families do not tend to yield good models for detecting certain remote homologs, especially if the new homologs are members of a new subfamily not represented in the family. This is just one reason why in certain cases, one of the two alignment directions might yield a far better alignment. As shown in Section 5.5, if one had prior knowledge of what direction would yield the better alignment, this knowledge would yield on average about a 20% improvement in alignment quality. When we use HMMscore to select the alignment direction, we realize about half of this improvement.

FSSP seed alignments cannot be used in this case; if the structure of the target sequence is not known, then there cannot be an FSSP alignment of its structural homologs.

Table 5.12 summarizes the results obtained by the consensus of the two alignment di-

Viterbi Alignments				
Seed	Build Method 1	Build Method 2	Shift Score	Optimal Subalignment
Target99	w0.5-global	None	0.356	**0.530**
Target99	w0.5-local	None	**0.368**	0.457
Target99	w0.5-global	w0.5-local	0.360	0.439
Target99	w0.5-local	w0.5-global	0.359	0.434
Target99	fw0.5-global	None	0.343	**0.514**
Target99	fw0.5-local	None	**0.368**	0.451
Target99	fw0.5-global	fw0.5-local	0.355	0.422
Target99	fw0.5-local	fw0.5-global	0.354	0.428
FSSP	w0.5-global	None	**0.447**	**0.602**
FSSP	w0.5-local	None	0.415	0.483
FSSP	w0.5-global	w0.5-local	0.416	0.467
FSSP	w0.5-local	w0.5-global	0.415	0.483
FSSP	fw0.5-global	None	**0.463**	**0.614**
FSSP	fw0.5-local	None	0.409	0.480
FSSP	fw0.5-global	fw0.5-local	0.424	0.464
FSSP	fw0.5-local	fw0.5-global	0.414	0.468

Character-based Posterior-decoded Alignments				
Seed	Build Method 1	Build Method 2	Shift Score	Optimal Subalignment
Target99	w0.5-global	None	0.377	**0.543**
Target99	w0.5-local	None	**0.395**	0.513
Target99	w0.5-global	w0.5-local	0.372	0.442
Target99	w0.5-local	w0.5-global	0.361	0.440
Target99	fw0.5-global	None	0.363	**0.525**
Target99	fw0.5-local	None	**0.388**	0.506
Target99	fw0.5-global	fw0.5-local	0.354	0.420
Target99	fw0.5-local	fw0.5-global	0.354	0.428
FSSP	w0.5-global	None	0.486	**0.620**
FSSP	w0.5-local	None	**0.491**	0.598
FSSP	w0.5-global	w0.5-local	0.431	0.469
FSSP	w0.5-local	w0.5-global	0.486	0.577
FSSP	fw0.5-global	None	**0.491**	**0.623**
FSSP	fw0.5-local	None	0.487	0.592
FSSP	fw0.5-global	fw0.5-local	0.425	0.466
FSSP	fw0.5-local	fw0.5-global	0.414	0.468

Table 5.11: The results shown are obtained by a consensus of global and local alignments for the same weighted build method.

Viterbi Alignments				
		Choice of Alignment Direction		
		Random		Better HMMscore
Build Method 1	Build Method 2	Shift Score	Optimal Subalignment	Shift Score
w0.5-global	None	0.356	**0.530**	0.388
w0.5-local	None	0.368	0.457	**0.399**
w0.5-global	w0.5 Global	**0.383**	0.425	0.373
fw0.5-global	None	0.343	**0.514**	0.371
fw0.5-local	None	0.368	0.451	**0.410**
fw0.5-global	fw0.5-global	**0.372**	0.413	0.373

Character-based Posterior-decoded Alignments				
		Choice of Alignment Direction		
		Random		Better HMMscore
Build Method 1	Build Method 2	Shift Score	Optimal Subalignment	Shift Score
w0.5-global	None	0.377	**0.543**	0.403
w0.5-local	None	0.395	0.513	**0.419**
w0.5-global	w0.5-global	**0.403**	0.449	0.403
fw0.5-global	None	0.363	**0.525**	0.390
fw0.5-local	None	**0.388**	0.506	**0.419**
fw0.5-global	fw0.5-global	0.386	0.427	0.387

Table 5.12: The results shown investigate the effects of consensus between two alignments of different direction: aligning the target sequence to the template homologs, and the template sequence to the target homologs. Results for standard global and local alignment are shown for comparison. For each template-target pair, we measured the average shift score of the two alignment directions, the average optimal subalignment score of the two directions, and the shift score for the alignment yielding the better reversed-sequence HMMscore. Results shown are the average of the results for all template-target pairs tested. The best results in each category are shown in boldface.

rections, with results on global and local alignment shown for comparison. The results reported were obtained with Target99 seed alignments. We chose to build the consensus builds with global alignment and not local alignment because of the relatively-small size of local alignments. If local alignments were further shortened, the remaining alignment might not be long enough to score well. This suspicion is reinforced in part by the results in Section 5.9.2, in which the consensus between global and local alignments does not yield an effective consensus.

The consensus alignments show significant improvement in most cases. This suggests that many of the positions aligned consistently by both alignments were accurate, and many of those not were either misaligned or overaligned positions. However, when we use HMMscore to select the better direction, the consensus method loses its edge. This is no surprise, as taking the consensus between the two directions blurs the distinctions between them. In the cases when one direction yields a reliable alignment but the other does not, throwing out all but the common positions compromises the reliable alignment.

In summary, consensus on alignment direction does not necessarily yield the best alignment. If one alignment is suggested to be far better than the other, according to its HMMscore, this better alignment should be used, and will not be improved by removing positions not in agreement in the other alignment. If there is no suggestion that one alignment is much better than the other, then the consensus can yield a significant improvement.

5.9.4 Consensus between posterior decoding and Viterbi

The next consensus explored is that between Viterbi and posterior-decoded alignments of the same sequence, using the same weighted builds. For this investigation, we took alignments produced by character-based posterior decoding, and by the Viterbi algorithm. We removed from the posterior-decoded alignment the positions not in close agreement in the Viterbi alignment and vise versa. Table 5.13 shows the results of this consensus, and compares the accuracy of the consensus alignments with the accuracy of the Viterbi and posterior-decoded alignments.

Seed	Build Method	Alignment Algorithm 1	Alignment Algorithm 2	Shift Score	Optimal Subalignment
Target99	w0.5	Viterbi	None	0.356	0.530
Target99	w0.5	Posterior Decoding	None	**0.377**	**0.543**
Target99	w0.5	Posterior Decoding	Viterbi	0.351	0.454
Target99	w0.5	Viterbi	Posterior Decoding	0.351	0.454
Target99	fw0.5	Viterbi	None	0.343	0.514
Target99	fw0.5	Posterior Decoding	None	**0.363**	**0.525**
Target99	fw0.5	Posterior Decoding	Viterbi	0.348	0.452
Target99	fw0.5	Viterbi	Posterior Decoding	0.347	0.451
FSSP	w0.5	Viterbi	None	0.447	0.602
FSSP	w0.5	Posterior Decoding	None	0.486	**0.620**
FSSP	w0.5	Posterior Decoding	Viterbi	**0.492**	0.579
FSSP	w0.5	Viterbi	Posterior Decoding	0.491	0.578
FSSP	fw0.5	Viterbi	None	0.463	0.614
FSSP	fw0.5	Posterior Decoding	None	0.491	**0.623**
FSSP	fw0.5	Posterior Decoding	Viterbi	**0.499**	0.589
FSSP	fw0.5	Viterbi	Posterior Decoding	0.498	0.588

Table 5.13: The results shown are obtained by a consensus of a global Viterbi alignment and a global character-based posterior-decoded alignment of the same sequences, using the same weighted build method. The best results in each category are shown in boldface.

This consensus does not yield improvement consistently. In FSSP alignments, the consensus alignment is of slightly better quality than the posterior-decoded alignment. In Target99 alignments, this combination compromises the posterior-decoded alignment, the better of the two.

5.9.5 Consensus between different weighted builds

The final consensus experiment investigated is the consensus between different weighted build algorithms. In these experiments, two global alignments are built using different weighted builds, and positions from the first alignment are removed if they are not in close agreement with the second alignment.

Table 5.14 reports the results of consensus builds for both Viterbi and character-based posterior-decoded alignments. In most cases, the consensus between the two global alignments results in an alignment of somewhat higher quality than either the global or local alignments of the same sequences. When the consensus alignment does not represent an improvement over the local alignment, it is not much worse.

Viterbi Alignments				
Seed	Build Method 1	Build Method 2	Shift Score	Optimal Subalignment
Target99	w0.5-global	None	0.356	**0.530**
Target99	w0.5-local	None	**0.368**	0.457
Target99	w0.5-global	fw0.5-global	0.341	0.442
Target99	fw0.5-global	None	0.343	0.514
Target99	fw0.5-local	None	0.368	0.451
Target99	fw0.5-global	w0.5-global	**0.373**	0.490
FSSP	w0.5-global	None	0.447	**0.602**
FSSP	w0.5-local	None	0.415	0.483
FSSP	w0.5-global	fw0.5-global	**0.484**	0.574
FSSP	fw0.5-global	None	0.463	0.614
FSSP	fw0.5-local	None	0.409	0.480
FSSP	fw0.5-global	w0.5-global	**0.487**	0.578

Character-Based Posterior-decoded Alignments				
Seed	Build Method 1	Build Method 2	Shift Score	Optimal Subalignment
Target99	w0.5-global	None	0.377	**0.543**
Target99	w0.5-local	None	**0.395**	0.513
Target99	w0.5-global	fw0.5-global	0.389	0.504
Target99	fw0.5-global	None	0.363	**0.525**
Target99	fw0.5-local	None	0.388	0.506
Target99	fw0.5-global	w0.5-global	**0.390**	0.495
FSSP	w0.5-global	None	0.486	**0.620**
FSSP	w0.5-local	None	0.491	0.598
FSSP	w0.5-global	fw0.5-global	**0.503**	0.578
FSSP	fw0.5-global	None	0.491	**0.623**
FSSP	fw0.5-local	None	0.487	0.592
FSSP	fw0.5-global	w0.5-global	**0.501**	0.592

Table 5.14: Results obtained by taking the consensus positions between alignments computed with different weighted build algorithms. Simple global and local alignment results are shown for comparison, and the best results in each category are shown in boldface.

Note that the `w0.5` and `fw0.5` weighted builds use the same weighting scheme, entropic weighting with an average savings per column of half of one bit, but use slightly different column regularizers. These different regularizers will yield slightly different amino acid posteriors at each alignment column, which in turn will yield slightly different alignments. However, both alignments will be built using a comparable balance of information between the training sequences and the priors. Different weighting schemes will also affect the alignment by yielding different amino acid posteriors. In contrast, different weighting schemes will yield different emphasis on the training alignment relative to the priors. The question of what weighting schemes will combine well is interesting, but is a question that we have not addressed here in interest of brevity.

5.9.6 Summary

When one compares different alignments of the same sequences, as generated by a variety of reliable but different methods, certain positions will be fairly common among most alignments. As has been observed in study of near-optimal alignments, positions common to many plausible alignments are more likely to be accurate than positions only aligned in a few. Following on this idea, we have explored the quality of *consensus alignments*:, alignments obtained by taking two different alignments, and removing all positions that are not in close agreement in both.

Obviously, if the consensus alignment is going to be of good quality, the two alignments one starts with must be of good quality themselves. If one starts with a good alignment, compares it with a poor alignment, and removes the positions that are not in close agreement, one will turn the good alignment into the poor one, in part. Therefore, one should not bother with consensus alignments unless both alignments used are likely to be of good quality themselves.

We explored various combinations of alignment parameters: global and local alignment, posterior decoding and Viterbi, different weighted builds, and different alignment *directions:* whether one aligns the target sequence to the template homologs or vise versa. The most successful combination tested was different weighted builds.

Taking the consensus between two alignments built with different directions yields an improvement in overall alignment score. However, there is a better application of alignment direction information. Certain alignment directions work better than others, and HMMscore yields considerable information regarding which direction will be better. When one takes the consensus between alignments built with different alignment direction, one blurs the distinction between the better and worse direction, and HMMscore ceases to be such an effective indicator of the better direction.

5.10 Putting it all together

After studying numerous alignment algorithm parameters and the quality of the alignments produced, we shall close by summarizing what combination of factors yielded alignments with the best quality overall. Here, we present the five best methods in three categories: best overall alignment with FSSP seeds, best overall alignment with Target99 seeds and the alignment direction chosen at random, and best overall alignment with Target99 seeds and the alignment direction chosen by reversed-sequence HMMscore.

Table 5.15 shows the five best methods for aligning with FSSP seed alignments. All five are a consensus of global alignments. Most involve two different weighted builds, and most start with a posterior-decoded alignment.

Table 5.16 shows the five best methods for aligning to Target99 seeds and selecting the alignment direction at random. Every one of the five involves posterior decoding. The best two, and three of the five, involve a consensus on the alignment direction. However, a simple posterior-decoded local alignment build is also among the top five, and scores about as well as the more complex methods.

Table 5.17 shows the five best methods on Target99 seed alignments when the alignment direction is chosen on the basis of reversed-sequence HMMscore. Once again, every one of the

Rank	Shift Score	Description
1	0.503	Consensus between posterior-decoded w0.5 global and posterior-decoded fw0.5 global
2	0.501	Consensus between posterior-decoded fw0.5 global and posterior-decoded w0.5 global
3	0.499	Consensus between posterior-decoded fw0.5 global and fw0.5 global Viterbi
4	0.499	Consensus between posterior-decoded w0.5 global and fw0.5 global Viterbi
5	0.498	Consensus between w0.5 global Viterbi and posterior-decoded fw0.5

Table 5.15: For FSSP seed alignments, this table describes the five methods that yielded the best overall shift score.

Rank	Shift Score	Description
1	0.403	Consensus between a posterior-decoded w0.5 global alignment of the target sequence to the template family and a posterior-decoded w0.5 global alignment of the template sequence to the target family
2	0.395	Consensus between a posterior-decoded w0.5 global alignment of the target sequence to the template family and a Viterbi w0.5 global alignment of the template sequence to the target family
3	0.395	A posterior-decoded w0.5 local alignment
4	0.394	Consensus between a posterior-decoded fw0.5 local alignment and a Viterbi w0.5 global alignment
5	0.393	Consensus between a posterior-decoded w0.5 global alignment of the target sequence to the template family and a posterior-decoded fw0.5 global alignment of the template sequence to the target family

Table 5.16: For Target99 seed alignments, this table describes the five methods that yielded the best overall shift score when the alignment direction was chosen at random.

Rank	Shift Score	Description
1	0.422	Consensus between a posterior-decoded fw0.5 global alignment and a posterior-decoded w0.5 local alignment
2	0.421	A posterior-decoded w0.5 local alignment
3	0.420	Consensus between a posterior-decoded fw0.5 local alignment and a Viterbi w0.5 global alignment
4	0.419	A posterior-decoded fw0.5 local alignment
5	0.416	Consensus between a posterior-decoded fw0.5 global alignment and a posterior-decoded w0.5 global alignment

Table 5.17: For Target99 seed alignments, this table describes the five methods that yielded the best shift score when the alignment direction was chosen according to the reversed-sequence HMMscore.

five best methods involves posterior decoding. The best method involves the consensus between two global posterior-decoded alignments built with different weighting schemes. However, simpler posterior-decoded local alignment scores nearly as well, and occupies two slots in the top five.

In summary, the recipe for the best alignments is as follows. If there is an FSSP alignment of the template sequence and its structural homologs, that should be used as the seed alignment. The final alignment should be a consensus of posterior-decoded global alignments built with different weighted builds.

If the template sequence has no FSSP alignment, then the alignment should be built with Target99 seed alignments, local posterior-decoding, and with the HMMscore selecting the alignment direction if possible. If obtaining the HMMscores is not possible, or if the two HMMscores are close, then a slight improvement might be realized by building a global posterior-decoded alignment of the template to the target family, building a global posterior-decoded alignment of the target to the template family, and selecting the positions in close agreement in the two alignments.

Chapter 6

Comparing two profile-based alignment systems on hard remote homologs

This chapter concerns profile-based methods for sequence alignment. As described in Chapter 3, these methods add homologs to an alignment by generating a profile from the columns of the alignment and then aligning the new homologs to the profile. Such methods are computationally-efficient, and can be quite sensitive. As the protein sequence databases grow, and as libraries of protein family alignments become more complete and representative, profile methods are emerging as a strong method for sequence alignment.

This chapter focuses on the task of aligning a target sequence to an alignment of template sequence and homologs, once a fold prediction has been made. At first glance, this action might seem unnecessary, particularly if the fold recognition method produces an alignment anyway. However, there are three good reasons to estimate this alignment after fold prediction, even if an alignment has already been estimated.

1. There are algorithms that yield more accurate alignments and algorithms that make more efficient use of computing resources, and they are usually not the same. During fold recognition, when thousands of potential matches between target sequence and template structure are tested, the alignment algorithm must be efficient. After the fold has been identified with an efficient method, the alignment can be re-estimated with a more precise method.

2. As crazy as it sounds, there are methods that can accurately identify folds but do not yield alignments, or anything from which alignments can be inferred [36]. Similarly, there are confounding instances of algorithms identifying the correct fold but producing an alignment that is entirely wrong [154].

3. A model intended for fold recognition should provide a general description of the fold. Once the fold has been identified, a more specific model can be better for aligning the target sequence to particular features of the template family. For instance, SAM-T98 fold recognition works best with the `w0.5` weighting scheme, but as shown in Chapter 5, the `w0.8` weighting scheme can yield more accurate alignments. The difference between the two schemes is that in `w0.8`, a greater proportion of the signal comes from the training alignment, rather than the priors.

After the fold has been detected, how does one obtain the most accurate alignment possible? The answer to that question would merit one or more Ph.D. dissertations in itself. Rather than address the entire question, we have addressed one small part by comparing two profile alignment systems: SAM [78], and the profile alignment module in CLUSTALW [180, 82]. For the larger question, the related literature is reviewed in Section 3.2.4.

By the time one has a fold predicted for a target sequence, one has an arsenal of information available that could be useful for aligning it with the template sequence. This information includes

- an alignment of the template sequence and its sequence homologs,

- an alignment of the target sequence and its sequence homologs,
- an alignment of the template sequence and its structural homologs, and
- secondary structure of the template sequence.

To build the best alignment, the obvious approach would seem to be using everything available. However, as with any modeling process, one should not automatically use a complex model when a simple model would work just as well. Therefore, we have set out to assess the value of each of the above classes of information to the alignment process, using SAM and CLUSTALW as a jury of two.

6.1 Methods

The objective of this work was to compare alignment methods in terms of the accuracy with which they align difficult remote homologs, where the level of difficulty is approximately that of a CASP3 fold recognition target.

Each pair of remote homologs was aligned as follows. First, one was assigned as the template sequence, and the other the target. All information pertaining to the structure of the target sequence was off-limits during the alignment process. Second, the template and target sequences were aligned by SAM and by CLUSTALW. In most cases, alignments were generated by estimating a profile from a seed alignment of the template sequence plus homologs and then aligning the template and target sequences to the profile. There were some variations on this step, as I shall discuss in the next paragraph. Third, the accuracy of the alignment was measured by extracting the pairwise alignment of the template and target sequences and by comparing that predicted alignment to each of three structural alignments of the same sequences.

We explored a number of variations on methods for aligning the template and target sequences. First, we varied the alignment used for estimating the profile. We experimented with alignments of the template sequence and structural homologs, alignments of the template sequence and sequence homologs, and "alignments" consisting of the template sequence only. To estimate how much information came from the alignment rather than from the aligned sequences, we used CLUSTALW to estimate a multiple alignment of the template sequence, target sequence, and template sequence homologs.

6.1.1 Selection of the remote homology pairs

We selected 200 remote homology pairs according to the following criteria.

- The three structural aligners were able to find a significant superposition of the two structures, according to each author's definition of significance:
  - DALI zscore ≥ 7.0,
  - VAST p-value ≤ 0.0001, and
  - Yale RMSD ≤ 4.0.
- The sequences were sufficiently dissimilar that a simple FASTA [148] pairwise alignment yielded a shift score of 0.4 or worse.

No deliberate steps were taken to ensure that the pair had at least a threshold level of similarity. However, the requirement of a significant superposition by all three structural aligners weeded out many more distant pairs than close ones. The 200 pairs ranged in difficulty from 3% to 24% identical, and the set was approximately 12% identical on average.

As both sequences in each pair were of known structure, each pair represented two different assignments of template and target sequences, yielding a total of 400 tests. Of these 200 pairs, 130 were used for optimizing the methods and 70 pairs, or 140 tests, were used for final assessment of the results. Table 6.1 lists the remote homology pairs in the optimization and test sets. To indicate the ranges of structural similarity and sequence homology of these pairs, Table 6.1 lists the percent identity of their structural alignment and their zscore according to DALI. The parallel quantities for the VAST and Yale aligners were not listed in the interest of brevity.

6.1.2 Selection of the template sequence alignments

Two types of sequence alignments were used in this investigation: structure-based alignments and sequence-based alignments.

The structure-based alignments were extracted from FSSP [67], a database of multiple alignments of structures superimposed by DALI. We included only those sequences which DALI superimposed with a zscore of 7.0 or better. For each template-target pair, we removed the target sequence from the FSSP alignment of the template sequence and structural homologs. Note that FSSP alignments consist of a concatenation of pairwise alignments generated by DALI, and no pairwise alignment influences the alignment of any other pair of sequences. Therefore, one can remove the target sequence to obtain an objective alignment of the template sequence and its remaining structural homologs.

Regarding the selection of sequence-based alignments, recall that our first objective was to compare the profile alignment methods in SAM and CLUSTALW. Therefore, there had to be sequence-based alignments built by an impartial third method. We used HSSP alignments [160]. For further impartiality, we re-estimated the HSSP alignments with both SAM and CLUSTALW. However, results on the re-estimated alignments proved to be consistent with those on the original alignment. In the interest of brevity, we did not report results obtained with the re-estimated alignments. Finally, because HSSP alignments tend to be small, we also used SAM-T99 alignments: alignments built by the 1999 version of SAM-T98 [93].

6.1.3 Assessment of alignment quality

Alignment accuracy was measured by comparison with structural alignments produced by three structural aligners: DALI [69], VAST [51], and the Yale aligner [50]. Section 4.6 describes

Optimization set							
Structure 1	Structure 2	DALI Zscore	% Identity	Structure 1	Structure 2	DALI Zscore	% Identity
1a0tP	1prn	11.9	8	1a34A	1stmA	10.1	7
1abrB	1bfg	15.1	9	1ad4A	1nsj	13.8	9
1ad4A	1dhpA	13.1	10	1ad4A	1ak5	9.9	13
1ad4A	1gowA	9.9	8	1ad4A	1dosA	9.3	9
1ad4A	1frb	9.2	9	1ad4A	1cnv	9.0	8
1ad4A	1onrA	8.7	15	1adjA	1pysA	8.2	22
1ae9A	1aihA	11.6	15	1agjA	1arb	17.4	15
1air	1idk	29.5	21	1ak5	1cnv	9.0	9
1ak5	1dhpA	7.1	13	1ak5	1onrA	7.1	8
1ako	3dni	19.1	15	1al3	1sbp	11.9	10
1amk	1dhpA	11.1	10	1amk	1dosA	9.2	9
1amk	1onrA	7.4	11	1amp	1cg2A	25.3	18
1amp	2ctc	12.3	12	1amy	2aaa	26.3	18
1amy	1pamA	18.2	21	1amy	1edg	9.3	8
1aozA	1nif	21.2	17	1aq0A	1edg	16.1	12
1aq0A	1eceA	14.2	9	1aq0A	2myr	12.4	12
1aq0A	1gowA	12.1	13	1aq0A	1frb	10.4	7
1aq0A	1dhpA	7.4	8	1arb	1havA	13.6	11
1aszB	1pysA	12.9	16	1at0	1vdeA	12.5	14
1auoA	1broA	16.4	19	1auoA	1cex	12.5	11
1auyA	1stmA	8.6	12	1ax4A	2dkb	19.0	11
1ax4A	2gsaA	17.3	13	1bco	1itg	12.1	12
1bco	1vsd	11.9	12	1bcpA	1lt3A	10.5	24
1bcpB	1bovA	7.0	8	1bdb	1wab	7.1	8
1bfg	1wba	8.6	6	1bmtA	3chy	10.5	11
1bovA	1lt5D	7.6	5	1broA	1ede	25.7	16
1broA	1din	15.4	17	1broA	1cex	10.2	13
1btkA	1btn	9.5	10	1btkA	1irsA	9.4	10
1btl	1pmd	18.0	11	1btl	2bltA	15.3	13
1btn	1mai	8.9	9	1btn	1irsA	8.5	13
1byb	1edg	15.9	9	1byb	1nsj	10.4	7
1cem	1gai	19.8	9	1ceo	2myr	19.7	13
1ceo	1gowA	18.8	11	1ceo	1aq0A	17.0	11
1ceo	1xyzA	16.0	9	1ceo	1dhpA	10.6	6
1ceo	1frb	9.3	6	1cex	1tca	10.8	10
1cex	1din	10.1	10	1cnv	2myr	12.4	9
1cnv	1dhpA	8.9	8	1ctn	2ebn	17.4	13
1dbqA	1pea	12.3	10	1dhpA	1nsj	10.7	9
1dhpA	2ebn	9.2	8	1dhpA	1dosA	8.8	12
1dhr	1eny	19.3	10	1din	1tca	12.7	11
1djxB	1rlw	15.6	16	1dorA	1ak5	16.7	11
1dorA	1dhpA	9.1	11	1dosA	1nsj	7.5	11
1dpgA	1ofgA	19.0	11	1dynA	1irsA	9.5	10
1dynA	1mai	8.4	10	1eceA	1edg	24.0	16
1eceA	1nar	11.9	9	1edg	2myr	17.4	8
1edg	1gowA	16.7	15	1edg	1xyzA	13.3	9
1esc	1wab	13.0	19	1fkx	1nsj	7.3	8
1frb	1igs	10.5	11	1gca	1pea	16.1	11
1gcb	1ppn	13.3	20	1gggA	1pda	8.8	16
1gnwA	1pgtA	17.7	16	1gowA	1xyzA	12.5	8

Test set							
Structure 1	Structure 2	DALI Zscore	% Identity	Structure 1	Structure 2	DALI Zscore	% Identity
1hnf	1neu	8.4	21	1hrdA	1lehA	32.3	20
1hyhA	1eny	7.1	11	1iae	1kuh	11.3	13
1idk	1rmg	11.2	12	1igs	1nar	7.8	5
1ihp	1rpa	25.4	19	1irsA	1mai	8.3	11
1jxpA	1agjA	11.6	15	1kuh	1sat	11.0	19
1lam	2ctc	10.6	5	1lucA	1cnv	10.7	8
1lucA	1dhpA	10.2	7	1lucA	1frb	10.1	7
1lucA	1xyzA	9.3	8	1lucA	1dosA	7.6	8
1lxa	1thjA	14.6	18	1lylA	1pysA	9.1	20
1mrp	1sbp	13.3	14	1nar	2myr	11.6	6
1ndh	2pia	18.4	16	1nsj	4xis	7.9	12
1nwpA	1rcy	9.3	24	1oatA	1ax4A	18.6	15
1onrA	1nsj	9.9	8	1onrA	1eceA	8.3	9
1opy	1ounA	14.3	8	1opy	1std	11.4	9
1pbgA	1edg	18.9	12	1pbgA	1cnv	12.7	8
1pbgA	1aq0A	12.6	10	1pbgA	1dhpA	9.5	4
1pea	2dri	17.8	11	1pea	1tlfA	17.1	8
1pea	8abp	15.1	10	1plq	2polA	18.5	12
1pov1	2mev1	10.5	20	1prn	2omf	18.2	14
1prtF	1tiiD	9.5	9	1prtF	3ullA	7.9	16
1reqB	1amk	8.5	7	1rlw	1rsy	12.3	21
1scuA	3chy	8.5	18	1sesA	1adjA	17.7	14
1smvA	2mev1	7.8	11	1tdtA	1lxa	16.9	15
1tdtA	1thjA	12.7	21	1thtA	1broA	16.3	12
1thtA	1din	15.1	13	1tlfA	8abp	25.5	15
1v39	1vid	9.0	9	1vhiA	2bopA	7.3	3
1vid	1yub	9.7	9	1wba	2ila	9.5	6
1wod	1sbp	23.4	15	1wod	1pot	16.6	15
1wod	1al3	15.1	10	1wod	1mrp	8.6	13
1xjo	2ctc	11.3	8	2i1b	1wba	10.0	12
2tysA	1dhpA	11.4	14	2tysA	1amk	11.1	12
2tysA	1dosA	11.0	8	2tysA	1eceA	11.0	10
2tysA	1ak5	9.8	12	2tysA	1edg	7.7	8
2tysA	1nsj	7.0	9	3pte	1btl	17.0	17
4mbp	1mrp	16.9	13	4mbp	1sbp	13.6	11

Table 6.1: Remote homology pairs used for optimization and testing

our reasons for using three structural aligners and the selection of the aligners that were used.

For each comparison between predicted alignment and structural alignment, we measured three quantities: the shift score, the number of residue pairs aligned correctly as a fraction of the number of pairs aligned (*alignment specificity*), and the number of residue pairs aligned correctly as a fraction of the number of correct pairs (*alignment sensitivity*). Although the shift score is highly correlated to both alignment sensitivity and specificity, we report alignment specificity and sensitivity because they are more familiar quantities to many readers.

For each predicted alignment, we obtained scores relative to four sets of structural alignments: DALI, VAST, Yale, and *closest:* whichever one of the three structural alignments the predicted alignment was most similar to. This quantity merits extra columns in a few tables because each structural alignment represents one significant superposition of the template and target sequences, and when one asks if a predicted alignment is good, one is asking if it is similar to some significant structural superposition. The shift score was used to select the closest alignment.

6.1.4 Description of the alignment methods

The results shown here were obtained with SAM version 3.1b and CLUSTALW version 1.8, the command line version of CLUSTALX [82].

The alignment procedure used with SAM was as follows. Given a template family alignment, we used the `w0.5` weighted build procedure to estimate a profile HMM from the alignment. We then aligned the template and target sequences to the model with the `align2model` program, yielding a pairwise alignment of the template and target sequences. `Align2model` was used with both the default Viterbi algorithm and with character-based posterior decoding (`-adp5`). For simplicity in comparing the results to CLUSTALW, we used global alignment.

The alignment procedure used with CLUSTALW was as follows. Given a template family alignment, the `-profile` option in CLUSTALW was used to generate a profile from the template family alignment, and to align the template and target sequences to that profile. CLUSTALW profiles were generated with and without secondary structure information. When secondary structure information was used in the profile, the secondary structure provided was the DSSP [88] secondary structure of the template sequence. α-helices and 3-10 helices were designated as helix, β-strands and β-bridges were designated as strand, and everything else was designated as loop. When multiple alignments were re-estimated with CLUSTALW, they were re-estimated with CLUSTALW's progressive alignment module, its default and most familiar module.

Both methods were optimized for alignment of hard remote homologs. The optimization of SAM was described in Chapter 5. Optimization of CLUSTALW was performed empirically, using the 130 remote homology pairs in the optimization set, and using the shift score of the closest structural alignment as an objective function. Separate optimizations were performed for CLUSTALW progressive alignment and CLUSTALW profile alignment. For CLUSTALW profile alignment, four separate optimizations were performed: for structure-based and sequence-based alignments, and with and without secondary structure masks. Thus, a total of five CLUSTALW optimizations were performed, representing the largest body of work behind this effort. The final optimized parameter settings were as follows.

- The pairwise gap extension parameter was decreased from 0.10 to 0.09 for progressive alignments.

- The gap extension parameter was decreased from 0.20 to 0.18 for progressive alignment and for three of the four cases of profile alignment. For profile alignment involving structure-based seed alignments and no secondary structure information, the gap extension parameter was decreased to 0.14.

- For profile structure-based alignments without secondary structure masks, the gap distance threshold was set to 3 and the hydrophilic gap penalty option was turned off.

- For profiles with secondary structure masks and sequence-based alignments, the strand gap penalty was increased from 4 to 5 with the terminal regions of the strands set to the first two positions at the start and no positions at the end of the strands.

- For profiles with secondary structure masks and structure-based alignments, the helix gap penalty was decreased to 2.
- Default parameters were used in all other cases.

Note that the tests described here represent very difficult tests for both methods. However, such investigations are worthwhile for two reasons. First, a method that can perform well on the hard cases can probably perform well on the easier cases, give or take minor adjustments in parameter settings. The converse is not true. Second, when one is using a method, one should have an intuition for its limitations. This requires determining the point at which the method might lose its effectiveness.

6.2 Results

6.2.1 How similar are the three different structural alignments?

When we compare predicted alignments to three structural alignments rather than one, are we really gathering new information? Alternatively, are the three structural alignments so similar that anything past the first comparison is redundant? These questions are worth asking before one goes to extra trouble to compare predicted alignments to not one but three sets of structural alignments.

We used the shift score to compare the three structural alignments for the 200 pairs of structures in the dataset. Figure 6.1 shows histograms of the shift scores comparing the DALI and VAST alignments, the DALI and Yale alignments, and the VAST and Yale alignments.

If two alignments are identical, their shift score is 1.0. If two alignments are completely different, their shift score is less than zero. In Figure 6.1, we see that for each pair of aligners, a large population of pairs has a shift score of 0.65 or better. This is the range in which an strong

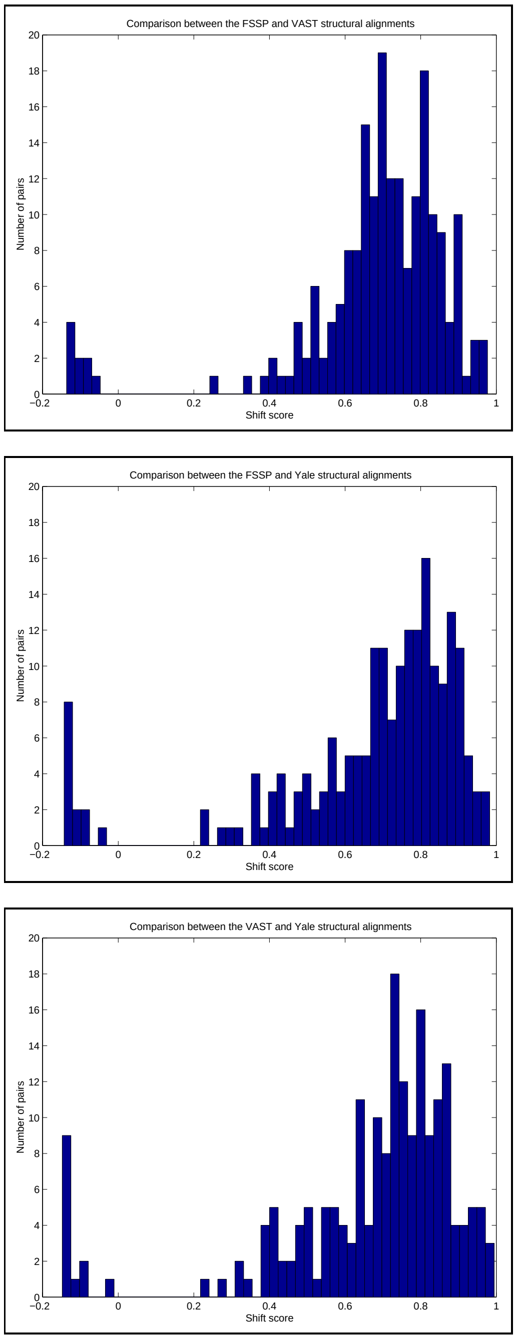


Figure 6.1: To compare the similarity of the alignments produced by the three structural aligners, we used the shift score. For all pairs of structures in the training set, we measured the shift score of the structural alignments from FSSP and VAST, FSSP and Yale, and VAST and Yale respectively. The histograms shown reflect the overall similarity of the FSSP and VAST alignments (top), the FSSP and Yale alignments (middle), and the VAST and Yale alignments (bottom).

homology modeling alignment will tend to score. However, the populations are not without skew. For all pairs of aligners, somewhere between 5 and 10% of the pairs of structures have a shift score of less than zero, indicating that their alignments are non-overlapping. Additionally, approximately 20% of the pairs have shift scores in the range of 0.2 to 0.5, the range of a good to excellent fold recognition alignment. Overall, the mean shift score for each pair of aligners is approximately 0.6, with a median shift score of approximately 0.75. On the whole, the alignments produced by DALI and VAST are the most consistent.

In general, the three alignments are more similar than they are dissimilar, but they do have their differences. The three comparisons are not redundant.

6.2.2 Aligning with structural homologs

First, we studied the accuracy of alignments produced from a profile estimated from FSSP seed alignments. Although FSSP alignments have a lower sequence homology signal than sequence-based alignments, we saw in Chapter 5 that their implied structural information can be very useful for producing profiles. Alignments produced from FSSP seed alignments were consistently better than those produced by sequence-based alignments.

Table 6.2 reports on the accuracy of alignments produced by SAM and CLUSTALW. The most apparent result is that CLUSTALW was not able to derive as much information from the alignment as SAM. This is probably the effect of the regularizers used in both methods. While CLUSTALW uses a weighted set of substitution values, SAM uses Dirichlet regularizers. Systems of substitution matrices, such as the one used in CLUSTALW, work fine for a small number of sequences. However, as the size of the alignment grows, weighted substitution matrices are less effective at ascertaining overall patterns of column conservation [91]. Particularly for alignments with low overall homology, "conserved" columns might show a conserved property more often than a conserved residue. In such cases, Dirichlet mixtures are better able to identify and reflect the conserved property [166].

Shift Score					
	Alignment	Structural Alignment			
Aligner	Options	FSSP	Vast	Yale	Closest
SAM	Viterbi	0.308	0.303	0.276	0.342
SAM	Posterior Decoding	**0.369**	**0.368**	**0.334**	**0.404**
CLUSTALW	None	0.140	0.124	0.120	0.159
CLUSTALW	Secondary Structure	0.143	0.129	0.126	0.163

Alignment Specificity					
	Alignment	Structural Alignment			
Aligner	Options	FSSP	Vast	Yale	Closest
SAM	Viterbi	0.300	0.325	0.289	0.346
SAM	Posterior Decoding	**0.356**	**0.393**	**0.344**	**0.409**
CLUSTALW	None	0.156	0.163	0.156	0.178
CLUSTALW	Secondary Structure	0.161	0.172	0.163	0.184

Alignment Sensitivity					
	Alignment	Structural Alignment			
Aligner	Options	FSSP	Vast	Yale	Closest
SAM	Viterbi	0.350	0.296	0.276	0.350
SAM	Posterior Decoding	**0.384**	**0.330**	**0.305**	**0.383**
CLUSTALW	None	0.210	0.170	0.167	0.214
CLUSTALW	Secondary Structure	0.215	0.179	0.175	0.224

Table 6.2: Performance of SAM and CLUSTALW on profile-based alignments and structure-based profiles. These results were obtained by aligning the target sequence to a profile generated from the FSSP alignment of the template sequence and its structural homologs. In all cases, SAM with posterior decoding (`-adp5`) yielded the best performance, as shown.

Other interesting results in Table 6.2 are that use of posterior decoding leads to a major improvement in SAM alignments, and secondary structure-dependent gap parameters lead to a minor improvement in CLUSTALW alignments.

6.2.3 Aligning with sequence homologs

To analyze the performance of profile methods on sequence-based homologs, we tested two sets of sequence homolog alignments: HSSP and SAM-T99. These alignments differ in the set of sequences aligned and in the method that aligns them. SAM-T99 alignments are far larger than HSSP alignments: for the structures in the test set, the SAM-T99 alignments contained on average 86 sequences, while HSSP alignments contained 15 on average. While the HSSP alignments are built by MaxHom, the SAM-T99 alignments are built by profile alignment with SAM hidden Markov models.

Table 6.3 shows the results of aligning to a profile derived from SAM-T99 and HSSP alignments. In this table, there are a number of points to observe. First, as in Section 6.2.2, SAM seems to derive more information from the alignment. However, here the gap is narrowed. Second, both SAM and CLUSTALW seem to benefit from the additional homologs in the SAM-T99 alignments. One might speculate that some of the difference in performance comes from the method used to estimate the seed alignment. However, when we experimented with allowing both SAM and CLUSTALW to re-estimate the HSSP alignment, results on the re-estimated alignments were consistent with those on the original alignments (data not shown). Third, the SAM alignments continue to benefit from posterior decoding; CLUSTALW alignments benefit from secondary structure-dependent gap parameters with SAM-T99 seed alignments, but not with HSSP alignments. This suggests that when the alignment is small, the secondary structure information is actually misleading as applied.

Shift Score						
	Alignment	Seed	Structural Alignment			
Aligner	Options	Alignment	FSSP	Vast	Yale	Closest
SAM	Viterbi	SAM-T99	0.159	0.170	0.164	0.199
SAM	Posterior-decoding	SAM-T99	0.194	0.212	0.202	0.239
SAM	Viterbi	HSSP	0.098	0.115	0.111	0.135
SAM	Posterior-decoding	HSSP	0.146	0.160	0.158	0.184
CLUSTALW	None	SAM-T99	0.130	0.146	0.146	0.174
CLUSTALW	Secondary Structure	SAM-T99	0.131	0.149	0.150	0.176
CLUSTALW	None	HSSP	0.092	0.094	0.102	0.121
CLUSTALW	Secondary Structure	HSSP	0.086	0.090	0.095	0.111

Alignment Specificity						
	Alignment	Seed	Structural Alignment			
Aligner	Options	Alignment	FSSP	Vast	Yale	Closest
SAM	Viterbi	SAM-T99	0.181	0.211	0.198	0.227
SAM	Posterior-decoding	SAM-T99	0.207	0.245	0.225	0.259
SAM	Viterbi	HSSP	0.127	0.158	0.149	0.164
SAM	Posterior-decoding	HSSP	0.161	0.194	0.184	0.205
CLUSTALW	None	SAM-T99	0.148	0.177	0.171	0.193
CLUSTALW	Secondary Structure	SAM-T99	0.150	0.180	0.173	0.193
CLUSTALW	None	HSSP	0.120	0.139	0.138	0.153
CLUSTALW	Secondary Structure	HSSP	0.114	0.134	0.134	0.143

Alignment Sensitivity						
	Alignment	Seed	Structural Alignment			
Aligner	Options	Alignment	FSSP	Vast	Yale	Closest
SAM	Viterbi	SAM-T99	0.220	0.199	0.195	0.231
SAM	Posterior-decoding	SAM-T99	0.247	0.227	0.217	0.255
SAM	Viterbi	HSSP	0.168	0.160	0.158	0.177
SAM	Posterior-decoding	HSSP	0.206	0.194	0.191	0.215
CLUSTALW	None	SAM-T99	0.203	0.188	0.188	0.216
CLUSTALW	Secondary Structure	SAM-T99	0.206	0.190	0.190	0.218
CLUSTALW	None	HSSP	0.162	0.145	0.150	0.168
CLUSTALW	Secondary Structure	HSSP	0.155	0.139	0.144	0.162

Table 6.3: Performance of SAM and CLUSTALW on profile-based alignments when the profile is derived from sequence homologs of the template sequence. The set of homologs were derived from SAM-T99 and HSSP, respectively.

Shift Score					
	Alignment	Structural Alignment			
Aligner	Options	FSSP	Vast	Yale	Closest
SAM	Viterbi	0.068	0.079	0.073	0.099
SAM	Posterior Decoding	**0.098**	**0.110**	**0.108**	**0.130**
CLUSTALW	None	0.081	0.082	0.085	0.106
CLUSTALW	Secondary Structure	0.064	0.064	0.068	0.088

Alignment Specificity					
	Alignment	Structural Alignment			
Aligner	Options	FSSP	Vast	Yale	Closest
SAM	Viterbi	0.108	0.131	0.122	0.139
SAM	Posterior Decoding	**0.126**	**0.151**	**0.145**	**0.160**
CLUSTALW	None	0.109525	0.126608	0.125168	0.137297
CLUSTALW	Secondary Structure	0.101	0.115	0.113	0.125

Alignment Sensitivity					
	Alignment	Structural Alignment			
Aligner	Options	FSSP	Vast	Yale	Closest
SAM	Viterbi	0.144	0.135	0.130	0.151
SAM	Posterior Decoding	**0.164**	**0.151**	**0.151**	**0.170**
CLUSTALW	None	0.153	0.133	0.136	0.156
CLUSTALW	Secondary Structure	0.134	0.120	0.123	0.141

Table 6.4: Performance of SAM and CLUSTALW on profile-based alignments when no homologs are available. These results were obtained by generating a profile from the template sequence and aligning the target sequence to the profile. In all cases, the SAM-T99 seed alignment worked better than the HSSP seed, and the SAM posterior decoding algorithm yielded the best results.

6.2.4 Aligning with no homologs

As a control, we derived profiles from the template sequence only and aligned the target sequence to the profiles. These results are shown in Table 6.4.

Not surprisingly, both SAM and CLUSTALW suffer when the alignment is made without any homologs, with both methods aligning only about 15% of the residues correctly. As seen with the HSSP seed alignments in Section 6.2.3, posterior decoding seems to improve the SAM alignments, but secondary structure-dependent priors seems to hurt the CLUSTALW alignments. Perhaps this reflects that secondary structure is not entirely conserved between remote homologs. In studies of remote homologs, on average approximately 80% of all structurally-superimposed positions have same secondary structure [156].

6.2.5 Profile alignment compared with progressive alignment

Finally, we compared the accuracy of alignments produced by profile methods to those produced by CLUSTALW's progressive multiple alignment estimation. Here, we provided CLUSTALW with the target sequence, template sequence, and homologs of the template sequence, estimated a multiple alignment of these sequences, and extracted and scored the pairwise alignment of the target and template sequences. We provided CLUSTALW with two different sets of homologs: SAM-T99 homologs and FSSP homologs. Re-estimating a structural alignment might sound strange, but doing so permitted us to contrast the information in the FSSP homolog set with that in the FSSP alignment.

Table 6.5 reports the performance of CLUSTALW on multiple alignment estimation. For comparison, we also report results on profile alignment as estimated previously. Here, we observe that in this case, progressive alignment results were not very sensitive to the choice of seed alignment. However, the choice between profile methods and progressive alignment was more important: profile methods yielded significantly better results than progressive alignment in all cases.

Shift Score						
			Structural Alignment			
Aligner	Method	Homologs	FSSP	Vast	Yale	Closest
SAM	Posterior decoding	SAM-T99	0.194	0.212	0.202	0.239
SAM	Posterior decoding	FSSP	0.369	0.368	0.334	0.404
CLUSTALW	Profile, Secondary Structure	SAM-T99	0.131	0.149	0.150	0.176
CLUSTALW	Profile, Secondary Structure	FSSP	0.143	0.129	0.126	0.163
CLUSTALW	Progressive	SAM-T99	0.091	0.091	0.088	0.112
CLUSTALW	Progressive	FSSP	0.092	0.088	0.096	0.113

Alignment Specificity						
			Structural Alignment			
Aligner	Method	Homologs	FSSP	Vast	Yale	Closest
SAM	Posterior decoding	SAM-T99	0.207	0.245	0.225	0.259
SAM	Posterior decoding	FSSP	0.356	0.393	0.344	0.409
CLUSTALW	Profile, Secondary Structure	SAM-T99	0.150	0.180	0.173	0.193
CLUSTALW	Profile, Secondary Structure	FSSP	0.161	0.172	0.163	0.184
CLUSTALW	Progressive	SAM-T99	0.122	0.139	0.130	0.148
CLUSTALW	Progressive	FSSP	0.122	0.134	0.138	0.145

Alignment Sensitivity						
			Structural Alignment			
Aligner	Method	Homologs	FSSP	Vast	Yale	Closest
SAM	Posterior decoding	SAM-T99	0.247	0.227	0.217	0.255
SAM	Posterior decoding	FSSP	0.384	0.330	0.305	0.383
CLUSTALW	Profile, Secondary Structure	SAM-T99	0.206	0.190	0.190	0.218
CLUSTALW	Profile, Secondary Structure	FSSP	0.215	0.179	0.175	0.224
CLUSTALW	Progressive	SAM-T99	0.157	0.135	0.132	0.163
CLUSTALW	Progressive	FSSP	0.157	0.133	0.143	0.163

Table 6.5: Comparison of the accuracy of alignments generated by profile methods by SAM and CLUSTALW to CLUSTALW progressive multiple alignment estimation.

6.3 Summary

We have focused on the task of aligning some remotely-related template and target sequences once the fold has been predicted. We have investigated two profile alignment methods in comparison to each other and to the popular CLUSTALW progressive alignment method. Both SAM and CLUSTALW's profile method generated significantly more accurate alignments than CLUSTALW's progressive multiple alignment method when provided with the same data. At the risk of generating yet another study in which the author's method is shown as most successful, SAM generated significantly better alignments than CLUSTALW. These results are consistent in comparisons with three different structural aligners, and care was taken to ensure that these three different aligners yielded three different alignments.

On average, the best method was able to correctly align about 40% of the pairs of residues. One might argue that the homologs studied in this investigation are too remote, and that there is little practical value in an alignment that is only 40% correct. To this point, I offer two counter-arguments.

1. As stated by homology modeling expert Roland Dunbrack [159], twilight zone alignments tend to be uneven. Some regions are conserved and have few indels, while others show more variation. Even if indel placement is not precisely correct, the placement is usually close enough to suggest what sections of the proteins are conserved in evolution. The number of indels near the functional residues of the template protein can yield valuable clues as to whether the functionality of the remote homolog is preserved.

2. There is little to gain by working on moderate or easy homologs, as many methods can produce a good alignment. In the twilight zone, where most methods begin to fail, there is much to gain in study of the nature of the failures. This chapter provides the groundwork for our work in predicting reliability of alignment regions, reported in Chapter 9.

Chapter 7

Analysis of pairwise contact potentials

Pairwise contact potentials, the statistical likelihood of pairs of amino acids to interact given their amino acid type, appear everywhere in protein structure prediction. Pairwise contact potentials have figured prominently in validation of partially-determined structures, in homology modeling, threading, and ab initio structure prediction, and in docking prediction.

Since pairwise contact potentials are not used in profile-based fold recognition, one would think that they could be valuable for alignment validation. Predicted contacts could be inferred from the structure of the template sequence and the alignment of the template and target sequences. If an alignment had a large number of unlikely contacts, that alignment might not be trustworthy. However, pairwise contact potentials have been regarded with suspicion, with their detractors claiming that they're little more than an awkward encoding of hydrophobicity. Thus, before one builds a system to harness pairwise contact potentials for alignment validation, one should be sure that they'd be worth the extra complexity.

In this chapter, I report my results assessing the information content of pairwise contact potentials. In Section 7.2, I detail limitations in pairwise information learned through building *contact potential functions*, functions to predict whether a pair of residues would be in contact given their amino acid types and perhaps given information on their physical environment. In Section 7.3, I explore the possibility that pairwise preferences for tertiary contacts might not be statistically significant when conditioned on burial or exposure information. Our conclusions are summarized in Section 7.5.

7.1 Related work

Pairwise contact potentials have a long history in prediction and assessment of protein structure. Pairwise contact potentials have been used to validate partially-determined structures, predict protein docking interfaces, and predict protein structure in ab initio and threading methodologies. They are very well-represented in the literature, with several reviews [163, 186, 98, 184, 85, 126].

Pairwise contact potentials measure the statistical likelihood of each pair of amino acids to interact relative to the frequency of the amino acids. They are typically measured over a database of actual protein structures. Some are based simply on pairwise frequencies [124]. Many involve separate frequency distributions for various ranges of pairwise distances [162, 22]. Others factor in additional information such as solvent accessibility measures [29], torsion angles [191], and various inter-atomic distances [112].

The statistical likelihoods of a predicted contact contributes to an approximation of the stability of the protein in the predicted structure. Individual likelihoods are combined into an overall energy approximation, typically with a formulation involving the exponential of the sum of the individual contact likelihoods. Threading algorithms search for the configuration that minimizes the approximation of overall energy by testing a very large number of configurations and applying some logic to modify portions of the predicted structure.

Encoded within pairwise contact potentials are representations of forces that play a large role in protein structure determination; chief among these is hydrophobicity [85]. Bryant and Lawrence estimated that approximately two thirds of the information in their pairwise contact potential was truly hydrophobicity information [22]. Casari and Sippl measured a simple contact potential based on the frequencies of interaction of all pairs of residues given a distance threshold, and used principal component analysis to derive a hydrophobicity measure for each amino acid from the pairwise potential. This hydrophobicity measure correlated to accepted hydrophobicity measures with correlation coefficients of 0.73 and greater. [25]. There have been successful threading algorithms that predicted protein structure based on simply hydrophobicity [71]. This has led various critics to wonder if they really reflect anything meaningful besides hydrophobicity, and if any remaining information is merely added or unnecessary complexity.

The complexity of certain pairwise contact potentials is also a subject of controversy. The "simpler" pairwise contact potentials involve at least 210 terms, and the more complex potentials involve thousands of terms. While some evidence suggests that the more complex potentials perform better [122, 108], there is other evidence suggesting that simple potentials can perform as well as the complex ones [141]. One compounding factor is that potentials are not trivial to test. A potential can be effective at identifying the native structure from a number of plausible, protein-like decoys, but can perform poorly at identifying a compact structure out of a pool of unfolded decoys. The overall lesson seems to be that the potential should be evaluated with a test appropriate to its application, and extra complexity should be justified or rejected at that time.

However, the harshest criticism of pairwise contact potentials came from Thomas and Dill [179]. For small peptide lengths of eleven and eighteen residues, they generated all possible protein sequences and estimated their structures using exact lattice models. Such peptides are short enough in length that the forces on them and their structures can be computed exactly. After classifying each residue as either hydrophobic or polar, they pointed out their first flaw in the logic underlying contact potentials. In application, pairwise interactions are assumed to be independent of each other. However, they are not independent; the interactions between hydrophobic residues dominate the structure. This dominance, together with restrictions due to chain length, implicitly defines the location of many of the polar residues. Therefore, the location of these polar residues is not determined by its attraction to or repulsion from its neighbors.

Next, Thomas and Dill added their contribution to the argument against complex contact potentials. They compared complex potentials dependent on distance and amino acid type with far simpler potentials dependent on distance and their simple hydrophobic-polar classification. When they graphed interaction energy versus distance, the more complex potential had the same form as the simpler potential for the three amino acids shown. Further, the simpler potential had the same form as another simple, distance-dependent potential that classified residues as interior or exterior. Thus, they surmised that the more complex potentials mostly reflect such facts as that the hydrophobics Isoleucine and Valine tend to be together in the protein interior.

Finally, Thomas and Dill tested a simple hydrophobic-polar potential against the far more complex Hendlich potential [60] at identifying the native fold from a set of decoys. The simple potential performed nearly as well as the more complex potential; out of 65 cases, the simple potential identified the correct structure 31 times, versus 37 times for the more complex potential. Moreover, their mistakes were consistent; when the complex potential failed to identify the native fold, the simple potential failed also.

More recent analyses of pairwise contact potentials have stated that the observed contact potential is influenced by the protein structure database in subtle ways. Zhang and Skolnick showed that the accuracy of the potentials depends on the stability of the structures in the database [195]. Furuichi and Koehl showed that if a contact cutoff of greater than eight angstroms is used, the size of the proteins in the database factors into the contact potential [49]. Also, and not surprisingly, Furuichi and Koehl showed that the topologies in the database influence the potential; if the structure database is dominated by helical proteins, the resulting potentials will perform well on helical proteins only.

Vendruscolo and Domany argued that pairwise contact potentials alone are insufficient to predict or verify protein structure [187]. Given a contact map of the crambin protein and maps of plausible decoys, they tried to find a contact potential function that would assign a lower energy to the native contact map than to the decoys. They trained a perceptron to estimate the contact potentials. Perceptron training did not converge, and the partially-trained perceptron could only recover 40% of the actual contacts.

Sunyaev et al [175] also presented a statistical argument that did not reflect well on contact potentials. They stated that if a potential describes the fit between the amino acids and environment, then the relationship between the amino acids, and between the amino acids and their environment, must be statistically significant. Using chi-squared tests and Bahahur theory, they tested if amino acid pairwise preferences were significant given backbone conformation, accessibility, and pairwise distances. They observed that certain amino acids are "average", and have little or no statistical preference for any combination of interaction partner and environment tested. These average amino acids included Alanine, Asparagine, Aspartic Acid, Glutamine, Histidine, Serine, Threonine, and Tyrosine.

With so many criticisms, how does one explain the historical success of contact potentials? The first argument is that the limitations of contact potentials do not necessarily render them ineffective in practice. While Sunyaev et. al. found no distinct association between eight amino acids and their environments or pairwise interaction partners, they found such a signal for the other twelve amino acids. Some threading methods involve pairwise contact potentials have been very successful in the CASP contests [128, 109]. These methods are getting information from somewhere.

Mirney and Shakhnovich [122] suggested that part of the success of threading methods in CASP2 was that the target proteins lent themselves well to threading. They observed that threading accuracy depends on two factors: the accuracy of the potentials and the similarity of the structures, and surmised that the success of the threading groups in the CASP2 contest was due in part to the targets, and their similarity to existing template structures. Further, when the template and target structure are very close (within 2 angstroms RMS deviation), threading methods will withstand a small amount of noise in the contact potentials. However, this does not mean that the potentials are unimportant; when they compared simple frequency-based potentials with more complex distance-based potentials, the frequency-based potentials performed significantly worse even when the structures are close. Further, one should note that in CASP3, the Bryant and Sippl threading teams made strong predictions on fold recognition targets with only fold-level structural similarity [128]. In summary, the selection of target proteins in CASP2 might have contributed to the success of the threading teams, but there was more to their success than a lucky selection of sequences.

There seems to be little argument at this point that contact potentials convey information. Cootes et. al. [30] analyzed the statistical relation between an amino acid, its neighbor, and its structural environment. Using log-linear analysis, they found strong, statistical relations between an amino acid and its neighbor, especially for long-range, tertiary interactions. Further, they showed that pairwise preferences were distinct by structural type, and were non-symmetric: if an IJ pair was observed, with I appearing in the sequence before J, a JI pair was not equally likely to be observed. Every one of the arguments saying that hydrophobicity is most of the information in pairwise contact potentials can be turned around to say that they encode information in addition to hydrophobicity. For example, when Thomas and Dill [179] tested a hydrophobicity-based potential against a more complex potential, their simple potential identified the native fold in 37 out of 65 cases while the more complex potential identified it in 41 out of 65 cases. While one can say that the simple potential performed nearly as well as the complex one, one can also say that the simple potential failed in four cases where the complex one succeeded.

However, whether the likelihood of predicted contacts would be a useful measure of alignment validation is a different question. An incorrect alignment can yield a prediction of a compact, protein-like structure or an implausible structure with large holes in the interior, and there is usually no prior way to guess which type of error is more likely. There is no general-purpose contact potential; some are better at detecting unfolded structures, and some are better at picking the native structure over protein-like decoys [184, 141]. Thus, at best, any given contact potential could detect certain classes of alignment errors, but only certain classes.

Similarly, potentials optimized for recognizing the native structure are not necessarily viable for recognizing similar structures. Kocher et. al. [96] evaluated various distance-based potentials on their ability to identify the native structure. They found that the effectiveness of the potential was related to the choice of amino acid reference point: potentials that measured distances from the centroid of the side chain were more effective than those that used C_β distance; those that used the distance between C_β atoms were more effective than those that measured the distance between C_α atoms. However, the more effective potentials rely on information that is not always conserved between remote homologs.

Flores et. al. [47] observed that as sequence similarity decreases, accessibility, side chain angles, and secondary structure all become less conserved. Chung and Subbiah [27] observed substantial differences in side chain torsion angles of remote homologs, suggesting that remote homologs make different side chain contacts. Russell and Barton [155] examined structurally-superimposed positions in remote homologs and *analogs*: proteins with similar structure but different function. They found no more than chance relation in accessibility, side-chain contacts, and similarity of secondary structure; in these three quantities, remote homologs and analogs showed no greater similarity than unrelated proteins show by chance. Finally, Russell et. al [156] studied pairs of structurally-aligned proteins and divided them according to their relation in the SCOP database [73] as homologs (same superfamily) or analogs (same fold, different superfamily). In structurally-superimposed positions, the SCOP homologs and analogs showed some similarity in secondary structure (81 - 85% conserved) but weaker similarity in accessibility (approximately 60% conserved). Thus, information that has been used to help contact potentials identify native structures is not conserved between remote homologs.

In summary, the lessons from the literature are that the information in pairwise contact potentials is probably significant. However, the potential should be chosen according to the application, and the choice of potential should reflect assumptions to be made within the application. For alignment validation, one should not rely on structural features that are not conserved in remote homologs: location of the side chains and accessibility. If one chooses a distance-specific potential, one should select a definition of pairwise distance that is reasonably conserved between remote homologs; side chain centroids are a poor choice, but C_α or C_β distances are better choices. There is active debate on whether complex or simple potential functions work better. Conditioning potentials on secondary structure might be of benefit, and tertiary interactions appear to be more interesting than local interactions.

7.2 Knowing one amino acid tells little about its neighbors

The first step in our investigation of pairwise contact potentials was to estimate a set of potential functions. Access to some actual potential functions would enable us to experiment with them, and observe what sort of information they seemed to convey by what situations in which they seemed to perform well.

In collaboration with Temple Smith's group and with Rick Lathrop, we experimented with a number of the major feature sets on pairwise contact prediction. In addition to the identity of the contact pair, these feature sets contain environmental inputs such as pairwise distance and accessibility, as described in Table 7.1. For each feature set, our goal was to predict the likelihood of contact given the interacting amino acids, plus any environmental attributes. Our approach was to train a system of neural networks to estimate the posterior probability of contact given the amino acid types and environmental features: $\hat{P}(a_1, a_2|\vec{e})$, where a_1 and a_2 are the types of amino acids involved in the interaction, and $\vec{e}$ is a set of inputs describing the environment of the interacting pair.

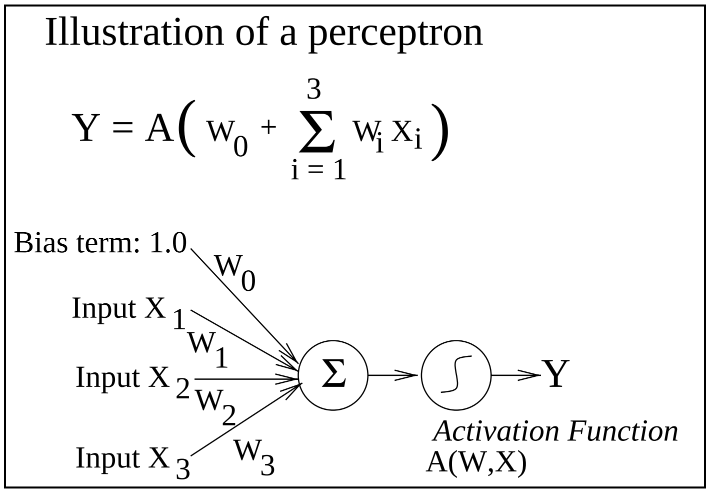


Illustration of a perceptron
$Y = A\left(w_0 + \sum_{i=1}^{3} w_i x_i \right)$
Bias term: 1.0
Input X 1
Input X 2
Input X 3
W0
W1
W2
W3
Σ
Y
Activation Function
A(W,X)

Figure 7.1: Illustration of a perceptron

7.2.1 Introduction to neural networks

Neural networks, or more pedantically *artificial neural networks*, are mathematical modeling systems inspired by models of learning in the brain [63, 14]. Learning in biological systems involves repeated adjustment to the strength of various synaptic connections between neurons. An artificial neural network involves inputs connected to one or more outputs with each connection represented by a *weight*, a real-valued number representing the connection strength. Neural network learning, or *training*, involves repeatedly presenting the network with a number of *training examples* consisting of inputs and target outputs, and adjusting the weights in order to better approximate the target outputs given the input.

The simplest flavor of a neural network, called a *perceptron*, is illustrated in Figure 7.1. This perceptron features one *output unit* $\vec{Y}$ connected to a layer of *input units* $\vec{X}$ by an *outer layer weight vector* $\vec{W}$. The input to the output unit Y is $W_0 + \sum_{i=1}^{N} W_i X_i$, the sum of the products of the inputs and their weights, plus the weight on the bias term. At the output unit, an *activation function* is applied to this input. The purpose of the activation function is to map this input to the expected range of the output. For example, if the perceptron models a probability function, the expected range of the output is from 0.0 to 1.0. An activation function that would map the range of real numbers into this output range is the *log-sigmoid activation function* $\frac{1}{1+e^{-X}}$. Also defined at the output unit Y is some *error function*, a function to quantify the error of the neural network by comparing the network output Y with the target output O. A common error function is squared error: $E = (Y - O)^2$. Alternatively, if the network estimates a probability function and the estimated value Y is an estimated probability with O being the actual probability, one might use an *entropic error* function $E = -O\log(Y) - (1-O)\log(1-Y)$. Thus, there is a set of equations relating the network error to the inputs and the weights.

Upon initialization, the weights of the neural network are assigned random values. During training, the network is presented with a number of training examples $(\vec{X}, O)$ consisting of a set of inputs $\vec{X}$ and a target output O. The learning algorithm calculates the network output and network error $E(Y, O)$, then changes the weights slightly according to a *learning rule*, such as $\vec{W}_{new} = \vec{W}_{old} - \nu\Delta W$. Here, ΔW is a *weight change vector*, often calculated from the neural network equations and ν is an algorithmic parameter called the *learning rate*, which serves to regulate the amount of change to the network to prevent fluctuation. In *back-propagation networks*, the weight change vector ΔW is computed by calculating the partial derivative $\frac{\partial Y}{\partial \vec{W}}$ to propagate the error backwards through the weights of the network. This is known as *gradient descent learning*.

For certain problems, the perceptron architecture is not sufficient to model the function desired. In such cases, one can use a *two-layered network*, in which there is one set of weights connecting the inputs to a *hidden layer* of perceptrons, and another set of weights connecting the hidden layer to the outputs. As previously, each perceptron unit has an associated activation function, whether the perceptron is part of the hidden layer or the output layer. Again, there is an error function defined for the outputs of the network. Thus, there is still a set of equations relating the error function to the inputs and weights of the neural network.

Another type of neural network layer is a *softmax layer* [18]. Softmax layers are typically used at the outermost layer of the network. Their effect is similar to an activation function: to derive a probability distribution from a vector of real-valued quantities. Each softmax output Y_i is associated with a softmax input S_i, and is related to the softmax input vector $\vec{S}$ by the equation: $Y_i = \frac{e^{S_i}}{\sum_j e^{S_j}}$

While neural networks might seem mysterious, they prove to have interpretations in Bayesian statistics. Consider a neural network trained for classification: when the neural network is given an example represented by the inputs $\vec{X}$, it estimates the probability with which the example belongs to each of a number of classes $\vec{Y}$. Such neural networks have been shown to estimate the Bayesian posterior probability of each output class Y_i [150].

Applications of neural networks include statistical systems with a very large number of inputs, and a basis for exploratory analysis when one believes some inputs and an output are associated but is unclear on the nature of the association. Critics point out that neural networks are difficult to decipher, and are too much of a black box: a person can throw data at a neural network package and hope for the best, but that's not good science. Certainly, when one works with neural networks, one should follow a proper experimental protocol. Part of this protocol entails ensuring that complexity is used only where justified: for example, extraneous inputs should be removed, and a multi-layer network should not be used where a perceptron is sufficient. Yet critics aside, neural networks have been applied successfully to a number of areas in computational biology including secondary structure prediction [152], estimating potential functions for threading [58] and gene prediction [185].

7.2.2 Description of the feature sets

To experiment with contact potential functions, we built neural-network-based contact potential functions for several of the major feature sets in the literature. This section describes those feature sets. Each feature set is detailed below, and the contents of the feature set are summarized in Table 7.1. All feature sets described here were built in the lab of Temple Smith [105] on the fifty-nine proteins listed in Table 7.2, a set representing most of the distinct protein folds in the PDB at that time.

Pairwise feature sets consist of tuples $(a_1, a_2, \vec{e})$, where $a1$ and $a2$ are amino acids in a contact pair and $\vec{e}$ describes some aspect of the environment surrounding a_1 and a_2. The environmental features differ slightly for each feature set. All feature sets contain some features describing the geometry of the pair, such as pairwise distance. Some contain additional fields describing environment around a_1 and a_2, such as accessibility measures.

Distance-based feature sets: Bryant-Lawrence, Jernigan, and Sippl

Many pairwise feature sets are based on the idea that amino acid pairs exhibit a regular pattern of pairwise interaction given distance. For example, if an isoleucine is interacting with another residue seven angstroms away, that other residue might be a phenylalanine. Pairwise feature sets that predict contact based on amino acid type and distance include *Bryant-Lawrence* [22], *Jernigan* [83], and *Sippl* [162]. The only environmental feature for these feature sets is pairwise distance. The differences between the feature sets concern the precise manner of measuring pairwise distance, and the resolution to which it is measured.

Sippl uses the distance between the C_α atoms, measured to a resolution of 0.1 Å. Jernigan defines the location of each residue as a point at the center of the side chain, and uses the distance between these points as the pairwise distance, and measures it to a resolution of 0.001 Å. Bryant-Lawrence defines the distance between two residues using a point 2.4 Å from C_α in the C_β direction, measured to a resolution of 0.1Å. This distance is referred to as the *peptide midpoint distance.*

All three feature sets define contact according to pairwise distance: two residues are considered in contact if they are not adjacent in the protein chain and if their pairwise distance is less than some threshold. Jernigan uses a pairwise threshold of 6.5 Å, Bryant-Lawrence uses 10 Å, and Sippl uses 15 Å. Further, each of these feature sets divide contact into various classes, according to the pairwise distance.

Feature set	Features
Bryant-Lawrence [22]	Peptide Midpoint Distance
Crippen [112]	O - N distance, N - O distance, C - N distance, N - C distance, O - Cbeta distance, Cbeta - O distance, N - Cbeta distance, Cbeta - N distance, Cbeta - Cbeta distance
Jernigan [83]	Side-chain Midpoint Distance
Sippl [162]	Alpha Carbon Distance
True-mrf [191]	Beta Carbon Distance, Omega angle Amino Acid 1 Phi Angle, Amino Acid 2 Phi Angle, Solvent Exposure at Amino Acid 1, Solvent Exposure at Amino Acid 2, Secondary Structure at Amino Acid 1, Secondary Structure at Amino Acid 2
Tetrahedron [29]	Window through which Amino Acid 1 looks at Amino Acid 2, Window through which Amino Acid 2 looks at Amino Acid 1, Secondary structure at Amino Acid 1, Secondary structure at Amino Acid 2, Fraction of Volume Accessible via the first window specified, Fraction of Volume Accessible via the second window specified, Solvent Exposure at Amino Acid 1, Solvent Exposure at Amino Acid 2, Beta Carbon Distance

Table 7.1: Contents of the feature sets used for estimating pairwise contact potential functions

1aak	1aba	1aep	1alc	1apa	1baa	1bgc	1byh
1cde	1cew	1dhr	1f3g	1hoe	1ifc	1lec	1lis
1mat	1mbd	1nar	1pkp	1plc	1rcb	1rec	1s01
1tie	1ubq	1yat	256b	2act	2ca2	2cpl	2cpp
2cyp	2end	2had	2hpr	2lzm	2mcm	2mhr	2sns
351c	3adk	3chy	3est	3tgl	4bp2	4cpv	4fgf
4fxn	5cpa	5cyt	5fd1	5tmn	7rsa	8dfr	9api
9rnt							

Table 7.2: List of the PDB structures contained in the Lathrop-Smith data set

Extended feature sets: Crippen, Tetrahedron, and True-mrf

A second class of feature sets contain pairwise distance plus additional features to describe the environment around the pair or the orientation of the two residues with respect to each other. These feature sets are *Tetrahedron* [29], *True-mrf* [191], and *Crippen* [112].

Crippen captures pairwise orientation through the distances between various parts of the two residues: $O-N$, $N-O$, $C-N$, $N-C$, $O-C_\beta$, $C_\beta-O$, $N-C_\beta$, $C_\beta-N$, and $C_\beta-C_\beta$. Altogether, these distances describe the relative orientation of the two residues with respect to each other. The feature set includes three different types of contacts with separate contact criteria. A backbone-backbone contact must have an $O-N$ distance of less than 3.2 Å and a $C-N$ distance of more than 3.9 Å. A backbone–side-chain contact must have a distance between O and C_β or N and C_β of less than 5.0 Å. These same distances are measured for side-chain–side-chain contacts; the cut-off is 9.0 Å, and there must be no atom within 1.4 Å of the pairwise axis.

True-mrf describes pairwise orientation through polar geometry rather than through distances. It orients the residue pair according to their C_β atoms: pairwise distance is defined as the distance between the C_β atoms, and the polar geometry is defined with respect to the $C_\beta-C_\beta$ axis. The *phi angle* of each residue is defined by drawing an axis through C_β and the center of the side chain. Phi is the angle between this side chain axis and the $C_\beta-C_\beta$ axis. The *Omega* angle is defined by looking down the $C_\beta-C_\beta$ axis, and measuring the angle between the two side-chain axes. In addition, True-mrf includes the secondary structure and *solvent exposure* at both residues. Solvent exposure is an accessibility measure designed for threading algorithms, where the goal is to find structural similarity between sequences that are not necessarily similar. The traditional solvent accessibility measure used in the DSSP program [88] is inappropriate for threading algorithms because it retains too much indirect information on the protein sequence. Solvent exposure removes this sequence information as follows. First, the side chains of all amino acids are replaced with the side chain of Alanine. Then, to avoid spurious holes in the interior of the protein, the

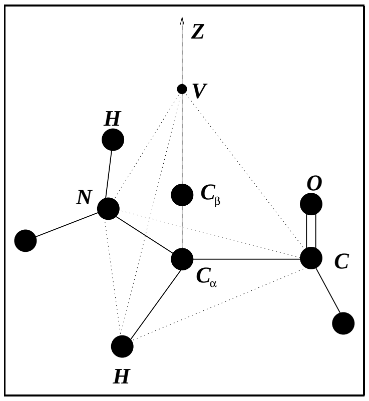


Figure 7.2: Idealized illustration of the geometry of the tetrahedron feature set [29]

radii of C_β and the water sphere are increased. Finally, Eisenberg's algorithm is used to calculate the accessible area [15].

Finally, the Tetrahedron feature set models the environment of each amino acid by means of abstract geometry, as described roughly in Figure 7.2. First, a coordinate system is established with the C_α - C_β direction forming the Z axis, the X axis bisecting the $NH - C_\alpha - CO$ angle, and the Y axis orthogonal to Z and X axes. A triangle is formed by placing one vertex at N, one at H, and one at C. A fourth vertex, V, is placed on the Z axis above C_β to form three more triangles equal in size to the first. This defines a tetrahedron centered at C_β. For each contact pair (a_1, a_2), the Tetrahedron feature set describes the window through which a_1 looks at a_2, the window through which a_2 looks at a_1, and the percentage of the window surface that is accessible to solvent. Also included in this feature set are the secondary structure and the solvent exposure for both a_1 and a_2. The solvent exposure is computed in the same way as in True-mrf.

7.2.3 Results and analysis

To experiment with contact potential functions, we trained neural networks to estimate a number of potential functions. Given a pair of interacting amino acids a_1 and a_2 observed in an environment described by $\vec{e}$, these neural networks estimated the probability that a_1 and a_2 would be in contact given environment $\vec{e}$: $\hat{P}(a_1, a_2|\vec{e})$.

Not all amino acids are equally likely, and this fact must be taken into account when judging neural net performance. Suppose a neural net was trained to predict a_1, the first amino acid in some pair, and was trained to predict it with no inputs. The net would minimize its losses by "playing the odds", predicting each amino acid according to its background frequency.

When a neural net is given a set of input features $\vec{x}$ and is asked to predict a_1 given $\vec{x}$, the value of feature set $\vec{x}$ is reflected by how strongly the net predicts a_1 given $\vec{x}$ relative to the background probability of a_1. For example, Alanine is a common amino acid; in general, about one out of every ten randomly-chosen amino acids will be Alanine. Thus, a neural net with no features would always predict Alanine with a probability of 0.1. Suppose a different neural net had a better feature vector $\vec{x}$, and when Alanine was the correct answer, this net predicted it with an average probability of 0.4. The ratio between 0.4 and 0.1 represents how much information the neural net is getting out of feature vector $\vec{x}$. We measure this information gain with a *log odds ratio*: $log(\frac{P(a_1|\vec{x})}{P(a_1)})$. A high log odds ratio reflects an informative feature vector $\vec{x}$.

For each feature set, we trained a neural net to estimate the probability of the amino acid pair given the environment, $\hat{P}(a_1, a_2|\vec{e})$. Early results showed that our performance improved if we divided this problem into two parts: first estimate the probability of the first amino acid given the environment $\hat{P}(a_1|\vec{e})$, then estimate the probability of the second amino acid given the first and the environment $\hat{P}(a_2|a_1, \vec{e})$. The pairwise likelihood $\hat{P}(a_1, a_2|\vec{e})$ is the product of these two terms. We estimated the probability of the pair by predicting the first residue given the environmental inputs and then predicting the second residue given the first plus the environmental inputs. Table 7.3 summarizes these results using the log odds ratio over the background frequencies.

Something interesting can be seen in Table 7.3: predicting the second amino acid given the first and other inputs is almost as hard as predicting the first given those inputs. In other words, *knowing the first amino acid is of little value for predicting the second.* This contradicts the popular hypothesis that pairwise contacts exhibit a distinct pattern. Clearly, further investigation was called for.

Continuing from this observation, we estimated the mutual information between the two amino acids in contact for each feature set. Mutual information measures the amount of information shared by two quantities; it is similar to correlation, but more general [31]. Mutual information is calculated according to the following formulae:

$$\begin{aligned}
N_A, N_B &= \text{Number of classes of quantities A and B respectively} \\
P(A_i) &= \text{Probability of class i of quantity A} \\
H(A) &= \text{Entropy of quantity A} \\
&= -\sum_{i=1}^{N_A} P(A_i)log_2(P(A_i)) \\
H(A,B) &= \text{Joint entropy of quantities A and B} \\
&= = \sum_{i=1}^{N_A}\sum_{j=1}^{N_B} P(A_i,B_j)log_2(P(A_i,B_j)) \\
I(A,B) &= H(A) + H(B) - H(A,B) \\
&= \sum_{i=1}^{N_A}\sum_{j=1}^{N_B} P(A_i,B_j)\log_2\left(\frac{P(A_i,B_j)}{P(A_i)P(B_j)}\right) \\
&= \sum_{i=1}^{N_A}\sum_{j=1}^{N_B} P(A_i,B_j)\log_2\left(\frac{P(B_j|A_i)}{P(B_j)}\right) \\
&= \sum_{i=1}^{N_A}\sum_{j=1}^{N_B} P(A_i,B_j)\log_2\left(\frac{P(A_i|B_j)}{P(A_i)}\right)
\end{aligned}$$

For each feature set, Table 7.4 lists $I(a_1, a_2)$, the mutual information between the contacting residues. By the general formula mutual information $I(a_1, a_2)$ is computed as follows: $I(a_1, a_2) = \sum_{a_1}\sum_{a_2} P(a_1, a_2)\log_2\left(\frac{P(a_2|a_1)}{P(a_2)}\right)$ where here the sums are over the 20 types of amino acids. When these probabilities are estimated from the data, we use the estimated probability $\hat{P}$ in place of the actual probability P. This is how the numbers in the column labeled $I(a_1, a_2)$ in Table 7.4 were calculated.

Feature Set	$log_2(\frac{\hat{P}(a_1\|\vec{e})}{\hat{P}(a_1)})$			$log_2(\frac{\hat{P}(a_2\|a_1,\vec{e})}{\hat{P}(a_2)})$		
	Training Set	Test Set	Verify Set	Training Set	Test Set	Verify Set
Bryant-Lawrence [22]	0.0110	0.0062	0.0049	0.0303	0.0071	0.0004
Crippen [112]	0.1347	0.1323	0.1212	0.1738	0.1560	0.1526
Jernigan [83]	0.0205	0.0186	0.0052	0.1055	0.0483	0.0488
Sippl [162]	0.0003	-0.0014	-0.0010	0.0059	0.0013	0.0017
Tetrahedron [29]	0.3227	0.2464	0.2190	0.3565	0.2483	0.2417
True-mrf [191]	0.2647	0.2241	0.2127	0.2917	0.2408	0.2264

Table 7.3: Results for estimating contact potential functions for each feature set. Given a pair of interacting amino acids a_1 and a_2 and environmental features $\vec{e}$, the potential functions were trained to estimate the probability that a_1 and a_2 are in contact given $\vec{e}$, $\hat{P}(a_1, a_2|\vec{e})$, or alternatively $\hat{P}(a_1|\vec{e})\hat{P}(a_2|a_1, \vec{e})$. The log likelihood ratios shown represent how much information the neural networks learned relative to the amino acid background distributions. The training set was used to adjust network weights. The examples in the test set (alternatively called *cross-training set*) were not seen during training, but overall test set performance was used to adjust training parameters. The verify set was an independent set of examples not seen until training was completed. The data was partitioned with at random with 60% of the examples in the training set, 20% in the test set, and 20% in the verify set.

The column labeled $log_2(\frac{\hat{P}(a_2|a_1,\vec{e})}{\hat{P}(a_2|\vec{e})})$ shows the conditional mutual information

$$\sum_{a_1}\sum_{a_2} P(a_1, a_2|\vec{e}) \log_2 \left(\frac{\hat{P}(a_2|a_1, \vec{e})}{\hat{P}(a_2|\vec{e})} \right)$$

where every probability is conditioned on knowledge of a particular amino acid environment. This latter quantity approximates how much information the first amino acid provides on the second, as estimated by the neural networks. This quantity is expected to be less than $I(a_1, a_2)$ when knowledge of the environment, such as whether the amino acids are buried or exposed, "explains away" the apparent dependency between them. This is what we observe. In short, our results showed that knowing the identity of either residue yields less than one tenth of one bit of information on the other residue. For contrast, maximum information is approximately four bits per residue and one single homolog yields approximately three bits per residue [91], if the homology is not too distant. Thus, the information available from amino acid potentials seems to be limited.

Feature Set	$I(a_1, a_2)$	$log_2(\frac{\hat{P}(a_2\|a_1,\vec{e})}{\hat{P}(a_2\|\vec{e})})$
Bryant-Lawrence [22]	0.0239	0.0130
Crippen [112]	0.0622	0.0193
Jernigan [83]	0.0876	0.0277
Sippl [162]	0.0131	0.0013
True-mrf [191]	0.0394	0.0250
Tetrahedron [29]	0.0805	0.0010

Table 7.4: Mutual Information and Log Likelihood Ratios on the Contact Pairs. Mutual information represents the strength of the pairwise signal between amino acids a_1 and a_2, and the log likelihood ratio approximates how much information the first amino acid provides on the second, as estimated by the neural networks.

7.3 Statistical analysis of pairwise contacts

In Section 7.2, we saw that pairwise contact patterns are weak. All the same, are they significant? This section describes our work to address that question.

Most people do not believe that protein structure is governed directly by specific pairwise preferences, such as isoleucines seeking out valines before threonines. Rather, most believe that observed pairwise preferences reflect that hydrophobic residues tend to be buried within the protein interior, and thus tend to be near other hydrophobic residues. This leads to the question of how much information is left after hydrophobic effects are factored out, and we have addressed this question by studying patterns of pairwise mutual information within buried regions and within exposed regions, and by comparing these patterns to what one would expect if contact pairs were statistically independent.

In addition, most believe that there are certain pairs of amino acids that exhibit distinct attractive or repulsive forces for clear biochemical reasons. We have examined how much pairwise signal can be attributed to two sets of forces:

- The preference for cysteines to interact with other cysteines, and
- Among charged residues, the attraction between residues of opposite charge and repulsion between residues of the same charge.

7.3.1 Description of the dataset

The dataset used for these experiments was assembled by the Temple Smith lab under the directions of Rick Lathrop. It is an enlarged version of the dataset described in 7.2.2 containing lists of pairwise contacts for 208 structures for a total of 45,505 contact pairs. All contacts are between residues in secondary structure elements: helices or beta strands. All had a C_β distance of eight angstroms or less, and were three or more positions apart in the protein sequence.

Of these 45,505 contact pairs, we focused on the 22,707 pairs involved in *tertiary* contacts: long-range interactions between proteins at least five residues apart in the protein sequence, a distance just long enough to omit pairs of residues in adjacent turns of α-helices. There are two good reasons to focus on tertiary interactions. First, earlier analysis found tertiary contacts to be the most significant [30]. Second, the very successful threading team led by Steve Bryant focuses in contacts separated by at least one turn of an α-helix [140].

Along with the interacting amino acids, the dataset contains various data describing their environment, including secondary structure and solvent exposure. *Solvent exposure*, described previously in Section 7.2.2, is an accessibility measure designed for threading algorithms. Figure 7.3 shows a histogram of the solvent exposures of the 91,010 residues in the 45,505 contact pairs in the dataset.

7.3.2 Pairwise information as influenced by exposure

First, we studied how the mutual information between interacting residues is affected by exposure. Specifically, we studied interacting pairs for which the solvent exposure of at least one residue was no more than some threshold value, and contrasted that with pairs for which both residues had a solvent exposure of greater than the threshold value. In preliminary work, we studied pairs for which both residues were buried, but found that when at least one residue was buried, the same patterns were evident and the pool of data was larger.

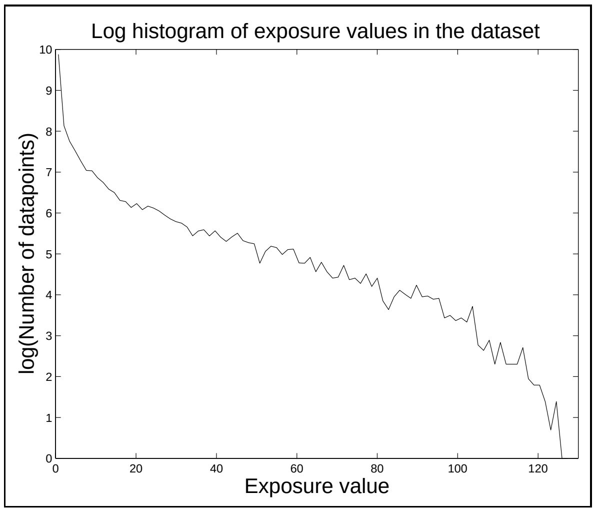


Figure 7.3: Histogram detailing the exposure values of the interacting residues

Observed mutual information depends on sample size

When comparing the mutual information of a large pool of data to that of a small pool, one must be aware that observed mutual information is heavily affected by sample size. When mutual information is measured with too small a sample size, it will tend to increase artificially [192]. Given the probability distributions for the interacting amino acid, we estimated their expected mutual information according to the equations shown below.

$$\begin{aligned}
(a_1, a_2) &= \text{Two interacting amino acids} \\
\hat{H}(a_1) &= \text{Estimate of the entropy of } a_1 \\
\hat{H}(a_2) &= \text{Estimate of the entropy of } a_2 \\
\hat{H}(a_1, a_2) &= \text{Estimate of the entropy of the amino acid pair } (a_1, a_2) \\
\hat{I}(a_1, a_2) &= \text{Estimate of the mutual information between } a_1 \text{ and } a_2 \\
\vec{\hat{P}}(a_1) &= \text{Estimated probability distribution of } a_1 \\
\vec{\hat{P}}(a_2) &= \text{Estimated probability distribution of } a_2 \\
\vec{\hat{P}}(a_1, a_2) &= \text{Estimated joint probability distribution } (a_1, a_2)
\end{aligned}$$

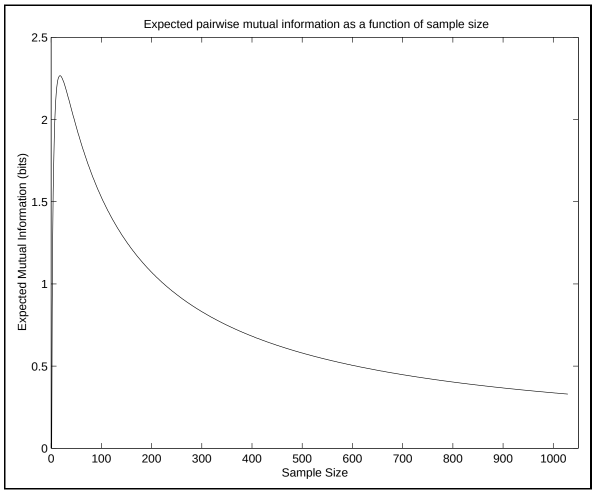


Figure 7.4: For the tertiary contacts, this graph shows the expected mutual information as a function of sample size. The expected mutual information was computed using the formulas shown in Section 7.3.2. If we accept that $0 \log(0) == 0$, this expected mutual information starts at zero when the sample size is zero, and reaches a maximum value of 2.26 with a sample size of 17.

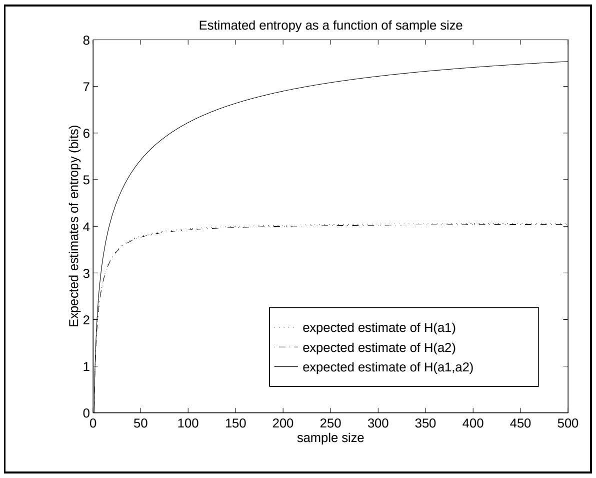


Figure 7.5: Expected entropy as a function of sample size

$$
\begin{aligned}
E[X] &= \text{Expected value of some quantity X} \\
E[\hat{I}(a_1,a_2)] &= E[\hat{H}(a_1)+\hat{H}(a_2)-\hat{H}(a_1,a_2)] \\
&= E\Bigg[\sum_{a_1}\hat{P}(a_1)\log_2(\hat{P}(a_1))+\sum_{a_2}\hat{P}(a_j)\log_2(\hat{P}(a_2)) \\
&\quad -\sum_{a_1}\sum_{a_2}\hat{P}(a_1,a_2)\log_2(\hat{P}_{i,j}(a_1,a_2))\Bigg] \\
&= \sum_{a_1}E\left[\hat{P}(a_1)\log_2(\hat{P}(a_1))\right]+\sum_{a_2}E\left[\hat{P}(a_2)\log_2(\hat{P}(a_2))\right] \\
&\quad +\sum_{a_1}\sum_{a_2}E\left[\hat{P}(a_1,a_2)\log_2(\hat{P}(a_1,a_2))\right]
\end{aligned}
$$

Here, when we estimate a probability distribution such as $\hat{P}(a_1)$, if we have N total observations and n_i observations that amino acid a_i is a_1, then our estimate of the probability that a_i is a_1, $\hat{P}_i$, is computed as $\frac{n_i}{N}$. Thus, we can write the estimated mutual information as follows below.

$$
\begin{aligned}
N &= \text{Total number of observations} \\
n_i &= \text{Number of times that amino acid } a_i \text{ is observed as } a_1 \\
n_j &= \text{Number of times that amino acid } a_j \text{ is observed as } a_2 \\
n_{ij} &= \text{Number of times that } a_i \text{ is observed as } a_1 \text{ and } a_j \text{ is observed as } a_2 \\
E[\hat{I}(a_1,a_2)] &= \sum_{i=1}^{20}\frac{n_i}{N}\log_2\left(\frac{n_i}{N}\right)+\sum_{j=1}^{20}\frac{n_j}{N}\log_2\left(\frac{n_j}{N}\right)-\sum_{i=1}^{20}\sum_{j=1}^{20}\frac{n_{ij}}{N}\log_2\left(\frac{n_{ij}}{N}\right)
\end{aligned}
$$

Note that the counts of observations n_i, n_j, and n_{ij} are derived from the data. Consider n_i, the number of times that some amino acid a_i is observed as the first amino acid, a_1. A priori, we do not know the value of n_i, except that it lies somewhere between 0 and N inclusive. However, we can estimate it as follows. Let x represent some value in the range from 0 to N. $P(n_i = x)$, the probability that we see x observations of amino acid a_i as a_1, can be estimated as shown below.

$$
P_i = \text{The true probability that amino acid } a_i \text{ is } a_1
$$

$$\begin{aligned} x &= \text{Some potential value for } n_i \\ P(n_i = x) &= \begin{pmatrix} N \\ x \end{pmatrix} (P_i)^x (1 - P_i)^{(N-x)} \end{aligned}$$

If we let y and z describe potential values for n_j and n_{ij} respectively, we can estimate the probability that y is n_j and z is n_{ij} in the same way.

$$\begin{aligned} Q_j &= \text{The true probability that amino acid } a_j \text{ is } a_2 \\ P_{ij} &= \text{The true probability that } a_i \text{ is } a_1 \text{ and } a_j \text{ is } a_2 \\ y &= \text{Some potential value for } n_j \\ z &= \text{Some potential value for } n_{ij} \\ P(n_j = y) &= \begin{pmatrix} N \\ y \end{pmatrix} (Q_i)^y (1 - Q_i)^{(N-y)} \\ P(n_{ij} = z) &= \begin{pmatrix} N \\ z \end{pmatrix} (P_{ij})^z (1 - P_{ij})^{(N-z)} \end{aligned}$$

Putting everything together, we can then estimate the expected mutual information according to sample size as follows.

$$\begin{aligned} E[\hat{I}(a_1, a_2)] &= \sum_{i=1}^{20} \left[\sum_{x=0}^{N} \frac{x}{N} \log_2(\frac{x}{N}) \begin{pmatrix} N \\ x \end{pmatrix} (P_i)^x (1 - P_i)^{(N-x)} \right] \\ &+ \sum_{j=1}^{20} \left[\sum_{y=0}^{N} \frac{y}{N} \log_2(\frac{y}{N}) \begin{pmatrix} N \\ y \end{pmatrix} (Q_j)^y (1 - Q_j)^{(N-y)} \right] \\ &- \sum_{i=1}^{20} \sum_{j=1}^{20} \left[\sum_{z=0}^{N} \frac{z}{N} \log_2(\frac{z}{N}) \begin{pmatrix} N \\ z \end{pmatrix} (P_{i,j})^z (1 - P_{i,j})^{(N-z)} \right] \end{aligned}$$

Note that this formula is an approximation. The n_i, n_j, and n_{ij} terms are treated as independent of each other. In reality, they are not independent because of such constraints as $\sum_i \sum_j n_{ij} = N$. However, while this equation might not be entirely realistic, it merits inspection for assessing the behavior of expected mutual information by sample size. Later in this section, we introduce *excess mutual information*, an alternative formulation that avoids this problem.

Figure 7.4 shows this expected mutual information as a function of sample size, calculated according to these equations with the observed probability distributions used to approximate their actual probability distributions. The expected mutual information starts at zero when the sample size is zero, and reaches a peak value of 2.26 bits with a sample size of 17. To understand why entropy is overestimated at small sample sizes, consider the expected estimates of entropy as shown in Figure 7.5. $\hat{H}(a_1)$ and $\hat{H}(a_2)$ both climb fairly quickly to their true values; they involve only twenty equivalence classes. At very small sample sizes, expected mutual information is high because $H(a_1)$, $H(a_2)$, and $H(a_1, a_2)$ all suffer from oversampling. In constrast to $H(a_1)$ and $H(a_2)$, $\hat{H}(a_1, a_2)$ involves 400 equivalence classes; a larger set of samples is needed derive an accurate estimate of the underlying probability distribution $P(a_1, a_2)$. This is a classic pitfall of *maximum likelihood methods*, methods in which probability distributions are estimated solely as a function of observed counts.

Having recognized the problem, we still need a method to factor out sample size when analyzing observed mutual information. In theory, we could apply the formula above to compute the expected mutual information for the sample size, and compare that to the observed value. In practice, direct application of the formula is viable for only small sample sizes, as it involves multiplying very large combinatorial terms with very small probability terms. This problem can be reduced by adding the logarithms of the two quantities rather than multiplying the quantities, but this is not a complete solution. As the combinatorial terms become very large and the probability terms become very small, one cannot add the logarithm of a very small probability to the logarithm of a very large combinatorial term without losing precision. Eventually, precision errors make the equation too inaccurate for direct application. A better alternative is to select random samples of the desired sample size, measure the mutual information of the randomly-chosen subset, and repeat the procedure enough times to yield a statistically-viable estimate.

However, when we study pairs in only buried or only exposed positions, there is a second problem with comparing their mutual information with that of all pairs. The distribution of amino acids in proteins varies as exposure varies. Buried positions tend to contain a greater proportion of hydrophobic amino acids, and exposed positions contain a greater proportion of charged amino acids. There are circumstances in which we would want to compare mutual information of buried pairs with that of equally-sized subsets of randomly-chosen pairs, but such comparisons do not tell us if the buried pairs have significant mutual information. Instead, they tell us how the strength of the pairwise signal changes as burial or exposure changes. Such comparisons are made within this section. To derive estimates of expected mutual information given sample size and joint probability distribution, I measured the mutual information of 1000 randomly-chosen samples of specified sample size.

Thus, to determine if the observed mutual information in a subset is significant, we need to compare it to the expected mutual information given both the sample size and appropriate probability distributions. In this work, we are focusing on the significance of pairwise preferences, and particularly if observed contacts are actually independent. Therefore, the correct way to assess significance of observed mutual information is to compare it to the mutual information expected given pairwise independence. If two quantities are independent, the probability of seeing any pair is the product of the probabilities of seeing the two members of the pair: $P(x, y) = P(x)P(y)$. To derive an estimate of the expected mutual information of the amino acid pair (a_1, a_2) if a_1 and a_2 are independent, we collected the set of observations and scrambled the order of a_2 observations with respect to the a_1 observations. This yields a set of observations with the same marginal distributions and the same sample size. We measured the mutual information on this sample, and repeated the test enough times to yield a statistically-viable result. We call this quantity *independent information* $I_I(a_1, a_2)$.

	Sampled mutual information (bits)	
Number of samples	Mean	Standard Deviation
10	0.0954	0.0039
100	0.0948	0.0059
250	0.0943	0.0065
500	0.0939	0.0064
1000	0.0943	0.0064
10000	0.0941	0.0063

Table 7.5: Mean values and standard deviations obtained when sampling expected mutual information given sample size and assuming pairwise independence

One advantage of comparing this approach is it yields a measure of the significance of the observed mutual information. One can consider the observed mutual information $\hat{I}(a_1, a_2)$ to be *significant* if, with high probability, $\hat{I}(x, y)$ is larger than the mutual information expected given pairwise independence and small sample size effects. To estimate this probability, while we measure independent information $I_I(a_1, a_2)$, we observe the proportion of the times that $\hat{I}(x, y)$ is greater than the mutual information measured on the permuted data. This yields a probability that the observed mutual information exceeds what would be expected given pairwise independence and due to sampling error: $P(\text{observed} > \text{expected given independence})$. By subtracting the independent information $I_I(x, y)$ from the observed mutual information $\hat{I}(x, y)$, we estimate the *excess mutual information* $I_E(x, y)$, the amount of mutual information beyond the amount that would be expected given sample size effects.

Exactly when is a result statistically viable? In informal terms, one has enough samples to yield a statistically viable result if adding random samples is not likely to alter the result. I determined this empirically by varying the number of samples measured and comparing the results. Table 7.5 shows the mean and standard deviation for sampled or mutual information of buried tertiary pairs for various numbers of samples. Looking at this table, we see that as few as 250 samples yields a stable approximation of the mean. Sampling for much more than 250 samples did not yield very different information, merely required additional computing resources. Therefore, I chose an approximation obtained with 500 samples.

In summary, given an observed mutual information, there are two estimates of the mutual information to consider.

1. First is the mutual information that would be expected given the sample size and if the interacting residues are independent. This estimate is calculated by permuting the order of observations of one amino acid relative to the other: independent information $I_I(a_1, a_2)$.

2. Second is the mutual information that one would expect from a same-sized sample of the full dataset. This estimate is arrived at by selecting observations from the dataset at random. This estimate becomes interesting when one is studying pairs chosen for some environmental factor, such as buriedness. It can be interesting to compare the pairwise information in an environmentally-chosen subset to that in the full dataset. However, one cannot compare observed mutual information values directly because the subset is smaller, and therefore will tend to have greater mutual information by small sample effects. By comparing the mutual information in the environmentally-chosen subset to the expected mutual information for a subset of the same size chosen at random, we can compare the strength of the pairwise signals.

7.3.3 Analysis of the pairwise information in buried and exposed subsets

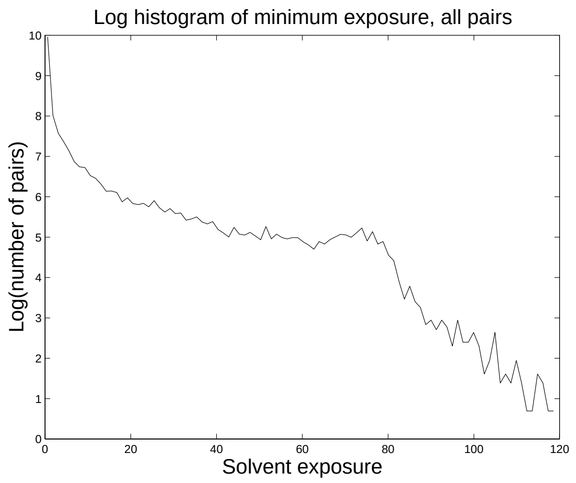
As mentioned previously, hydrophobic forces, not pairwise contact propensity, is considered the dominant factor in protein folding [34]. When hydrophobic residues are buried in a protein core, their stability depends more on burial than on identity of their neighbors. Charged residues, which should have a distinct interaction pattern, are more common in exposed positions. This suggests that pairwise mutual information should be greater in exposed positions and less in buried positions.

To test this hypothesis, we studied the mutual information of interacting pairs as a function of their burial and exposure. We defined “their burial or exposure” to be the minimum of their two solvent exposure values. This divided the dataset into two subsets: those for which the solvent exposure of the more buried residue was less than a threshold value, and those for which the solvent exposure of both residues was greater than the threshold value. In preliminary work, we divided the data into four subsets according to the solvent exposure of each residue, and observed that studying the minimum exposure yielded a similar pattern with reduced experimental complexity. Figure 7.6 shows a histogram of the minimum solvent exposure values for all pairs and for the tertiary pairs. Note that for both cases, the vast majority of all pairs have at least one residue fully buried, and a fair number have one residue partially buried. The x-axes of these histograms are scaled according to the data: there are pairs which both have a solvent exposure above 120, though almost all have a minimum exposure of 80 or less for all pairs, or 60 or less for the tertiary pairs. Note further that the full dataset contains more exposed pairs than the tertiary dataset. This is mostly due to pairs of residues in adjacent turns of exposed helices.

To see which exposure threshold yielded the most interesting division of buried versus exposed, we tested a range of thresholds from 0 to 80 for the full dataset, and from 0 to 60 for the tertiary dataset. For each threshold, we divided the data into buried and exposed subsets. For each subset, we measured the observed mutual information, and compared it to the mutual information we would expect given the sample size, with two different estimates of expected mutual information.

1. Independent information $I_I(a_1, a_2)$, the mutual information expected assuming pairwise independence was calculated by scrambling the order of observations of the second amino acid relative to the first, as described in Section 7.3.2.

2. The mutual information expected if the probability distribution did not vary with exposure reflects the amount of mutual information of a same-sized subset of pairs chosen randomly from the dataset. If pairwise preferences are less distinct in buried positions and more distinct in exposed positions, then the observed mutual information should be less than this estimate in buried positions and greater than this estimate in exposed positions.



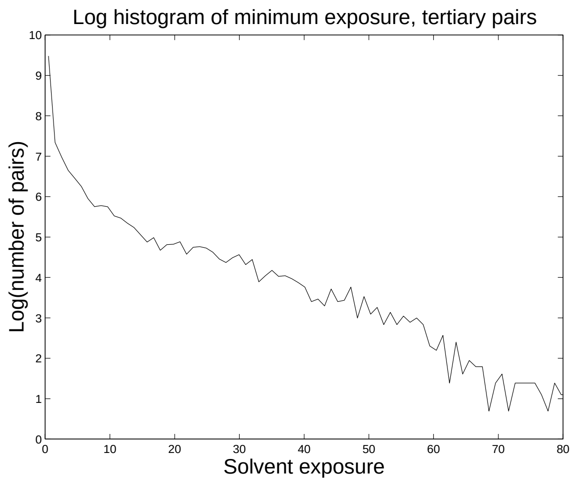
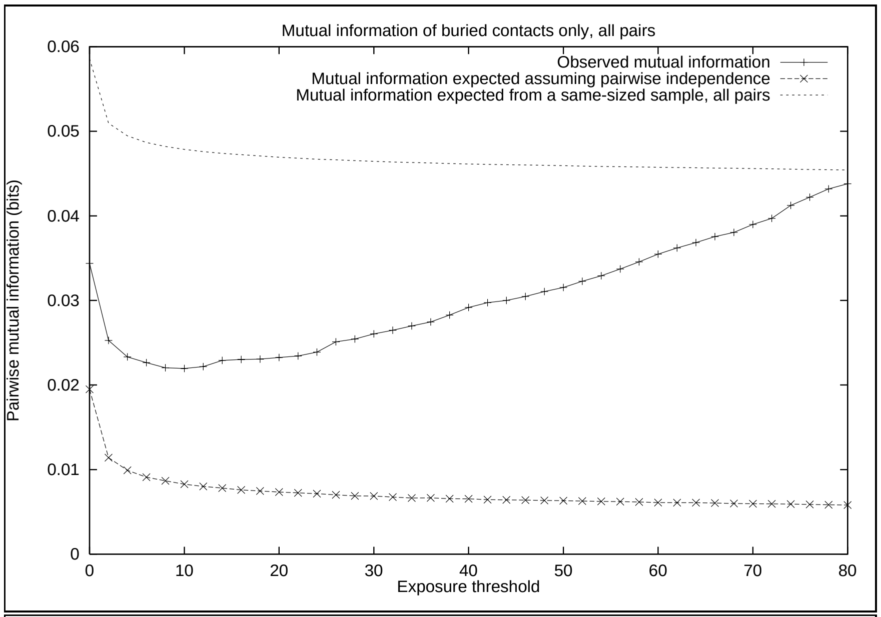


Figure 7.6: We considered the exposure of a pair to be the minimum of the two exposure values. These histograms show the ranges of minimum exposure values for all pairs in the dataset (top) and for the tertiary pairs only (bottom). The x-axes are scaled according to the range of the data.



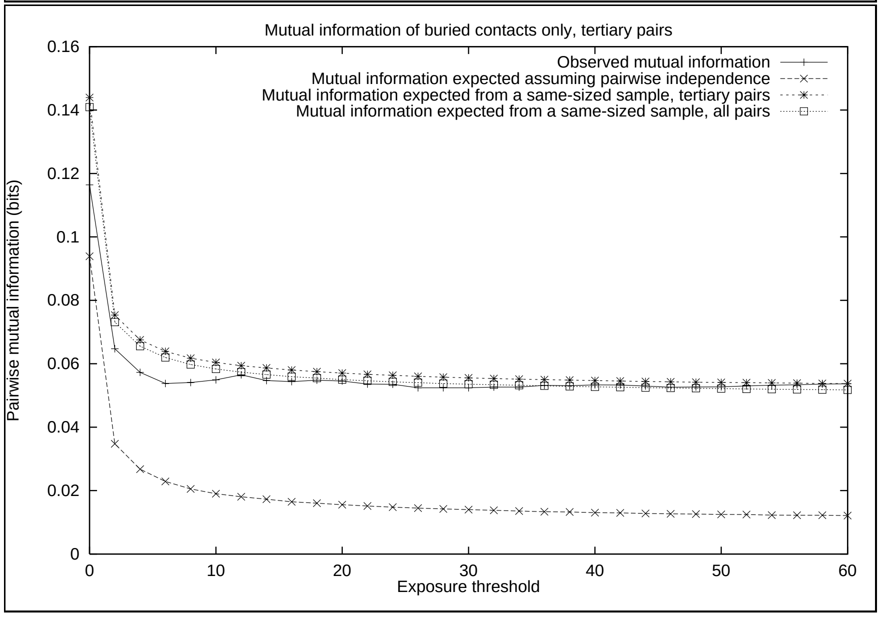
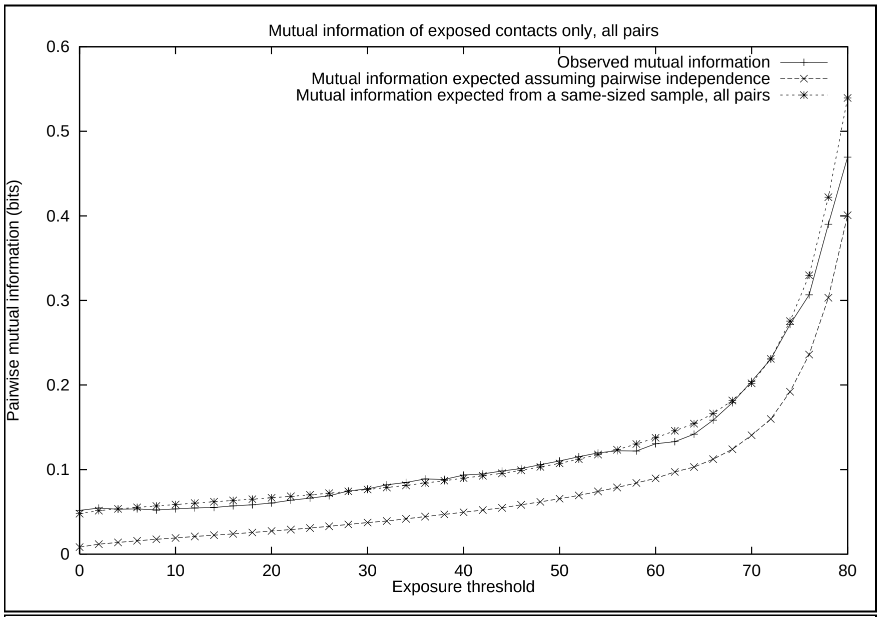


Figure 7.7: Comparison between observed and expected mutual information in buried positions: all pairs (top) and tertiary pairs (bottom). A pair is defined as buried if the solvent exposure of the more buried residue is less than the exposure threshold.



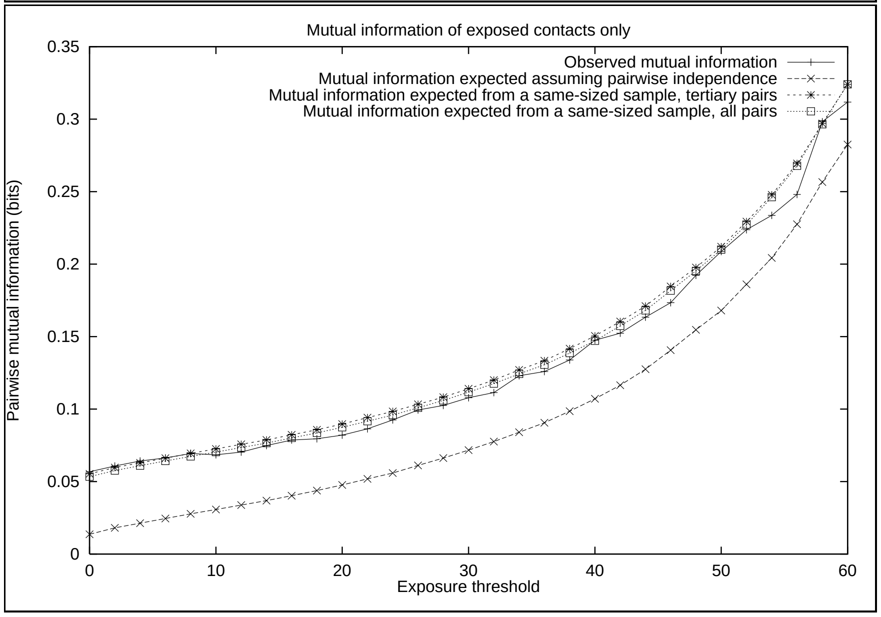


Figure 7.8: Comparison between observed and expected mutual information in exposed positions: all pairs (top) and tertiary pairs (bottom). A pair is defined as exposed if the solvent exposure of the more buried residue is greater than the exposure threshold.

Pairwise mutual information given burial or exposure

Figure 7.7 compares the observed and expected mutual information for buried positions for all pairs and for the tertiary pairs only. Figure 7.8 makes the same comparison for exposed pairs.

In all figures, we note that the observed mutual information is substantially greater than what would be expected if the interactions were independent of amino acid type. In every case, the observed mutual information was greater than the independent information $I_I(x,y)$, the mutual information sampled by scrambling the order of observation of the second residue with respect to the first. With 500 scramblings of the order of observation of the two residues, we can say that there is at least a $\frac{499}{500}$ chance that the observed mutual information is *significant*, greater than what would be expected given simply small sample-size effects. By this analysis, there is little doubt that pairwise preferences are significant.

However, pairwise preferences appear to be less distinct in buried positions, especially for short-range interactions. In Figure 7.7, when we compare the mutual information in observed buried positions to that expected from a same-sized sample drawn without regard to burial, the observed pairwise information drops to about half of the expected value for the set of all pairs, including short-range interactions. This gap is at its widest at an exposure threshold of 8. For tertiary pairs, we also see a gap between observed and expected mutual information at this point, though the gap is far less pronounced. One would expect that the pairwise signal decreases in buried positions, as hydrophobic contacts tend to be less specific than polar contacts; in that sense, this data is not surprising. What is more surprising is that long-range interactions appear to be less affected by hydrophobic forces than short-range interactions.

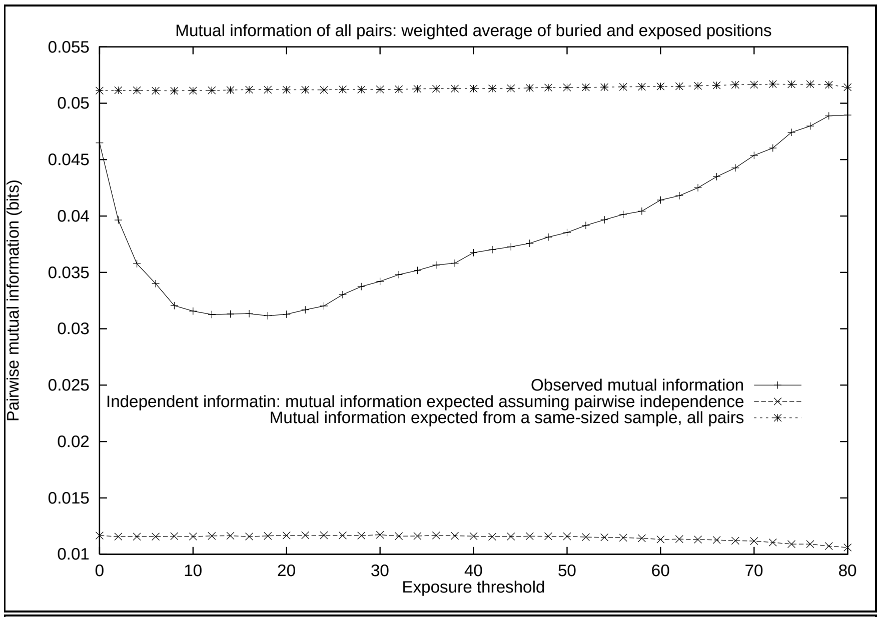
In Figure 7.8, we see that the amount of pairwise information does not vary much according to exposure threshold. For both graphs, at the most exposed positions, the observed mutual information appears to be slightly lower than what would be expected given the sample size and the background pairwise distribution. However, this decrease is probably not significant; at high exposure values, the sample sizes are so small that what we observe is probably just noise in the data.

Mutual information of long-range versus short-range contacts

In Figures 7.7 and 7.8, we see that when we focus on tertiary contacts, the mutual information appears to increase. However, most of this appears to be sample-size effects. When we factor out sample size by comparing the observed mutual information between tertiary pairs to the mutual information expected from a same-sized sample drawn from all pairs, there is only a marginal increase in mutual information. This suggests that tertiary contacts are slightly more interesting than shorter-range contacts, but only slightly.

Overall mutual information conditioned on exposure

Next, we computed the overall mutual information as a weighted average of the buried and exposed subsets, weighted according to probability of burial or exposure. Figure 7.9 shows the overall mutual information observed, expected assuming pairwise independence, and expected from a same-sized sample drawn independent of exposure. The first point is that something different appears to be at play between the short-range contacts and the long-range contacts. In pairs that include short-range contacts, there is a pronounced dip in buried positions in *excess mutual information:* information in excess of what would be expected given the sample size and assuming pairwise independence, $I_I(a_1, a_2)$. Tertiary pairs also show a dip in excess mutual information in the buried positions, but the dip is far less pronounced. Excess mutual information is highlighted in Figure 7.10. In this figure, we see that while there are exposure thresholds for which almost half of the excess mutual information is lost for the larger set of pairs, there is always at least 0.02 bits of excess information. In the tertiary pairs, there is always at least 0.035 bits of excess mutual information.



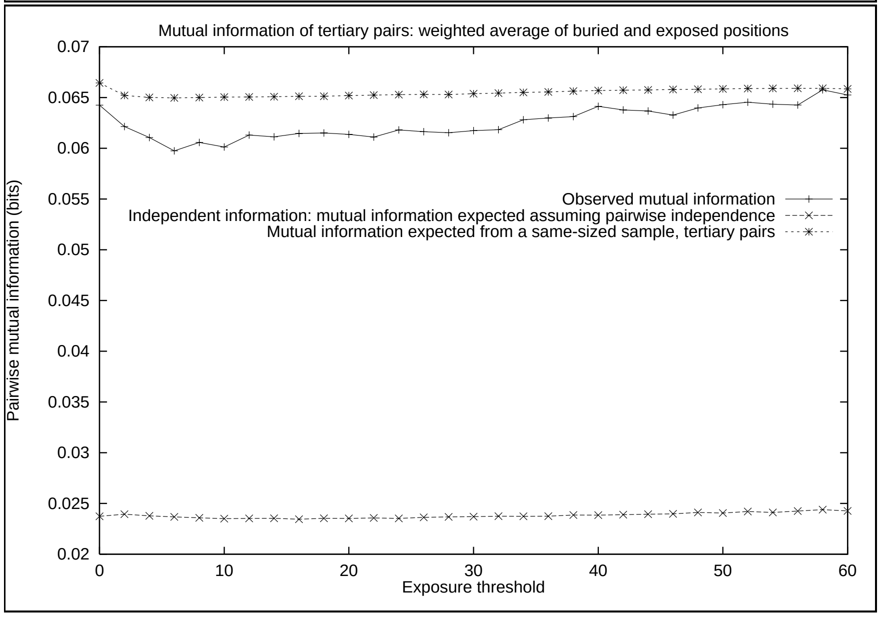


Figure 7.9: Overall mutual information conditioned on exposure: all pairs (top) and tertiary pairs (bottom). A pair is defined as exposed if the solvent exposure of the more buried residue is greater than the exposure threshold. The pairs were conditioned on exposure by weighting the buried and exposed mutual information according to probability of burial or exposure.

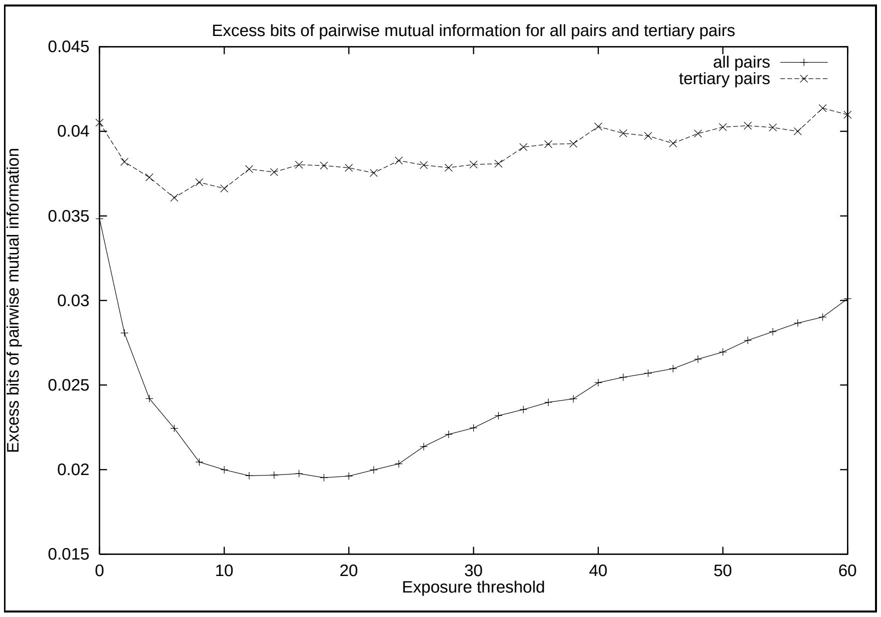


Figure 7.10: *Excess mutual information:* amount observed in excess of the amount expected by chance for an independent set of pairs of the same size. Overall excess mutual information was computed by weighing excess information in buried and exposed positions according to the probability of burial or exposure.

Second, looking at the overall mutual information for the tertiary pairs and comparing it to what would be expected given the sample size for the set of pairs including short-range interactions, we see again that the tertiary pairs contain slightly more information. At any sample size, the mutual information expected given the tertiary distribution is about 0.002 bits greater than that expected given the larger distribution.

Analysis of pairwise information by secondary structure

Next, we studied the dependence between mutual information and secondary structure. Focusing on the tertiary contacts we divided the contacts into subsets according to secondary structure type: strand-strand, helix-helix, and strand-helix. As a reminder, this dataset was created for threading purposes, and does not include loop positions. The number of pairs in each class are shown in Table 7.6.

Following the analysis in Section 7.3.3, we compared the amount of mutual information for each subset to the amount that would be expected if contact patterns exhibited pairwise independence. We estimated the amount expected assuming pairwise independence and given the sample size by permuting the order of observation of the second amino acid relative to the first. As described in Section 7.3.2, this process was repeated for 500 random permutations. To study the dependence between mutual information and exposure, we conditioned the mutual information on exposure as follows. We divided each set of contacts into buried and exposed subsets, according to a threshold on the solvent exposure of the most-buried residue in the pair. For both buried and exposed subsets, we measured the observed mutual information and estimated the expected mutual information. We arrived at an overall value by computing a weighted average between buried and exposed mutual information by sample size.

For each secondary structure class, we computed excess mutual information, the difference between the mutual information observed and expected assuming pairwise independence. The excess mutual information was conditioned on exposure for a range of exposure thresholds. This data is shown in Figure 7.11.

The first point to notice on this graph is that all classes of secondary structure contacts have excess mutual information. However, helix-helix contacts are the least-informative, while strand-strand contacts are the most informative by far. This corroborates earlier work that contacts can be meaningfully partitioned according to secondary structure type, and contacts between β-strands are most significant [30]. Another interesting point is that strand-strand contacts seem far more dependent on exposure than other types of contacts; while the other curves are fairly flat, give or take noise, the strand-strand curve exhibits a pronounced dip at an exposure threshold of 6.

Table 7.6 further quantifies the effect of secondary structure type on contact information. This table reports the observed mutual information for contacts of each secondary structure type to that of a same-sized sample drawn at random from the set of all pairs. While helix-helix contacts have approximately 25% less information than an same-sized set chosen without regard to secondary structure, strand-strand pairs have close to 50% more. It is clear that β-strand contacts are a rich

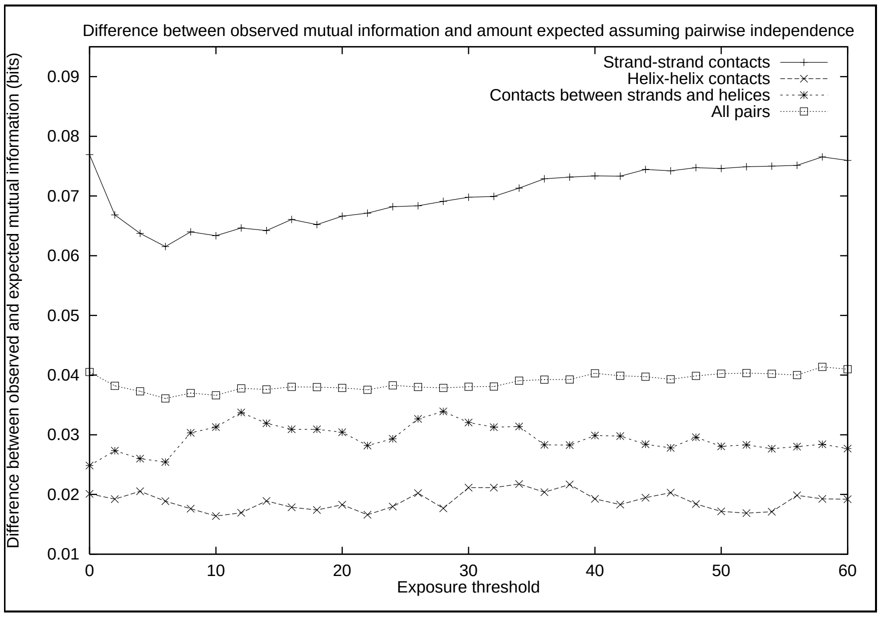


Figure 7.11: Following the analysis in Section 7.3.3, this figure shows the excess pairwise mutual information by secondary structure.

source of pairwise contact information.

7.4 Contribution of special amino acids

Certain classes of contacts make sense for special biochemical reasons:

- First, cysteines frequently interact with other cysteines, as pairs of cysteines can form disulfide bonds. Also, cysteines often cluster around metal ions. Even harsh critics of pairwise

		Mutual information (bits)	
Contact type	Number of pairs	Observed	Expected given same-sized sample, all pairs
All pairs	22707	0.0537	0.0537
Strand-strand pairs	11266	0.0997	0.0651
Helix-helix pairs	6147	0.0635	0.0856
Strand-helix pairs	5294	0.0817	0.0930

Table 7.6: Mutual information according to secondary structure types: amount observed versus amount expected for an equally-sized subset drawn at random and without replacement from all secondary structure types. Comparing the observed mutual information to the information that would be expected from a same-sized sample drawn independently of secondary structure provides some intuition on how strong the pairwise interaction signal is for each secondary structure class.

potentials accept this attraction as one genuine pairwise preference.

- Second, a few of the 20 amino acids carry a charge, and might reasonably be attracted to residues of opposite charge and repulsed from residues of like charge. The residues classified as charged are arginine, aspartic acid, glutamic acid, lysine, and histidine. Note that histidine is not always classified as charged; it carries no charge at neutral pH, but adopts a charge slightly below neutral pH.

Here, focusing on the tertiary pairs, we address the question of how much of the observed pairwise mutual information can be explained by these special forces.

Note that when a special attraction exists between two residues, those residues interact with increased frequency and those residues interact with others with decreased frequency. For example, when cysteines interact frequently with other cysteines, they do not interact frequently with the other nineteen amino acids. This aberrant lack of interaction is as much a source of pairwise mutual information as the interaction between special residues. So, when measuring the impact of special forces on pairwise mutual information, one must address both the increased frequency of interaction where the force applies and the decreased frequency of interactions where it does not. We have addressed this by comparing the mutual information between all residues with the mutual information between residues not affected by the special forces. For example, to examine the amount of pairwise mutual information explained by the cysteine-cysteine attraction, I compare the pairwise mutual information of all twenty amino acids to that of the nineteen amino acids that are not cysteine. My justifications for this approach are as follows:

- This observed change in mutual information acts as an upper bound for the signal resulting from the force. If N bits of information are lost by removing cysteine from the frequency tables, then no more than N bits of information can be attributed to the cysteine-cysteine attraction. If N is surprisingly small, then this upper bound might be sufficient to say if the force is not significant.

- A separate test can be used to determine how much of the pairwise preference of the affected residue is due to the special force. For example, one might measure the frequency with which cysteine interacts with residues other than cysteine, and test if this frequency is significantly different from the frequencies of amino acids other than cysteine. If those frequencies are not significantly different, then one could state that cysteine has no contact preference besides interacting with itself. Further, one could then argue that when one compares the mutual information between all pairs with the mutual information between all pairs not including cysteine, the difference is explained by the special attraction of cysteine to itself.

Tables 7.7 and 7.8 summarize the change in mutual information seen by removing special residues. Table 7.7 describes the bits of excess information remaining after the special residues are removed, and Table 7.8 compares the observed mutual information to the expected information in a same-sized sample drawn from the tertiary dataset.

When the cysteine residues are removed from the dataset, the bits of excess mutual information drops by 11%, and the observed mutual information drops to 10% of what would be expected from a same-sized sample drawn at random. While these percentages might not seem like a large fraction, note that only 4% of the pairs involve a cysteine residue. With 4% of the interactions representing 10% to 11% of the information, the interactions must be viewed as important. For contrast, consider interactions involving charged residues. These interactions represent 23% of the pairs in the dataset. When they are removed, the excess mutual information decreases by 32%, and the observed mutual information drops by 35% relative to a same-sized sample drawn at random. If these 23% of the interactions involve 32% to 35% of the information, then the charged residues have more pairwise signal than most residues. However, on a per-interaction basis, the cysteines are a richer source of pairwise interaction.

When both charged residues and cysteines are removed, the number of pairs decreases by 26%. The bits of excess mutual information decrease by 46%, and the observed information drops to 48% of the amount expected from a same-sized sample drawn at random. In summary, almost

Pairs Analyzed	Mutual information (bits) Observed	Mutual information (bits) Expected assuming pairwise independence	Bits Saved
All pairs	0.0537	0.0116	0.0421
Removing Cysteine	0.0482	0.0108	0.0374
Removing charged residues	0.0370	0.0082	0.0288
Removing both	0.0299	0.0073	0.0226

Table 7.7: Mutual information of interacting residues including and excluding those subject to special physiochemical forces: cysteine and the charged residues arginine, aspartic acid, glutamic acid, lysine, and histidine. Observed mutual information is compared to the independent mutual information, that expected given the sample size and assuming pairwise independence. *Bits saved* refers to the difference between these two quantities.

Pairs Analyzed	Sample Size	Mutual information (bits) Observed	Mutual information (bits) Expected given same-sized sample, all pairs	Bits Lost
All pairs	22707	0.0537	0.0537	0
Removing Cysteine	21843	0.0482	0.0536	0.0054
Removing charged residues	17510	0.0370	0.0566	0.0197
Removing both	16729	0.0299	0.0573	0.0275

Table 7.8: Mutual information of interacting residues including and excluding those subject to special physiochemical forces: cysteine and the charged residues arginine, aspartic acid, glutamic acid, lysine, and histidine. Observed mutual information is compared to the amount expected if same-sized samples are drawn at random from the set of all pairs. The second quantity will usually be larger than the first, and *bits lost* quantifies this difference.

half of all pairwise information comes from just over a fourth of the data, and interactions involving 6 of the 20 amino acids.

To further assess the contribution of these special forces to pairwise mutual information, we reduced the 20-letter amino acid alphabet to smaller alphabets according to special forces exerted by the residues. These alphabets were as follows: cysteine and non-cysteine; positively-charged, negatively-charged, and other; and positively-charged, negatively-charged, cysteine, and other. For each case, we measured the excess mutual information for each alphabet as outlined in Section 7.3.3. Figure 7.12 shows the excess mutual information for each alphabet as a function of the exposure threshold.

According to Figure 7.12, of the 0.035 bits of mutual information observed in pairwise interactions, 0.005 bits can be attributed directly to the cysteine-cysteine attraction. While 0.005

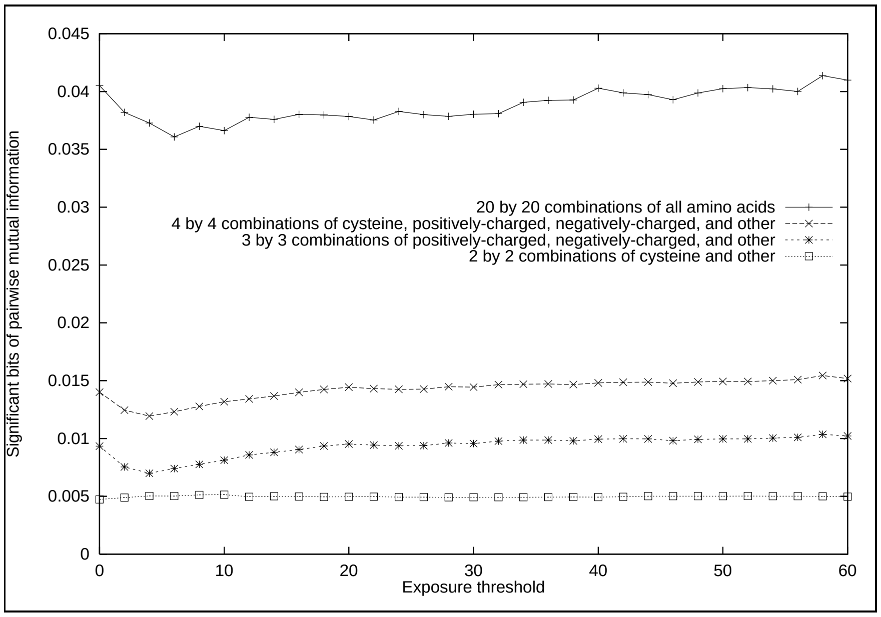


Figure 7.12: To assess the amount of pairwise mutual information explained by special forces, we reduced the 20-letter amino acid alphabet to smaller alphabets according to the special forces studied: cysteine and non-cysteine; positively-charged, negatively-charged, and other; and cysteine, positively-charged, negatively-charged, and other. Here, the excess mutual information in each case is shown as a function of the exposure threshold.

bits might not sound like a large amount, recall from Table 7.8 that only 4% of the observed pairs include a cysteine residue. Therefore, one twentieth of the data contains about one seventh of the information.

When measuring the information according to charge, we experimented with whether or not to consider histidine charged. Histidine is not charged at neutral pH, but becomes charged when the pH drops slightly. In preliminary experiments, we compared the excess mutual information obtained when we classified histidine as charged to that obtained when we classified it as neutral. When we classified it as charged, the excess mutual information was slightly larger (data not shown). Therefore, for later work, we classified histidine as charged.

In Figure 7.12, we see that charge accounts for somewhere between 0.007 and 0.010 bits of excess mutual information. At the exposure threshold of 8, interactions between charged residues account for approximately 0.0075 bits of excess mutual information, leaving the amount of pairwise information not attributable to charge at 0.0275 bits. At this same threshold value, 0.0125 bits of excess information can be attributed to special forces. This is approximately one third of all of the excess mutual information; there is only 0.0235 bits of excess mutual information beyond what can be attributed to special forces.

7.5 Summary

The pairwise mutual information of interacting amino acids is quite low, less than one twentieth of one bit. Tertiary contacts appear to be more informative than those that include short-range interactions, but most of that apparent information is merely small sample-size effects. Interactions involving Cysteine or charged residues represent almost half of the information in the pairwise distribution, even though these pairs represent only one-fourth of all interactions.

If one considers contacts that involve short-range interactions, almost half of the apparent pairwise information will be due to hydrophobic forces. Tertiary contacts are also affected by hydrophobic forces, but the effect is far less pronounced: approximately one tenth of the information in tertiary contacts can be attributed to hydrophobicity. The largest body of short-range interactions studied was residues in adjacent turns of α-helices. While pairwise contact potentials contain excess information to help predict these contacts, other approaches might be better. Modern secondary structure predictors are up to 78% accuracy [90], and α-helices are one of the easier secondary structure classes to identify [152]. Instead of using weak information between short-range contacts to predict α-helices, a better approach is to focus on the tertiary contacts and to leave the helix prediction to a secondary structure predictor.

While mutual information between tertiary pairs is limited as a whole, it is influenced heavily by secondary structure type. Contacts between residues in α-helices carry the least information; at all exposure thresholds, they carry substantially less information than other types of contacts. Contacts between β-strand residues are the most informative of all classes analyzed. Thus, if the goal is contact prediction, the most sensible strategy is to build a specialized predictor for β-strand contacts and ignore the rest.

Chapter 8

Prediction of β-strand contacts

As shown in Chapter 7, when we study amino acid contacts, the patterns of contacts between interacting β-strands are particularly interesting. Not only do β-sheet contacts yield a stronger signal than general contacts, that signal degrades less in the hydrophobic core of the protein, the portion most critical for protein structure prediction. Furthermore, a β-strand contact is usually comprised of more than one pair, so the contact information accumulates along the β-strand. Contrast this to random coil, where two residues might be close enough to interact, but their sequence neighbors might not be close enough to have any effect on each other. Thus, for the goal of harnessing pairwise contact information for alignment validation, concentrating first on β-sheet contacts is a sensible strategy.

This chapter describes study into antiparallel β-sheet contact potentials that I performed in collaboration with Albion Baucom and Lydia Gregoret [56]. Our two β-strand contact potential functions take as input two β-strand segments and associated input and estimate the probability that their central residues would be in contact. Our first contact potential function simply estimated probability of β-strand contact or no β-strand contact. The second divided up β-strand contacts into a number of different classes based on topology and accessibility patterns, and estimated the probability that the inputs were in each class of contact or not in contact.

Prior work in this area is outlined in Section 8.1, while Sections 8.2 and 8.3 describe our β-strand contact potential functions. Our findings are reported in Section 8.4 and summarized in Section 8.5.

8.1 Related work

While α-helices and β-sheets exist in proteins in approximately equal proportions [193], β-sheets are more of a mystery. Scientists are never certain whether to consider them secondary or tertiary structure phenomena. On one hand, they feature long-range tertiary contacts. On the other hand, β-strands are secondary structure elements, though they do not exist outside of β-sheets. Recent evidence suggests that β-strand formation is the first step in β-sheet formation [164]; prior beliefs were quite different[34]. Unlike all-α proteins, all-β proteins are difficult to study and to design [131]. Secondary structure prediction is harder on β-strands than α-helices [152]. While amino acid α-helix propensities are easy to estimate and apply, β-strand propensities are more complex. The stability of an amino acid in a β-strand depends on whether the strand is an edge strand or in the interior of the sheet [121], whether it is position is hydrogen-bonded to neighboring residues [168], whether the sheet is parallel or antiparallel [121, 80], and the type of amino acids nearby [193, 138]. Yet there is consensus that β-strand residues exhibit distinct preferences in pairwise interactions.

The seminal work in β-strand contact prediction was published by Tim Hubbard [75]. Under Hubbard's system, contact was predicted on the basis of one residue on the first β-strand and a five-residue segment on the second, a subset of the information used by my predictor. With this information Hubbard constructs propensities on β-strand contacts, and uses these propensities to score potential contacts. Hubbard says little about exactly how these propensities are computed. Many authors have observed different amino acid residue potentials in different topologies [110, 95, 121]. In a similar vein, Hubbard considers a number of distinct contexts separately: whether the strands are parallel or antiparallel, and whether or not the central residues are hydrogen-bonded.

Hubbard used his β-strand contact potentials for successful protein structure prediction in the CASP1 contest [76]. He made ab initio predictions by using PHD [152] to locate possible β-strands, and used his potential function to assemble these β-strands into a protein structure. In the fold recognition section, Hubbard used HMMer HMM software [43] to align the target sequences to existing folds and used his β-strand potential function to validate and select candidate alignments.

Approximately five years after Hubbard's publication, Zhu and Braun published a work quite similar to Hubbard's [197]. In their work, they divided β-strand contacts into four subsets based on if the strands are parallel or antiparallel and whether or not the central residues were hydrogen-bonded. They then measured the frequencies of each pair of amino acids in each of the following positions, as illustrated by Figure 8.1: i and j, i and $j \pm 1$, and i and $j \pm 2$. They used these frequencies as statistical energies. They do not cite the Hubbard paper, and were perhaps not aware of its existence.

Anders Krogh applied neural networks and β-strand contact prediction to ab initio structure prediction [101]. He slid two windows of nine residues along proteins, using neural nets to predict if the central residues represented a β-sheet contact pair. He did not confine his analysis to known β-strands, but instead trained his neural nets to distinguish β-strand contact pairs from everything else. He was surprised to find little correlation between contents of β-strands in contact. However, note that unlike others, Krogh used one network to represent all β-strand contacts. Perhaps his lack of observed signal was due to attempting to capture various distinct signals with one single model.

8.2 Inputs to the β-strand contact potential functions

All data used for this work has been developed in the Gregoret lab by Albion Baucom and Lydia Gregoret [56].

1aak	1aan	1aba	1acf	1add	1ade	1amp	1aor	1apa	1arc
1asq	1ast	1atl	1atp	1bam	1bbp	1bcx	1bec	1bhp	1bia
1bmd	1bnd	1bnh	1bov	1bpb	1btn	1byd	1cbg	1cca	1cdc
1ceo	1cew	1chd	1chm	1clc	1cmc	1cpt	1csn	1ctf	1ctm
1ctu	1cxi	1cyx	1daa	1dea	1dfn	1dhr	1dpg	1dri	1dsb
1dts	1dup	1dyn	1eaf	1edt	1eny	1epb	1esl	1ezm	1fba
1ffe	1fiv	1fna	1frd	1frn	1fxd	1gal	1gar	1gca	1gcb
1gd1	1gky	1goh	1gox	1gp1	1gpc	1gpm	1gpr	1grj	1gse
1gtq	1han	1hip	1hmy	1hny	1hoe	1hpl	1hpt	1hqa	1htp
1hur	1hxn	1hyh	1hyl	1ino	1irk	1knb	1kpt	1lat	1lba
1lcp	1lfa	1lfi	1lla	1llo	1ltt	1mat	1mdc	1mdt	1mhl
1mla	1mml	1mmr	1mol	1mpp	1msa	1msc	1mup	1nar	1nba
1nci	1nfp	1npx	1nsd	1oac	1omp	1onc	1opb	1opr	1otf
1otg	1ova	1ovb	1oxa	1pbn	1pbp	1pda	1pea	1pfk	1pgx
1phh	1php	1pht	1pii	1plq	1pne	1pnf	1pnk	1poc	1pox
1prn	1psp	1ptr	1pya	1pyd	1qor	1rbp	1rcb	1rec	1reg
1rhd	1rib	1ris	1rnb	1rsy	1rtp	1rve	1sac	1sbp	1sca
1sce	1ses	1shb	1shg	1slt	1smn	1smp	1spa	1sts	1tca
1tcs	1ten	1tfd	1tgx	1the	1tie	1tif	1tig	1tml	1tnd
1tpl	1trb	1trh	1trk	1tup	1ubq	1udh	1urn	1vca	1vcc
1vhh	1vmo	1vrt	1vsf	1wap	1wsy	1xad	1xxa	1xyz	2aai
2acu	2ak3	2ayh	2aza	2bbk	2bop	2btf	2cbe	2cht	2ci2
2cmd	2cpl	2ctv	2cym	2fcr	2fke	2fxb	2glr	2glt	2gn5
2gsq	2had	2hft	2hhm	2hip	2hpd	2hpr	2kau	2lao	2liv
2ltn	2mad	2mcm	2mnr	2msb	2nck	2nrd	2olb	2oxi	2pab
2paz	2pf2	2phl	2phy	2pia	2pk4	2por	2reb	2rhe	2rn2
2rsl	2rsp	2sar	2scp	2sil	2sn3	2sns	2sod	2stv	2tgi
2tmd	2trx	3adk	3b5c	3cd4	3chy	3cox	3dfr	3dni	3gap
3grs	3hsc	3il8	3lad	3ovo	3rp2	3rub	3sc2	3sdp	3tgl
4blm	4bp2	4cla	4fgf	4gcr	4htc	4icd	4tms	4ts1	5cts
5dfr	5er2	5fbp	5fd1	5fx2	5gst	5hvp	5pep	5rub	5sga
5tnc	6ebx	6lyz	6taa	7enl	7lzm	7nn9	7pcy	7tim	821p
8acn	8atc	8cat	8cpa	8cpp	8dfr	8fab	8gpb	8i1b	8rxn
8tln	9abp	9ldt	9pap	9pti	9rnt	9rsa	9wga	9xia	

Table 8.1: Contents of the Gregoret-Baucom dataset [56]

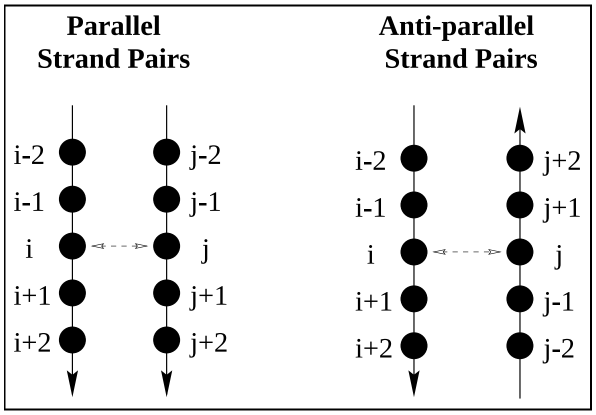


Figure 8.1: Diagram of the windowing terminology

The protein structure dataset used for these experiments contains 339 proteins with a maximum of 30% homology. All structures have a resolution of 2.3 angstroms or better, and all redundant chains were removed from the protein structure definition. The structures contained in this dataset are listed in Table 8.1.

In addition to the set of structures, Baucom and Gregoret have provided us with *topological accessibility* measurements for all proteins in the set. Topological accessibility measures solvent accessibility by stripping off the side-chains and using a sphere of somewhat larger diameter compared to what is used in standard accessibility computations; the standard calculations use a sphere that approximates the size of a water molecule..

The primary input for the potential function is sequence data. Given two β-strand residues which might be in contact, we extracted a window of five residues from each strand, centered at the possible contact pair. These central residues are referred to as i and j, and the windows containing them are referred to as the *i-window* and *j-window*. By convention, the *i-window* appears in the sequence before the *j-window*. The residues in the *i-window* are referred to, in sequence order, as *i-2, i-1, i, i+1, i+2*, and the residues in the *j-window* are referred to as *j-2, j-1, j, j+1, j+2*. This terminology is illustrated in Figure 8.1.

While these windows represent known β-strands, not all ten residues are required to be

β-strand residues. Both i and j must be β-strand residues, and both windows must contain at least two additional β-strand residues. This permits making predictions near the end of strands and on short strands. A further note is that this predictor did not address β-bulges; all strands used in this investigation were unbulged.

In addition to the sequence data, we looked at certain environmental data: accessibility, strand membership information, and *chain separation*, the number of residues between i and j in the protein sequence.

8.2.1 Representation of the sequence data

Neural networks, described in Section 7.2.1, expect that each input represents a distinct signal, and different values of that input represent different strengths of the signal. Thus, each input should have one distinct meaning rather than each value of the input having a distinct meaning. For example, if input i was to represent an amino acid i with value 1 representing $i =$ Ala, value 2 representing $i =$ Arg, and so forth, that would not be a good representation. A better representation is to have 20-input bit vector to indicate which amino acid residue i is: bit 1 is set if $i =$ Ala, bit 2 is set if $i =$ Arg, and so forth. Another reasonable representation is to select a set of characteristic properties to describe each amino acid. Rather than seeing the amino acid type, the net would see a set of values describing properties of that amino acid.

Both of the good representations described above have pitfalls. For the characteristic property representation, the properties must be chosen wisely. If the relevant information is not in the property set, neural net performance will suffer. The pitfall of the bit vector representation is that it leads to a lot of inputs, and hence a lot of free parameters. Unless the training set is very large, the net might not have enough data to estimate all of the free parameters, and thus might not be able to learn as well. Overall, the best course of action is to try both representations and choose the one which works better. Preliminary work has suggested that the representative property method is better for this problem. We have used a set of twelve properties hand-picked by

Lydia Gregoret, listed in Table 8.2. Later experiments used a reduced set of six properties selected from the twelve by Lydia Gregoret (data not shown). This reduced set of properties was referred to as *properties lite*.

8.2.2 Additional inputs

In addition to sequence data, we experimented with including as inputs topological accessibility, strand membership information, and *chain separation*, the distance on the protein chain between the i and j residues, measured in number of amino acids.

Note that determining accessibility and strand membership requires structural information. Since this potential function was to be used in a structure prediction context, actual accessibility and strand membership were not technically legitimate inputs. The following, however, would be legitimate inputs:

- Predicted accessibility and strand membership, as generated by a program such as PHD [152], or
- Predicted accessibility and strand membership of the target structure, as inferred from the template structures and a predicted alignment of the target and template sequences.

Because both options add complexity to the experimental design, we decided to start with the "cheating inputs", the actual accessibility and strand membership information, to see if they were useful. Because the predicted values would probably not be as informative as the actual values, we decided that we would incorporate strand membership or accessibility predictions only if the actual inputs gave the potential function a decided edge.

8.2.3 Construction of the false examples

Construction of the false examples is designed to mimic the natural protein folding process of each segment of the sequence choosing to be in contact with some other segment. For each actual

	Amino Acid						
Property	**A**	**R**	**N**	**D**	**C**	**E**	**Q**
Anti-parallel β-Sheet Propensity	0.9	0.82	0.54	0.54	1.19	0.85	0.85
Aromaticity $*$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
β-Strand Propensity $*$	0.83	0.93	0.89	0.54	1.19	0.37	1.1
β-Turn Propensity $*$	0.67	0.95	1.56	1.46	1.19	0.74	0.98
Bulkiness	11.5	14.28	12.82	11.68	13.46	13.57	14.45
Charge $*$	0.0	1.0	0.0	-1.0	0.0	-1.0	0.0
Free Energy of Transfer	1.24	0.24	0.77	0.71	1.6	0.65	0.65
H-Donor	0.0	1.0	1.0	0.0	0.0	0.0	1.0
H-Acceptor	0.0	0.0	1.0	1.0	0.0	1.0	1.0
Parallel β-Strand Propensity	1.0	0.62	0.53	0.53	1.13	0.48	0.48
Polarity $*$	8.1	10.5	11.6	13.0	5.5	12.3	10.5
Volume $*$	0.68	0.67	0.59	0.56	0.59	0.63	0.64

	Amino Acid						
Property	**G**	**H**	**I**	**L**	**K**	**M**	**F**
Anti-parallel β-Sheet Propensity	0.56	1.12	1.54	1.26	0.82	1.19	1.4
Aromaticity	0.0	0.5	0.0	0.0	0.0	0.0	1.0
β-Strand Propensity $*$	0.75	0.87	1.6	1.3	0.74	1.05	1.38
β-Turn Propensity $*$	1.56	0.95	0.47	0.59	1.01	0.6	0.6
Bulkiness	3.3	13.69	21.4	21.4	15.71	16.25	19.8
Charge $*$	0.0	0.5	0.0	0.0	1.0	0.0	0.0
Free Energy of Transfer	1.24	1.01	1.48	1.36	0.0	1.3	1.36
H-Donor	0.0	0.5	0.0	0.0	1.0	0.0	0.0
H-Acceptor	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Parallel β-Strand Propensity	0.79	0.38	2.6	1.42	0.62	1.13	1.17
Polarity $*$	9.0	10.4	5.2	4.9	11.3	5.7	5.2
Volume $*$	0.58	0.64	0.83	0.83	0.74	0.71	0.74

	Amino Acid						
Property	**P**	**S**	**T**	**W**	**Y**	**V**	
Anti-parallel β-Sheet Propensity	0.42	0.87	1.3	1.4	1.68	1.53	
Aromaticity $*$	0.0	0.0	0.0	2.0	1.3	0.0	
β-Strand Propensity $*$	0.55	0.75	1.19	1.37	1.47	1.7	
β-Turn Propensity $*$	1.52	1.43	0.96	0.96	1.14	0.5	
Bulkiness	17.43	9.47	15.77	21.67	18.08	21.57	
Charge $*$	0.0	0.0	0.0	0.0	0.0	0.0	
Free Energy of Transfer	0.89	1.01	0.95	1.24	0.83	1.42	
H-Donor	0.0	1.0	1.0	0.5	1.0	0.0	
H-Acceptor	0.0	0.0	0.0	0.0	0.0	0.0	
Parallel β-Strand Propensity	0.35	0.7	0.59	1.17	1.08	2.63	
Polarity $*$	8.0	9.2	8.6	5.4	6.2	5.9	
Volume $*$	0.7	0.58	0.66	0.71	0.68	0.79	

Table 8.2: The amino acid property set by Webster, Nambudripad, and Buturovic [190]. $*$ denotes properties retained for the *properties lite* property set.

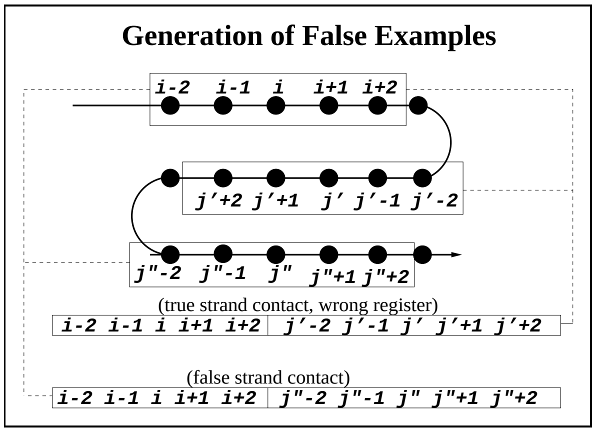

Generation of False Examples
i-2 i-1 i i+1 i+2
j'+2 j'+1 j' j'-1 j'-2
j"-2 j"-1 j" j"+1 j"+2
(true strand contact, wrong register)
i-2 i-1 i i+1 i+2 | j'-2 j'-1 j' j'+1 j'+2
(false strand contact)
i-2 i-1 i i+1 i+2 | j"-2 j"-1 j" j"+1 j"+2

Figure 8.2: Generation of the false examples, showing two different types of false contact

i residue, another j residue chosen at random according to constraints:

- Both i and j are β-strand residues,
- i and j are part of the same protein, and
- According to the DSSP program [88], i and j are not interacting in a β-sheet contact.

Data from i and j are then combined to form a single false example. Note that no preference is given to the location of j relative to the actual contacts of i: this false contact could involve two β-strands that are not in contact, or two β-strands in contact with a different register shift. These different types of false examples are illustrated in Figure 8.2.

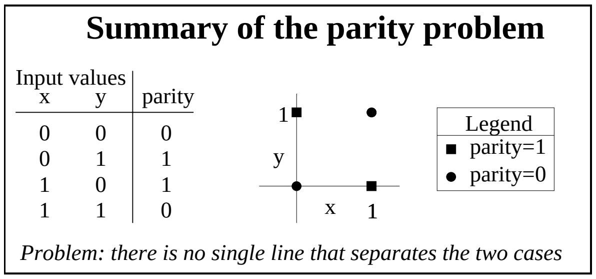
8.3 Architecture of the potential function

The neural net was trained to distinguish true from false examples based on the input. Both true and false examples are both divided into a training set and cross-training set, where the training set was used for adjustment of the weights. The cross-training examples were not seen during training, but overall cross-training performance was used to adjust the learning rate and to decide when to stop training. No separate validation set was used. The training algorithm used softmax and a maximum likelihood learning rule [18], an adaptive learning rate [63], and weight decay [14].

8.3.1 Motivation for classifying β-strand contacts

Initial work was done with a potential function that estimated two values: the probability that the strands were in contact, and the probability that they were not. However, early results showed that the actual contact was often not the top scoring one; often the top score went to the *(i, j+1)* or *(i, j-1)* contact [12]. On consideration of neural net learning of the parity function, this behavior made some sense.

Given two binary-valued inputs, their parity should be 1 if both input values are the same and 0 if they are different. This simple function cannot be learned by a single layer neural net. The equation used by a single layer neural net is $w_1x + w_2y + b \geq \mathit{threshold}$: the equation of a line if there are two inputs or of a hyperplane if there are three or more inputs. As shown in Figure 8.3, there is no single line that can separate inputs for which parity=0 from those for which parity=1. A neural net with hidden units can address the parity problem, but assuming a finite number of training examples, the hidden layer will not be able to address other functions as well. If the problem posed to the neural net involves some intricate function on top of the parity function, the neural net might not be able to learn either part of the problem.


Summary of the parity problem

Input values		
x	y	parity
0	0	0
0	1	1
1	0	1
1	1	0

Problem: there is no single line that separates the two cases

Figure 8.3: Illustration of why the parity problem cannot be solved by a single layer neural net

The characteristic of β-sheets that makes this problem analogous to the parity problem is the *bipolar* nature of many β-strands. Due to packing constraints, the side chains of adjacent β-strand residues point in opposite directions; if you think of a β-sheet as a horizontal surface, alternate rows of side chains point up and point down. Many β-sheets are exposed to the solvent on one side only, and the strands of these sheets are characterized by an alternating pattern of hydrophobic and hydrophilic side chains. If two β strands are correctly oriented, then hydrophobic residues are in contact with other hydrophobics and hydrophilics are in contact with other hydrophilics. Given that *(i, j+1)* or *(i, j-1)* often has a higher score than *(i,j)*, the neural nets did not seem to be learning this pattern.

To address this problem, we divided the realm of β-strand contacts into classes based on exposure and topology. We then built neural nets to estimate the probability with which an example belongs to each class.

The classes of contacts

Each window was classified as edge, buried, or bipolar. The central residues were classified as buried or exposed. With two strands, this classification scheme yields six attributes per window pair. Not all combinations of inputs occur naturally; for example, if a window is buried, the central residue will always be buried. Additionally, not all combinations occur frequently enough to permit statistical analysis. Certain less-frequent classes were added to the "other" class. The final list of classes is as follows:

- Both the *i-window* and *j-window* are bipolar; residue i is exposed.
- Both the *i-window* and *j-window* are bipolar; residue i is buried.
- The *i-window* is bipolar, the *j-window* is an edge strand, residue i is buried.
- The *i-window* is bipolar, the *j-window* is on an edge strand, residue i is exposed.
- The *i-window* is on an edge strand, the *j-window* is bipolar, residue j is exposed.

- The *i-window* is on an edge strand, the *j-window* is bipolar, residue j is buried.
- Both the *i-window* and *j-window* are buried.
- The *i-window* is buried, the *j-window* is on an edge strand.
- The *i-window* is on an edge strand, the *j-window* is buried.
- Both the *i-window* and *j-window* are on edge strands.
- Other

In addition to the eleven classes shown, there is one "false class" representing β-strands not in contact. This is equivalent to the false set in the previous experiments. When the classifier is presented with an input example, it estimates the probability with which the example falls into each class, including the false class.

Handling of unlabeled true examples

As labeling true examples was a manual and labor-intensive process, most of the true examples were not labelled. Therefore, there were three categories of input examples:

1. Examples known to represent false contacts,
2. True examples of known classes, and
3. True examples of unknown classes.

During training, when the neural net was presented with an example from some known class, it adjusted the weights according to the standard learning rule. When it is presented with an unknown example, it estimates the probability with which the example belongs to each class and adjusts the weights according to the normal learning rule scaled by the estimated probability with which the example belongs to the class. This scheme is called *competitive learning* [79].

8.4 Results

8.4.1 Results on true or false classification

This section summarizes the results obtained with a potential function that estimates two values: the probability that the inputs represent a β-strand contact and the probability that they do not.

We built potential functions for antiparallel β-strand pairs with a variety of inputs, and tested their performance at discriminating between true and false examples in the cross-training set. These results are summarized in Table 8.3. Note that all results shown in this table were obtained with the same number of training examples. One would expect results obtained with smaller input sets to perform better, as there are more training examples per free parameter. However, the strong performance shown by chain separation is significant.

The strong performance of chain separation is likely an unintended outcome of false set construction. As stated earlier, false contacts are built by taking one residue i and choosing at random some residue j from the same protein such that the two residues must both be in some β-sheet contact, but cannot be in contact with each other. Close residues are more likely to interact than distant neighbors; this is the motivation behind studying chain separation [90]. So, true contacts are likely to be close in the sequence compared to randomly-chosen residue pairs. Thus, the neural net was probably relying too heavily on chain separation: predicting the likelihood of contact solely on the closeness of i and j. While it might be true that distant residues are less likely to interact than close ones, predicting solely on this point would not yield a useful potential function. While we chose to not using chain separation as an input, a reasonable workaround would be to build a false set with the same chain separation probability distribution as the true set.

While strand membership seems to be a powerful input, this is probably also an unintended consequence of experimental design. Examination of the weight matrix showed that the neural nets focus on strand membership whenever it is available. This led us to suspect that the neural nets

Properties	Accessibility	Strand Membership	Chain Separation	Number of inputs	Accuracy in True-False Discrimination (%)
X	X	X	X	151	87.45
X	X	X		150	81.44
X	X			140	62.11
X		X	X	131	86.45
X		X		130	80.71
X			X	121	80.74
X				120	63.32
		X	X	11	86.73
		X		10	76.10
			X	1	80.53

Table 8.3: Percent accuracy in true/false contact discrimination with a single class for true contacts. Each row represents a separate experiment, and each X indicates features used in that experiment.

were merely looking for symmetry between the *i-window* and *j-window* strand membership. For example, with antiparallel strand pairs, if residues *i* through *i+2* were strand members and residues *j-2* through *j* were strand members, then then the two strands might contain three interacting residues. In contrast, if the strand members were *i-2* through *i* and *j-2* through *j*, the only place the windows could make contact is at *i* and *j* and strand contact would not be likely. Again, while this strand membership pattern might yield genuine information, it would not yield a useful potential function.

Finally, we noted that accessibility did not appear to be a valuable input. As shown in Table 8.3, results of experiments including accessibility were not very different from results of the analogous experiments not including accessibility. Further note that these results were obtained with actual accessibility. As described in Section 8.2.2, we planned to use actual accessibility in preliminary work to see if it proved valuable; if so, we would replace it with predicted accessibility. Since predicted accessibility would likely not be as informative as actual accessibility, we decided to omit accessibility in future experiments.

Therefore, in later work, we chose to concentrate on patterns of amino acid identity or properties, and omit strand membership, chain separation, and accessibility.

8.4.2 Results with multiple classes of true contacts

This section summarizes the results obtained using a potential function that estimates contact probability according to multiple classes of contacts.

As described in Section 8.4.1, we had decided to no longer use chain separation, strand membership information, or accessibility because earlier work suggested they did not yield a useful potential function.

To reduce network complexity, we reduced the size of the property set by selecting from the twelve inputs six which seemed most useful and representative in biological terms. This *properties-lite* set was chosen by Lydia Gregoret, and initial results (not shown) suggested it was as effective as the larger property set [12]. In addition to properties-lite, we experimented with two other representations of sequence information: a bitvector representation of amino acid identity, and *polarity*. Polarity is one of the properties in the properties-lite set, and one which should be especially valuable. Beta-sheets frequently have a bipolar structure, with alternating hydrophobic and polar residues. In bipolar sheets, polar residues are in contact with other polar residues and hydrophobics are in contact with other hydrophobics. To estimate how much of the observed information could be attributed to polarity, we experimented with using only polarity. Note that the polarity value used was not binary, so more information was conveyed than simply if a residue was hydrophobic or polar.

Table 8.4 presents the accuracy of the potential functions in discriminating true from false contacts in the cross-training set. Here, the probability of contact was obtained by summing the probability for each of the classes of contact.

Overall performance was modest. Looking at Table 8.4, and comparing them with those in Table 8.3, we see that the potential function with that worked with multiple contact classes did not achieve the same performance as the one that used a single class of contact. The simpler potential function achieved an accuracy of 63% when using as input only data derived directly from

Inputs	Accuracy in True-False Discrimination (%)
Properties-lite	59.26
Amino Acid Identity plus Properties-Lite	62.84
Amino Acid Identity	61.91
Polarity	62.06

Table 8.4: Percent accuracy in true/false contact discrimination with multiple classes of true contacts.

the sequence; no results with the more complex output function achieved quite the same level of performance.

However, the results were significantly better than might be obtained with random guessing. When presented with one true and one false example, the potential value assigned a higher value to the true example more than 50% of the time. Further, the potential function seemed to be able to select the contacting residues (i, j) over the pairs with one register shift such as $(i, j-1)$ [12].

Finally, comparing the polarity experiment with the others suggests that most of the information the network was learning was patterns in polarity. This suggests that the predictor had learned about the bipolar nature of β-sheets, but might not have learned more past that.

8.5 Summary

We succeeded in harnessing some β-strand contact information, as the potential functions performed substantially better than random guessing. However, the information gain was not overwhelming. In retrospect, this is consistent with the results shown in Chapter 7: there is genuine signal in pairwise potentials, but the amount of information is limited.

Chapter 9

Prediction of alignment position reliability

Accurate alignment of a target protein sequence to a family of homologs is a fundamental problem in computational molecular biology. *Fundamental* might seem like a strong word, but its use here is appropriate. The Webster dictionary definition of "fundamental" includes the following: "serving as a basis supporting existence or determining essential structure or function". If the family of homologs includes a template sequence of known structure, the alignment of the target and template sequence yields a prediction of the structure of the target protein and serves as a starting point for homology modeling. Multiple alignments are used to predict secondary structure [152], functional residues [134], evolutionary relationships [167], to identify new homologs [144], and to locate genes in new genomes [102]. Unfortunately, alignment methods sometimes make mistakes, as seen in Chapter 6. Any mistakes made in the alignment will be reflected in its application. A system that would be of tremendous practical value in molecular biology is one that studied an alignment in which some new sequence had been added to a family, and produce a meaningful measure of the reliability of each section of the alignment, preferably without requiring access to the alignment program. No good system exists today.

There are at least three good reasons to investigate predicting reliability of alignment columns. First, it offers the option to post-process and hopefully improve predicted alignments; unreliable regions can be removed, and core regions predicted as unreliable can be realigned. Second, estimates of column reliability can be incorporated into systems that build on predicted alignments, such as tools for homology modeling or predicting functional residues. Such tools currently weight all alignment positions equally. Third, study of factors that suggest an unreliable alignment position permit us to assess where alignment systems make mistakes, in hopes of improving the alignment systems.

We have explored one complex and three simple systems for predicting unreliable regions in alignments of remotely-related target and template sequences, aligned by hidden Markov models. The simple systems are single physical or statistical quantities considered to reflect alignment reliability.

1. The log odds ratio given the amino acid type of the residue aligned. The log odds ratio relates the conditioned probability of the residue given its amino acid type aa and given the alignment column to the unconditioned probability of that amino acid type: $\log\left(\frac{P(aa|column)}{P(aa)}\right)$. The log likelihood ratio reflects how much more likely the residue is to appear in the column given the other residues already there.

2. A similar log likelihood ratio derived from the HMM forward-backward table, reflecting the posterior probability of the residue to align to the column given all paths through the model. This posterior decoding column cost is related to the posterior-decoded alignments we saw in Chapter 5. In Chapter 5, we saw that when we compute alignments according to posterior decoding information, examining all paths through the model rather than the single best-scoring path, alignment quality is substantially improved over the standard method. In this chapter, we examine the impact of using this posterior decoding information to validate

alignments, whether or not they were built with posterior decoding.

3. The distance in the template sequence to the nearest helix or strand residue. This measure follows the intuition that long loop regions are less conserved in evolution, and are more difficult to model and align.

The complex system is a neural network trained to predict the reliability of an alignment column, containing the above inputs and others. We have tested the four systems by using them to predict alignment column reliability, removing those columns predicted as unreliable, and measuring the change in alignment scores. On hard remote homologs, our systems removed 73.5% of the unreliable positions, while preserving 81.8% of the accurate ones. This yielded a 15% improvement in overall alignment quality over alignments estimated by our best methods.

9.0.1 Related work

Though few such systems have even been proposed, there are brave souls who have addressed this problem.

Jong Park [146] developed a tool that predicts reliable regions in a pairwise alignment based on the sequence similarity in regions of the alignment. This system was never demonstrated to be effective; with so little information at its disposal, it is not clear that it would be. According to its author, it is no longer under development [142].

Kimmen Sjölander developed a system that predicts reliable regions in the alignment of a template sequence on the basis of a log likelihood ratio. For each alignment column i where some residue of amino acid type aa_j from the target sequence is aligned to the column, the system estimates $-\log\left(\frac{P(aa_j|column_i)}{P(aa_j)}\right)$ where $P(aa_j|column_i)$ is the conditioned probability of aligning amino acid aa_j to column i given the alignment data in the column, and $P(aa_j)$ is the unconditioned probability of amino acid aa_j. A threshold on this log likelihood ratio was then used to identify and remove the less reliable positions. Though this system has been described in conference presentations, it is not published in any conference proceedings or elsewhere in the literature [165].

Two authors have observed that seed alignments contain regions of greater conservation and regions of greater variability, and that alignment to the variable regions should be regarded with greater suspicion. Dopazo [35] suggested a method to quantify variability within a sequence alignment based on the PAM distance between each of the aligned sequences, and demonstrated it on one alignment. Again, this is hardly a convincing demonstration. He suggested that his measure might be useful for gauging alignment reliability, but did not apply it. This work is not heavily cited in the literature, and it does not appear that any other author has applied his measure to alignment reliability. Laurio [106] observed that when an HMM estimated from an alignment is used for detecting remote homologs, the high variability alignment regions do not seem to be informative; the states at corresponding model positions can be replaced with states with default characteristics without any adverse impact on fold recognition accuracy. This suggests that the important information in the high variability alignment regions is not their contents, but their length. Note that the focus of Laurio's work was on fold recognition accuracy rather than alignment reliability. However, the work depends on alignment reliability. Within Laurio's framework, the structure of a target sequence is predicted on the basis of the score of its alignment to HMMs of various structural families.

Vingron and Argos [189] proposed a system that estimates the reliability of each position in an alignment according to the score of the best alignment that excludes that position. Such second-best alignments are referred to as *near-optimal alignments*. The motivation to study them is the intuition that when a sequence is aligned, there may be regions of the sequence for which many distinct alignments are likely, and alignment of such regions should be viewed with more skepticism than those for which there is one optimal alignment. Vingron and Argos showed their reliability index to be effective on five alignments with pairwise identities ranging from 39% to 16%, built with free end gaps and affine internal gap costs. While five alignments is not enough for a convincing demonstration, their results sparked further interest. Zuker [200] extended it to local alignment. Naor and Brutlag [129] studied the statistics of near-optimal alignments, and observed that the regions included in the greatest number of alignments tend to correspond to the conserved core regions of the protein. Yet central to all of these works is the assumption that a high reliability index implies an accurate position, rather than a high-scoring but misleading position. Mevissen and Vingron [120] tested this assumption on 109 alignments: 19 with sequence identity in the range 25-30%, 46 in the range 30-40%, and 44 in the range 40-50%. They observed that when the overall sequence identity is 30% or greater, positions with a high reliability index are misaligned very rarely. In the 25-30% identity range, a strong score did not preclude misalignment, though errors were more frequent at the positions with lower reliability indices. They did not address alignments with less than 25% identity. At this time, 25% identity is regarded as near the top end of the twilight zone [154], and homology within this range would constitute a comparative modeling test rather than a fold recognition test [87]. Zhang et. al [196] took a similar approach to post-processing alignments. Rather than omitting single positions found to be important to the alignment according to the score, their algorithm assembles a final subalignment from regions that are bordered by a substantial drop in alignment score. They presented a rigorous mathematical basis for their system, but the system itself was only demonstrated on a few examples.

A number of groups have addressed near-optimal alignments and alignment reliability by estimating *forward-backward probabilities* for each residue and position, the probability with which the residue is aligned to the position, given all possible alignments [70, 123, 198, 183]. These forward-backward probabilities, described earlier in this chapter, are then used with the Viterbi algorithm, or some close relative, to build a most-probable alignment. The alignment is then truncated according to some threshold probability. This technique is referred to as *posterior decoding*. Though it is more time-consuming than standard Viterbi alignment, it has been shown to yield more accurate alignments [70]. As detailed in Chapter 5, our experience is that it aligns more positions accurately than Viterbi, but suffers somewhat from over-alignment, perhaps suggesting that we have not found the optimal threshold value.

Most recently, Yu and Smith [194] suggested that these forward-backward probabilities can be useful to determine what positions in an alignment are most reliable. For each amino acid aa_j from the target sequence, aligned to some column i in the template alignment, they compute the log likelihood ratio $-\log\left(\frac{P_{fb}(aa_j|column_i)}{P(aa_j)}\right)$, where $P(aa_j)$ is the prior probability of amino acid aa_j and $P_{fb}(aa_j|column_i)$ is the posterior probability of aligning amino acid aa_j to alignment column i, given the posteriors calculated in the forward-backward table. They provided a mathematical justification on why this quantity should be useful, and demonstrated it on two test cases.

9.1 Methods

The goal of this work was to study alignments of hard, remote homologs, as aligned by SAM, and to identify the reliable regions. The ultimate goal of this line of work is a general-purpose predictor to identify the reliable regions of alignments built with any method. However, any investigation toward a general method should start with a more specific method as a proof of concept. We do hope that some of the lessons learned in this work apply to alignments produced by other methods. However, that point has not been tested at this time.

9.1.1 Description of the dataset

The focus of this work was on hard remote homologs. As described in Chapter 6, the literature states consistently that many methods can align two close or moderate homologs accurately, but as homology becomes more remote, accurate alignment becomes a greater challenge. For this work, there are two good reasons to focus on remote homologs. First, “the twilight zone”, now described as below approximately 20% identity, is where contemporary alignment methods begin to fail and where the next generation of improvements needs to be made. Second, if one is setting out to identify accurate regions, there must first be inaccurate regions.

The dataset used for these experiments was described in Section 6.1.1. Briefly stated, we identified 200 pairs of structures with high structural similarity and low sequence similarity. High structural similarity was determined by significant structural superposition by three structural aligners: DALI [68], VAST [51], and the Yale aligner [50]. Low sequence similarity was determined by removing any pairs for which the FASTA pairwise aligner [148] could generate an accurate alignment, where the criterion for accuracy was a shift score of 0.4 or better with the closest of the three structural alignments.

As described in Section 6.1.1, we divided this set of 200 pairs into an optimization set of 130 pairs and a validation set of 70 pairs. The optimization pairs were used to develop this method, and the test pairs were reserved for final system validation. As another validation step, we applied the completed system to the 170 pairs of structures used to optimize SAM alignments as described in Chapter 5. The purpose of this step was to compare the trimmed alignments in quality to the alignments built with the methods described in Chapter 5.

9.1.2 Description of the alignments

Both global and local alignments produced by SAM tend to contain suspect regions, as described in Chapter 5. Both types of alignments could benefit from trimming. However, we have focused on global alignment for two reasons. First, global alignments typically have better optimal subalignment scores than local alignments, indicating that a greater number of positions are aligned accurately with global alignment. Second, global alignments tend to contain more inaccurate positions than local alignments; they need more trimming. Thus, global alignments represent both greater need and greater opportunity for an alignment trimming system.

I have experimented with alignments built by two different methods, using SAM. These represent two of the best methods available for aligning remote homologs with SAM. The methods used were as described below. All alignments were built with the `w0.5` weighted build option.

1. Global alignments built with FSSP seed alignments and character-based posterior decoding (the `-adptyle 5` alignment option). As shown in Chapter 5, this is currently the best method

available to generate alignments with SAM. A system which shows improvement on these tests will have the distinction of improving our best methods.

2. Global alignments build with SAM-T99 seed alignments and the default Viterbi algorithm. When SAM-T99 alignments are built, the methodology used is local Viterbi alignment with the `w0.5` weighted build. That methodology is “the workhorse”, the way in which SAM is used most often. This test has three merits. First, it represents more of a general test than the previous one. Second, it provides the opportunity to study when “normal” SAM alignments are in error, in hopes of improving the methods. Third, it can be applied even when the structure of the template sequence is unknown.

While the ultimate goal in this line of research is a general-purpose prediction method, we did not build one general-purpose predictor at this time. Rather, we built two sets of specialized predictors for the two tests described above. In experimental design, there is often a tradeoff between generality and performance. We recognize that a general-purpose predictor might never match the performance of the predictors described in this chapter. However, building specialized predictors for the two cases permitted us to better establish the proof of concept of this line of research.

9.1.3 Training, cross-training, and validation alignments

The dataset described in Section 9.1.1 contains 200 pairs of protein structures with strong structural similarity and weak sequence similarity. This set of 200 pairs was divided into an optimization set of 130 pairs and a validation set of 70 pairs.

Note that each pair represents two alignment tests: aligning the first sequence to the second and its homologs (*Direction A*), and aligning the second sequence to the first and its homologs (*Direction B*). Thus, there were 260 alignments in the optimization set.

We trained our neural networks on the 130 Direction A alignments. These 130 alignments contained approximately 20,000 target residue alignment positions, or 20,000 data points. These 20,000 data points were divided at random into training and cross-training sets, with approximately two-thirds of the data going to the training set. The training set was used to adjust neural network weights. Overall performance on the cross-training set was measured after each training epoch, and was used to select the best weight set and update the adaptive learning rate.

Once the neural network was trained, it was used to predict the reliability of the columns of the 130 Direction B alignments. We then trimmed these alignments according to the predicted reliability of each position, removing positions where some target sequence residue was aligned with predicted reliability below some threshold value. We experimented with a number of threshold values, and selected a final threshold value empirically. Scoring involved using the shift score to compare each predicted alignment against three structural alignments of the same sequences, where the structural alignments were produced by DALI, VAST, and the Yale aligner.

Next, we used the neural network and reliability threshold defined according to the optimization alignments to trim the validation alignments. We measured trimming performance by scoring the initial alignments against structural alignments, removing alignment positions predicted as unreliable, and scoring the trimmed alignments against the structural alignments. To score the alignments, we compared each predicted alignment to structural alignments of the same sequences as produced by the three structural aligners. For each comparison, we measured three quantities: the shift score; *alignment specificity*, the number of residue pairs aligned correctly as a fraction of the number of residue pairs aligned; and *alignment sensitivity*, the number of residue pairs aligned correctly as a fraction of the number of correct pairs.

As a final step, to better compare the trimmed alignments to the alignment-building methodologies described in Chapter 5, we applied our trimming systems to those alignments. Here, we used the shift score to compare the trimmed alignments to structural alignments produced by DALI, and compared the trimmed alignments to others on the basis of the average shift score.

The single-valued predictors were developed using much the same system. As their development merely required estimating a threshold value, we estimated this value empirically on the 130 Direction B alignments in the optimization set. Validation of the single-valued predictors followed exactly the same procedure as that of the neural networks.

9.1.4 Neural network architecture and training

This section describes the methodology we used to train neural networks for estimating the reliability of target sequence alignment positions. We described neural networks in Section 7.2.1. For the work in this chapter, we trained neural networks to look at data representing a window of alignment columns, centered on the column in question, and estimate the reliability of the alignment of the target residue to the column. In the sections that follow, we describe the target reliability value, the inputs to the neural network, the network architecture, and steps taken to minimize complexity.

Target value for reliability prediction

For each predicted alignment, each position containing a target residue was assigned a reliability score. This score was obtained by comparing the predicted alignment to the FSSP structural alignment, and recording the shift column score. This measure is an intermediate value in calculation of the shift score, as described in Chapter 4. For a column aligning two residues a and b, the column score is computed as follows.

$$\begin{aligned}
\text{subscore(residue)} &= \frac{1+\epsilon}{1+|\text{shift(residue)}|} \text{ if shift(residue) defined} \\
&\quad 0 \text{ otherwise} \\
\text{column score} &= \text{subscore}(a) + \text{subscore}(b)
\end{aligned}$$

The column score ranges from -0.4 to 2.0, with higher values indicating greater reliability.

The neural network is trained to estimate the probability that the column was *reliable*, where a reliable column is defined as one which is included in an optimal subalignment. In preliminary work, I obtained a target probability from the column scores by estimating the probability with which the column would be included in an optimal subalignment, given its column score. According to preliminary analysis, this computation was as follows:

- When the column score is 0 or less, the column is never included. If a column has such a score, then either it's the result of over-alignment or both residues in the column are shifted by five or more positions.

- When the column score is 0.8 or higher, the column is always included. Columns with scores in this range are either correctly aligned, or both residues are shifted by one position. Note that among biochemists, a shift of 1 is often not regarded as serious, and can be construed "stylistic" result of converting a high resolution atomic prediction of structural superposition to a lower-resolution alignment [28].

- In other cases, the probability of a reliable column decreases rapidly as the column score decreases. As shown in Figure 9.1, the square of the column score is a noisy but reasonable approximation.

Alignment column information

The input to the neural net was a vector of information on a window of alignment columns, centered on the column in question. Based on preliminary experiments, I started with a window size of 15 alignment columns, centered on the column in question. The window size was optimized in later steps.

Information was collected on each column of the predicted alignment, as described in the following list. The data was derived, as indicated, from the input alignment or from the *seed alignment*. The seed alignment is the alignment of the template sequence and its homologs to which

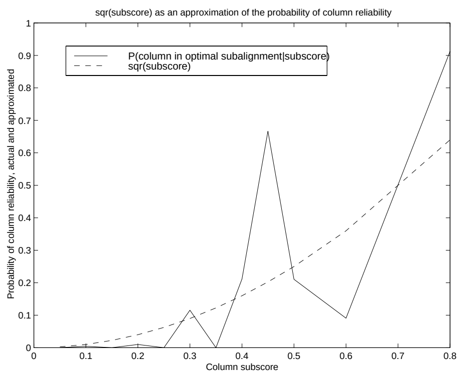


Figure 9.1: Depicted is the probability that the column is included in an optimal subalignment as a function of the column score. While this probability is not a smooth function, the square of the column score is a reasonable approximation. Data for this figure was obtained from 170 alignments built by SAM with an early version of the `w0.5` weighted build.

the target sequence had been aligned. In practice, we obtained it by removing the target sequence from the input alignment.

Forward-backward information, included in the input set, is complex enough to merit a few separate paragraphs. Forward-backward information reflects the likelihood with which each target sequence residue aligns to each column, given all possible alignments of target sequence and seed alignment. This information was obtained by estimating a model from the seed alignment with the `w0.5` weighted build and using the `grabdp` program to generate a posterior-decoding (`*.pdoc`) file for the target sequence. This file dumps out costs of aligning each residue to each position, where the costs are estimated as a negative log of the posterior. Three sets of costs are output: *transition*, *match*, and *insert costs.* Chapter 5 discussed posterior decoding to estimate an alignment according to the match costs or transition costs. Posterior decoding with the `-adp5` switch estimates alignments according to the match and insert costs, while posterior decoding with the `-adp4` switch estimates alignments according to the transition costs.

Match costs, the most intuitive, reflect the cost of emitting the amino acid at each node in the model, or aligning the residue at each column in the alignment. Insert costs reflect the cost of inserting the residue before each column in the alignment. Together, match and insert costs reflect the alignment column where the target residue is most likely to appear.

Transition costs reflect the cost of aligning the residue to each column in the alignment given the model's transition parameters, or gap parameters. The difference between transition costs and match and insert costs is that transition costs reflect transition probabilities directly and column probabilities indirectly, while the opposite is true for the others.

Not all values described below were available in all cases. When input values were not available, we set the inputs to default values. The default values were selected in preliminary experiments to approximate the midpoint of the experimental range.

The following list describes main quantities that we experimented with as input to the neural networks. As described in Section 9.1.4, we started with a large set of inputs and pruned down to those most relevant. Note that this list is not comprehensive. Out of mercy to my thesis committee, I have not described every quantity that I experimented with, only those that proved most significant.

Indel locations: Within the window, we noted all positions where there was a gap in the target sequence or an insert in the alignment.

Column Log Odds: This quantity reflects the likelihood that the target residue would align to the column, given the amino acids already aligned to the column. The column log odds is the log likelihood ratio $\log\left(\frac{P(aa_t|column)}{P(aa_t)}\right)$, where where aa_t is the target sequence amino acid aligned to the column, $P(aa_t|column)$ is the conditioned probability of amino acid aa_t in the column, and $P(aa_t)$ is its unconditioned probability. The probabilities were extracted from a model of the seed alignment built with a `w0.5` weighted build.

Savings: The `estimate-dist` program was used with the `w0.5` weighted build to measure the *savings* at each column of the seed alignment. Savings is defined as the difference between the entropy of the amino acid prior probability distribution and the entropy of the amino acid prior probability distribution.

Non-loop distance: When the template residue at the column in question is in a strand or helix, the non-loop distance is zero. When the template residue is in a loop, the non-loop distance is the number of residues to the nearest end of the loop. Long loop regions tend to be less conserved than strands or helices, so positions aligning long loops should be regarded with more skepticism. This quantity was used both as a stand-alone predictor and as an input to the neural networks. Its value was held constant over gaps, and at positions for which the template secondary structure was not defined because its structure was not fully resolved. When the non-loop distance was used as input to the neural network, the value used was $\frac{1}{1+D}$. When it was used as a stand-alone predictor, its raw value was used.

Predicted Secondary Structure Similarity: Intuition states that an alignment is more trustworthy in positions where the target sequence is predicted to have the same secondary structure as the template sequence. This is especially true if the target sequence is predicted with high probability to have the same secondary structure as the template sequence. For each target sequence, we obtained a prediction of its secondary structure by applying the `predict-2nd` program to a `SAM-T99` alignment of the target sequence and its sequence homologs. `Predict-2nd` is a neural network program, and the trained neural network used was `t99-2877-IDaa13-5-8-7-10-5-9-11-ehl-seeded4-trained.net`. For each position where some target residue is aligned to the template sequence, we obtained from the `predict-2nd` output $-\log\left(P(SS(\text{template})|\text{target alignment})\right)$, the probability of the template secondary structure, given the alignment of the target sequence and its homologs.

Posterior Decoding Information: Posterior decoding information, as described above, reflects the likelihood of each target residue aligning to each seed alignment column given all alignments of target sequence to seed alignment. The information was obtained by generating a model of the seed alignment with a `w0.5` weighted build, and using the program `grabdp` to estimate and print out forward-backward information on the alignment of the target sequence. Below are descriptions of the precise posterior decoding quantities used.

Posterior-decoded Transition Cost: This quantity reflects the likelihood that the target residue would be aligned at the seed alignment column. It is derived from the probability of the targetresiduealigning to the seed column given the model's transition parameters, or gap parameters. This quantity reflects gap likelihoods directly and column probabilities indirectly.

Posterior-decoded Column Cost: This quantity also reflects the likelihood that the target residue would be aligned at the seed alignment column. In contrast to the previous quantity, it reflects column costs directly and transition costs indirectly. It is computed as $-\log(P(r_i|col_j))$, where r_i is the target sequence residue, col_j is the column where r_i is aligned, and $P(r_i|col_j)$ is the probability of residue r_i aligning to column col_j according to the column's amino acid probability distribution.

Posterior-decoded Insertion Cost: This quantity reflects the likelihood that the target residue would be inserted before the seed alignment column rather than aligned to it. The column cost and insertion cost together reflect where the target residue is most likely to appear. The insertion cost is computed as $-\log(P_{ins}(r_i|col_j))$, where r_i is the target sequence residue, col_j is the column where r_i is aligned, and $P_{ins}(r_i|col_j)$ is the probability of residue r_i inserted before column col_j.

Other Posterior Decoding Data: There is a chance that the target sequence residue is not aligned to the column where its cost is minimized. Therefore, for both transition-based and character-based posterior decoding, we investigated information comparing the position where the residue appears to the position where it would appear at lowest cost, whether aligned or inserted. We investigated this difference in cost, and the distance in the alignment between the column where the target residue appears and its lowest-cost position. For neural network input, the distance was encoded as $\frac{1}{1+D}$.

Neural network architecture and training

The neural networks used in this experiment are feed-forward back-propagation networks. The neural network training is performed with the `Aquanet` program, based on software originally written by Albion Baucom.

Each training example is described as a combination $(r_i, \vec{c}, P(r_i))$, where target sequence residue r_i is aligned to a column described by the input vector $\vec{c}$. $P(r_i)$ is the actual probability that residue r_i is aligned accurately, defined on the basis of its column score as described in Section 9.1.4. The neural network estimates $\hat{P}(r_i|\vec{c})$, the estimated probability that residue r_i is aligned accurately given data on surrounding columns $\vec{c}$.

The neural network was trained with an entropic learning rule. For each example $(r_i, \vec{c})$ with target output $P(r_i))$, the error function is defined as

$$-P(r_i)\log\left(\hat{P}(r_i|\vec{c})\right) - (1 - P(r_i))\log\left(1 - \hat{P}(r_i|\vec{c})\right).$$

The networks used the standard feed-forward gradient descent algorithm, with the log-sigmoid activation function [63]. and the learning rule was obtained by calculus derivation of the error function. We experimented with single-layer and two-layer neural networks, selecting the final architecture empirically.

When we experimented with neural networks with hidden layers, no windowing was applied to the hidden layer.

As described in Section 9.1.3, the data for building the neural networks was divided randomly into thirds, with two thirds assigned as a training set and one third assigned as a cross-training set. We performed on-line training, updating the weights after every training example in the training set. After each training epoch, we kept the weights fixed and measured overall performance on the cross-training set. The training algorithm used an adaptive learning rate, which was modified according to cross-training performance. Further, the final weight set was selected according to cross-training performance.

Reducing complexity

Neural networks perform best when presented with a small number of relevant inputs. Additional inputs often do not help, because they represent additional statistical parameters that the neural network must estimate with the same amount of data. Therefore, when working with neural networks, one should try to simplify the input set whenever possible.

We used *sensitivity analysis* [97] to identify and remove extraneous inputs. This technique measures the change in network error relative to the change in each input. When measured over the training set, sensitivity analysis reflects which inputs the network emphasizes least, and those inputs can be removed. Direct examination of the weights can yield similar information for a perceptron, but is not viable for a multi-layer network. Techniques such as weight decay [63] yield a similar effect, but in our experience, sensitivity analysis works better for removing extraneous inputs.

Our analysis performed numerous iterations of sensitivity analysis. In each iteration, we would train a neural network, estimate the sensitivity of the network error to each input, and study the input sensitivities. When we studied the input sensitivities, we would watch for two things: which positions in the window of alignment columns are not important, and which inputs are not important in any column. When an input appeared to not be important in any column, we would remove that input for all columns and train a new neural network. If the new network achieved comparable performance, as estimated by its best cross-training performance, we would continue with another input. If the network performance suffered after removal of some input, we would reinstate the input and try removal of a different one. Even if the network appears to be insensitive to a certain input, its value must still be tested empirically because it might be providing indirect benefit to some other input. A frequent occurrence in sensitivity analysis is that after removal of a seemingly-unimportant input, some previously important input appears far less useful. Neural networks tend to compare inputs, and particularly tend to emphasize their differences. When one removes an input that does not appear important and some previously-important input loses its value, this suggests that the important input is valuable in comparison to the less-important one.

Note that there are two schools of thought in feature selection: *top-down* and *bottom-up*. Top-down feature selection involves starting with a large set of inputs and gradually removing those which appear unimportant. Bottom-up feature selection involves starting with features proven to be valuable independently and gradually adding other features if they help. In this instance, we have followed the top-down approach in hopes of not missing any indirect relation between features.

Using sensitivity analysis, we pruned the feature set until we arrived at a set of features that could not be further reduced without compromising network performance. After optimizing the input set, we optimized the window size empirically. After optimizing the window size, we optimized the network architecture empirically. The reader can rest assured that before we arrived at our final networks, we tested that all complexity in the final system served a purpose.

9.2 Results

We had two goals for this work. The first was to improve alignments produced by the best method we have available, currently global alignment with FSSP seed alignments and character-based posterior decoding. The second goal was to improve alignments built with more commonly-used methods: global alignment with Target99 seed alignments, a method that has the major advantage that it requires no structural information. While improvement over the best method can be exciting, improvement over a commonly-used method is perhaps more valuable. Further, if improvement can be shown over two different methods, there is less chance that the performance gain is particular to the method.

For each of these two cases, we investigated one complex and three simple predictors of the accuracy of the alignment of a target sequence residue to a seed alignment column. The simple predictors were column log odds, non-loop distance, and posterior-decoded column cost. Each of these predictors is described in Section 9.1.4. The complex predictor was a neural network that used these simple predictors plus other data described in Section 9.1.4 to predict column reliability. For each of the two classes of alignments, we empirically estimated a trimming threshold for each predictor, and then applied the predictor and threshold value to a library of validation alignments.

While the ultimate goal of this line of work is a general-purpose alignment reliability estimation system, the immediate goal was to show that the methodology can work. Therefore, we built separate predictors for the FSSP and Target99 alignments, and analyzed them separately. At this time, we have not addressed building one predictor to analyze any alignment no matter what its source. However, when we analyze what information worked well for each case, their similarity bodes well for a more general method.

9.2.1 Neural network optimization

As described in Section 9.1.4, we optimized the neural networks in the following order. First, we optimized the input feature set, using sensitivity analysis to indicate inputs that were not important during training. All features used as input to the neural networks are described in Section 9.1.4.

After optimizing the feature set, we optimized the size of the window over which we collected input features, where a window size of one would collect data merely on the column in question and a window size of three would collect data on the two adjacent columns as well. Finally, we optimized the network architecture.

For the FSSP neural network, only five features proved to be worth including in the neural network. In order of importance, they are the posterior-decoded column cost, the column savings, locations of gaps within the window, the predicted secondary structure similarity, and the posterior-decoded insertion cost. Each of these features is described in Section 9.1.4.

For the neural network to evaluate Target99 alignments, seven features proved worthwhile.

In order of importance, they are the posterior-decoded column cost, the column savings, distance in alignment columns from where the residue is aligned to where its transition-based posterior decoding cost is minimized, distance in alignment columns to where the character-based posterior decoding cost is minimized, the predicted secondary structure similarity, location of gaps within the window, and the posterior-decoded insertion cost.

Features which did not prove worthwhile in either case include the column log odds, the non-loop distance, the transition-based posterior decoding cost, the difference in posterior decoding costs between the position aligned and the lowest-cost position. Note that this does not mean these features are of no value. More likely, it indicates that other quantities conveyed similar information but were slightly more informative.

After network optimization, the final architecture for the global posterior-decoded FSSP alignments used a window of thirteen alignment columns centered at the position in question, for a total of 65 inputs. It had a hidden layer with one hidden unit. When a neural network has a hidden layer with one single hidden unit, the effect of the hidden layer is to permit the network to represent a wider variety of higher-order functions, due to the presence of the additional activation function. For Target99-based alignments, the optimized neural network used a window of fifteen alignment columns centered at the position in question, for a total of 105 inputs. It had a hidden layer with ten hidden units. In both cases, windowing was applied to the outer layer of network inputs only; the hidden layer involved no windowing.

9.2.2 Selecting prediction thresholds

For all of the predictors, including the neural network, we selected a trimming threshold empirically, as follows. First, for each alignment to be trimmed, we measured the predictive quantity for each column where a target residue was aligned. Next, for each predictive quantity, we experimented with a range of threshold values. We defined a trimming range as from some minimal value to the threshold for cases where a small quantity suggests a less-reliable position, or

Predictor	Shift Score	Range for Removal
None	0.402	Nothing removed
Non-loop Distance	0.402	Greater than 5
Column Log Odds	0.405	Less than -2.7
Posterior Decoding Column Cost	0.417	Greater than 1.9
Neural Network	0.418	Less than 0.20

Table 9.1: On FSSP alignments, we realized minor improvements by trimming out positions aligned to the middle of long loops or positions scoring as very unlikely given the column log odds. More substantial improvements were realized by trimming according to the posterior decoding column cost or the neural network prediction. For each predictor, this table describes the trimming ranges that yielded best results for the threshold selection alignments. For comparison, the average optimal subalignment score for the untrimmed alignments was 0.533. The shift score for local posterior-decoded alignments was 0.366, and the shift score for local Viterbi alignments was 0.269..

from the threshold to some maximal value for cases where a large quantity suggests a less-reliable position. Then, for each alignment to be trimmed, we removed any target sequence positions for which the predictive value fell within the trimming range. To remove a position, we modified the alignment so that where a target sequence residue was previously aligned to a column, the target residue was instead inserted before the column and the column remained empty. Finally, we scored the trimmed alignments against FSSP structural alignments, and measured the mean shift score for the trimmed alignments. For each predictor, we chose the threshold according to the value that yielded the best shift score.

As described in Section 9.1.3, we trained the neural networks on one set of alignments and estimated the thresholds with a different set of alignments. The data used to estimate the thresholds had never been seen by the neural networks until they were fully optimized.

Table 9.1 shows the threshold values that yielded the best results for the posterior-decoded FSSP alignments. All predictors had some threshold value that yielded an improvement to the alignments. For the non-loop distance, we found our best results by removing the middle of long loops; removing alignment positions where the number of template sequence residues to the nearest non-loop residue was more than five. Performance improvements were realized by removing positions with a column log odds of less than -2.7 or a posterior decoding column cost of greater than 1.9. Neither of these represents a very stringent threshold; it indicates removing only those positions

Predictor	Shift Score	Range for Removal
None	0.259	Nothing removed
Non-loop Distance	0.262	Greater than 3
Column Log Odds	–	No effective range found
Posterior Decoding Column Cost	0.287	Greater than 0.7
Neural Network	0.294	Less than 0.15

Table 9.2: On Target99 alignments, we realized improvement by trimming according to the distance to the nearest non-loop residue, the posterior decoding column cost, or according to the neural network prediction. No improvement was realized by trimming according to the column log odds. For each predictor, this table describes the trimming ranges that yielded best results on the threshold selection alignments. For comparison, the average optimal subalignment score for the untrimmed alignments was 0.408; the shift score for Viterbi local alignment was 0.251 and the shift score for posterior-decoded local alignment was 0.270

of rather low confidence. The neural network predictions yielded the best performance gain when we removed positions with less than a 20% prediction of reliability. Again, this corresponds to removing those positions where some target residue is aligned, but the alignment signal is weak.

Table 9.2 shows the thresholds found best for the Target99 alignments. Compared to the global posterior-decoded FSSP alignments, these alignments are of lower quality. Trimming by column log odds was not effective on these alignments. So many low-probability positions were aligned that any threshold value cut away too much of the alignment: while the remaining positions tended to be accurate, the alignments were too short to yield good shift scores.

For both the non-loop distance and the posterior decoding column cost, the best threshold found was lower than that for the global posterior-decoded FSSP alignments. Compared to the FSSP-based alignments, they achieved their best performance in this case with lower thresholds. More positions were inaccurate, and the lower thresholds correspond to more trimming.

In contrast, the best threshold for the neural network was slightly lower than it had been for the global posterior-decoded FSSP alignments. This does not indicate that fewer positions should be trimmed: 54% of all positions in the global Viterbi Target99 alignments have a neural net prediction of less than 0.15 while 53% of the positions in the FSSP-based alignments have a prediction of less than 0.2. Rather, it reflects that a greater portion of training data was unreliable alignment positions, leaving the network with a greater bias toward predicting unreliability.

	Shift Score			
Method	FSSP	VAST	Yale	Closest
Original Alignments	0.369	0.365	0.334	0.404
Local Posterior-decoded Alignments	0.311	0.309	0.289	0.340
Trimming by Non-loop Threshold	0.373	0.368	0.337	0.408
Trimming by Column Log Odds	0.371	0.367	0.336	0.407
Trimming by Posterior Decoding Column Cost	0.377	0.371	0.340	0.412
Trimming by Neural Network	**0.382**	**0.372**	**0.344**	**0.414**
Optimal	0.512	0.464	0.437	0.520

Table 9.3: Results on trimming the 140 validation alignments according to the various predictors of alignment reliability: posterior-decoded FSSP global alignments. The shift scores for posterior-decoded FSSP local alignments are shown for comparison. The best results in each category are shown in boldface..

9.2.3 Validation results with three structural aligners

Using the thresholds derived in Section 9.2.2, we applied the predictors to a library of 130 validation alignments. As described in Section 9.1.3, these alignments was not seen during neural network training or threshold selection. For global Viterbi Target99 alignments, we had not found an effective threshold for the column log odds predictor, and therefore did not apply it to the validation alignments.

For each alignment in the validation set, we measured the remaining predictive quantities for all positions where some target sequence residue is aligned. We then removed positions from the alignment according to the threshold value for each predictor, and scored each alignment according to the structural alignments. Tables 9.3 and 9.4 show the average shift score comparing each set of trimmed alignments to structural alignments from FSSP, VAST, the Yale aligner, and to whichever of the three the trimmed alignment was closest. The results for global posterior-decoded FSSP alignments are shown in Table 9.3, and Table 9.4 shows the results for global Viterbi Target99 alignments.

In Table 9.3, we see that all methods tested yielded an improvement over the original alignments. The best performance is achieved by the neural network and by posterior decoding column cost. These methods yield an improvement of 2 to 3.5%. While this might sound like a modest gain, recall the method used to build these alignments, posterior-decoded global alignments

	Shift Score			
Trimming Method	FSSP	VAST	Yale	Closest
Original Alignments	0.193	0.207	0.206	0.238
Local Viterbi Alignments	0.178	0.188	0.189	0.208
Local Posterior-decoded Alignments	0.214	0.216	0.223	0.251
Trimming by Non-loop Threshold	0.193	0.206	0.206	0.237
Trimming by Posterior Decoding Column Cost	0.222	0.225	0.230	0.257
Trimming by Neural Network	**0.233**	**0.235**	**0.236**	**0.267**
Optimal	0.352	0.330	0.330	0.368

Table 9.4: Results on trimming the 140 validation alignments according to the various predictors of alignment reliability: Target99 global alignments. The shift scores for local Viterbi and local posterior-decoded alignments are shown for comparison. The best results in each category are shown in boldface.

with FSSP seed alignments, is one of our best methods. Therefore, any improvement over the original alignments is noteworthy.

In Table 9.4, we see that the non-loop threshold method was not effective at the global Viterbi Target99 alignments. However, the other two methods were, and yielded an improvement of approximately 15% where an optimal method might yield an improvement of about 45%. As this improvement is shown relative to three structural alignments, there is little risk that the improvement is coupled somehow to one of the structural aligners.

To further explore which positions are removed and which are retained by trimming, we further analyzed the validation alignments relative to the FSSP structural alignments. We focused on three categories of alignment positions: aligned accurately, over-aligned, and misaligned by five or more positions. We counted the number of positions in each category before and after trimming. This data is shown in Table 9.5.

Two points are clear from Table 9.5. First, both methods are doing something right. For the Target99 global alignments, they remove approximately 60% of the positions that are significantly misaligned and 85% of the over-aligned columns while retaining as much as 89% of the accurately-aligned columns. Similar results are seen for the global posterior-decoded FSSP alignments. Second, the neural network does not seem to yield much performance gain over posterior decoding column cost. Posterior decoding column cost had been the dominant source of information

Positions Remaining After Trimming: Global Posterior Decoding FSSP Alignments				
Trimming	Total Positions	Aligned Accurately	Over-aligned	Greatly Misaligned
None	24802	8987	8972	5661
Posterior Decoding Column Cost	19514	8270	4277	3745
Neural Network	16164	7700	3064	2725

Percentages Remaining After Trimming: Global Posterior Decoding FSSP Alignments				
Trimming	Total Positions	Aligned Accurately	Over-aligned	Greatly Misaligned
Posterior Decoding Column Cost	78.7%	92.0%	47.7%	66.2%
Neural Network	65.2%	85.7%	34.2%	48.1%

Positions Remaining After Trimming: Global Viterbi Target99 Alignments				
Trimming	Total Positions	Aligned Accurately	Over-aligned	Greatly Misaligned
None	27101	6121	8522	9701
Posterior Decoding Column Cost	13317	5264	3566	2941
Neural Network	13955	5460	3418	3400

Percentages Remaining After Trimming: Global Viterbi Target99 Alignments				
Trimming	Total Positions	Aligned Accurately	Over-aligned	Greatly Misaligned
Posterior Decoding Column Cost	49.1%	85.9%	41.8%	30.3%
Neural Network	51.5%	89.2%	40.1%	35.0%

Table 9.5: Analysis of what positions are removed in alignment trimming from the validation alignments. Alignment positions were divided into categories according to the shift of the target residue relative to the FSSP structural alignments: *aligned accurately*, where the shift is zero; over-aligned; and *greatly misaligned*, shifted by five or more residues. For each category, shown is the number and percentage of positions remaining after trimming for global posterior-decoded FSSP alignments (top) and global Viterbi Target99 alignments (bottom). The trimming thresholds used are those derived in Section 9.2.2

in the neural network. It is likely that this is the main source of information, and the other inputs might not be as significant as they appeared. Where the two methods appear to yield different results, much of the distinction might come from the differences in their threshold values. Because posterior decoding column cost is far simpler, it is the superior predictor.

9.2.4 Further investigation of posterior decoding column cost

Previously, we examined the effects of trimming alignments using posterior decoding column cost on global alignments built with two methods: one of our best methods, global posterior-decoded FSSP alignments; and a more general method, global Viterbi Target99 alignments. We saw that posterior decoding information was effective in both cases; alignment trimming removed most of the over-aligned positions and many of the greatly misaligned positions while retaining most of the accurate positions. Now, we shall address the question of how this trimming method fares on alignments built with other methods.

For both Target99 and FSSP seed alignments, we tested alignment trimming by posterior decoding column cost on alignments built with the following methods: global posterior decoding alignment, local posterior decoding alignment, and global Viterbi alignment. According to the results in Chapter 5, all three alignment strategies could benefit from alignment trimming, as all have substantially larger optimal subalignment scores than shift scores. We did not explore local Viterbi alignment because the results in Chapter 5 for two reasons. First, the gap between shift score and optimal subalignment score is not as large as for the other alignment methods; it does not stand to benefit as much from alignment trimming. Second, local Viterbi alignments tend to be short compared to global Viterbi alignments or posterior-decoded alignments; if they were trimmed further,the remaining alignment might not be long enough to be useful.

Previously, we derived results for global posterior decoding alignment with FSSP seed alignments and global Viterbi alignment with Target99 seed alignments. For the work in this section, we derived results for four additional methods: global Viterbi alignment and local posterior

Alignment Method	Shift Score	Optimal Subalignment	Trimmed Alignment	Trimming Threshold
FSSP Global Posterior Decoding	0.402	0.533	0.417	1.9
FSSP Local Posterior Decoding	0.366	0.466	0.377	1.8
FSSP Global Viterbi	0.337	0.493	0.376	1.2
Target99 Global Posterior Decoding	0.271	0.410	0.293	0.9
Target99 Local Posterior Decoding	0.270	0.338	0.271	1.2
Target99 Global Viterbi	0.259	0.408	0.262	0.7

Table 9.6: Shown are trimming results on the threshold selection alignments for each of the alignment methods tested. The results shown for each method are the shift score, optimal subalignment score, trimmed alignment score,and trimming threshold. The alignments were trimmed by removing positions with a posterior decoding column cost that exceeded the threshold. For comparison, the local Viterbi alignment scores for these structures are 0.269 for FSSP seed alignments and 0.251 for Target99 seed alignments.

decoded alignment with FSSP seed alignments, and global and local posterior decoding alignment with Target99 seed alignments. Along with the results on these four new methods, we reported the results of the two previous ones for the convenience of the reader. For all six cases the alignments were built with the `w0.5` weighted build method.

For consistency with the previous results, we followed the same experimental scheme. First, we built alignments for the threshold selection structures listed in Section 9.1.3. For each alignment building method, we experimented with trimming the alignments according to the posterior decoding column cost, and empirically selected a threshold value for each alignment building method. Next, we built alignments for the validation structures listed in Section 9.1.3, and trimmed each alignment according to the threshold determined for its building method.

Table 9.6 lists the thresholds selected for each building method along with a number of average shift scores to reflect trimming effectiveness: the score for untrimmed alignments, the score for trimmed alignments, and the optimal subalignment score for untrimmed alignments. In all six cases, we found some threshold that yielded an improvement in alignment quality.

Recall from Chapter 5 that for the `w0.5` weighted build, local Viterbi alignment performed slightly better than global Viterbi alignment with Target99 seed alignments, and slightly worse than global Viterbi with FSSP seed alignments. Here, local Viterbi alignment performs substantially

		Shift Score			
Alignment Method	Score	FSSP	VAST	Yale	Closest
Fssp Global Posterior Decoding	Untrimmed	0.369	0.365	0.334	0.404
	Trimmed	0.377	0.371	0.340	0.412
	Optimal	0.512	0.464	0.437	0.520
Fssp Local Posterior Decoding	Untrimmed	0.311	0.309	0.289	0.340
	Trimmed	0.328	0.323	0.302	0.355
	Optimal	0.421	0.384	0.365	0.424
FSSP Global Viterbi	Untrimmed	0.297	0.294	0.266	0.333
	Trimmed	0.339	0.323	0.295	0.364
	Optimal	0.462	0.418	0.392	0.471
Target99 Global Posterior Decoding	Untrimmed	0.231	0.242	0.242	0.278
	Trimmed	0.241	0.247	0.247	0.284
	Optimal	0.387	0.358	0.360	0.401
Target99 Local Posterior Decoding	Untrimmed	0.214	0.216	0.223	0.251
	Trimmed	0.218	0.224	0.227	0.254
	Optimal	0.304	0.272	0.280	0.312
Target99 Global Viterbi	Untrimmed	0.193	0.207	0.206	0.238
	Trimmed	0.222	0.225	0.230	0.257
	Optimal	0.352	0.330	0.330	0.368

Table 9.7: Results of alignment trimming by alignment method, validation set. For each alignment method, the alignments were trimmed according to posterior decoding column cost and the threshold values listed in Table 9.6. This table scores the alignments against each set of structural alignments before and after trimming.

worse than global Viterbi. The set of structure pairs used here is harder than the set used in Chapter 5. Recall further that the difference between local and global alignment is that in local alignment, low-scoring regions at the beginning and end of the alignment can be omitted without incurring large gap costs. Because of the difficulty of these alignment tests, much of the alignments tend to be low-scoring. In consequence, the local alignments tend to be very short, and often misaligned. That is the reason why the global alignment scores are stronger in this case, and not in Chapter 5. The score for local Viterbi alignment with FSSP seed alignments might strike the reader suspicious, strangely like the scores for Target99 seed alignments. However, it is not the result of some accidental substitution of seed alignment. When local Viterbi alignments were built using FSSP seed alignments, the alignments were often entirely misaligned. Because the structures included in this set tend to have a global structural similarity, global alignment is more likely to find a correct alignment on the basis of a small region of high signal.

Table 9.7 assesses the performance of the alignment trimming against four sets of structural alignments for the validation structures. For each alignment method, the table lists the shift score and optimal subalignment score of the alignment before trimming, and the shift score of the alignment after trimming. In all cases, alignment trimming yielded some improvement in alignment quality. For global Viterbi alignment, it yielded a substantial improvement.

9.2.5 Validation results compared with other methods

To put these results into perspective, we applied the completed alignment trimming methods to the 170 pairs of structures used in Chapter 5 to optimize SAM alignment methods. We had concluded Chapter 5 by describing the five methods with best overall performance. Now, as we conclude this chapter, we will compare the best of those methods to the trimming methods described in this chapter. Table 9.8 shows the best results overall on FSSP alignments, and Table 9.9 shows the best results overall on global Viterbi Target99 alignments.

The three best methods for aligning remote homologs to FSSP seed alignments involve the alignment trimming methods of this chapter. Either the neural network or the posterior decoding column cost yields a substantial improvement over untrimmed alignments. All of the seven best methods involve some form of alignment trimming, either the trimming methods in this chapter or "poor man's alignment trimming", the consensus between two different alignments.

For Target99 seed alignments, the best method is a consensus method involving one seed alignment of the template sequence plus homologs and one seed alignment of the target sequence plus homologs. After that, the next three methods involve the alignment trimming methods of this chapter. Most of the best methods involve some form of alignment trimming. The only exception is posterior-decoded local alignment.

Trimming did not greatly benefit posterior-decoded local alignment. For FSSP seed alignments, it yielded a very minor improvement, though not enough to make the list of the best methods. For Target99 seed alignments, it slightly worsened alignment quality. Recall that Table 9.7 showed

Shift Score	Description
0.519	Character-based (`-adp5`) posterior-decoded w0.5 global, trimmed according to neural network predictions
0.508	w0.5 global Viterbi, trimmed according to the posterior decoding column cost
0.507	Character-based (`-adp5`) posterior-decoded w0.5 global, trimmed according to the posterior decoding column cost
0.503	Consensus between `-adp5` posterior-decoded w0.5 global and `-adp5` posterior-decoded fw0.5 global
0.501	Consensus between `-adp5` posterior-decoded fw0.5 global and `-adp5` posterior-decoded w0.5 global
0.499	Consensus between `-adp5` posterior-decoded fw0.5 global and fw0.5 global Viterbi
0.499	Consensus between `-adp5` posterior-decoded w0.5 global and fw0.5 global Viterbi

Table 9.8: Shown are the seven best methods for building an alignment with FSSP seed alignments. This assessment includes both methods involving alignment trimming and those not involving alignment trimming. For comparison, w0.5 local Viterbi alignment yielded an average shift score of 0.415. The shift scores for untrimmed posterior decoding alignment were 0.486 for global alignment and 0.491 for local alignment. Other methods that did not score as well are shown in Chapter 5.

Shift Score	Description
0.402	Consensus between an `-adp5` posterior-decoded w0.5 global alignment of the target sequence to the template family and and `-adp5` posterior-decoded w0.5 global alignment of the template sequence to the target family
0.397	w0.5 global Viterbi, trimmed according to the posterior decoding column cost
0.396	w0.5 global Viterbi, trimmed according to the neural network predictions
0.396	`-adp5` posterior-decoded w0.5 global alignment, trimmed according to the posterior decoding column cost
0.395	Consensus between an `-adp5` posterior-decoded w0.5 global alignment of the target sequence to the template family and a Viterbi w0.5 global alignment of the template sequence to the target family
0.395	An `-adp5` posterior-decoded w0.5 local alignment
0.383	`-adp5` posterior-decoded w0.5 local alignment, trimmed according to the posterior decoding column cost

Table 9.9: Shown are the seven best methods for building an alignment with Target99 seed alignments. This assessment includes both methods involving alignment trimming and those not involving alignment trimming. For comparison, w0.5 local Viterbi alignment yielded a shift score of 0.368, untrimmed global posterior decoding scored 0.377, and untrimmed global Viterbi scored 0.356. Other methods that did not score as well are shown in Chapter 5.

that trimming yielded only a slight improvement for local posterior decoding. Since the performance gain did not carry over to this validation set, it appears to be a fluke of one dataset rather than a significant trend. However, trimming made a consistent improvement to global alignment with both the Viterbi and posterior decoding alignment algorithm.

In summary, based on the data from this chapter and from Chapter 5, the best way to align two remote homologs is as follows. If there is an FSSP alignment available for the template sequence with at least a couple homologs, use it as a seed alignment; preliminary investigation suggests that the performance advantage of FSSP seed alignments is realized with only a few homologous sequences. If no FSSP seed alignment is available, use a Target99 seed alignment. Align the target sequence to the seed alignment with global posterior decoding, and trim it according to the posterior decoding column cost. If one is using a Target99 seed alignment, it might be instructive to align the template sequence to the target sequence and homologs with global posterior decoding, and study the consensus of the two alignments. However, most of the performance gain will be realized by trimming one posterior-decoded global alignment.

9.3 Summary

Alignment methods make mistakes. However, some of the mistakes can be identified. We have explored four methods for identifying suspect positions in HMM-generated alignments: three simple predictors and a more complex neural network. We have applied these predictors successfully on hard remote homologs. When we have removed positions predicted as reliable, we have seen substantial improvement in overall alignment quality, with an experimental process involving an number of careful experimental safeguards. The results shown here should hold up in practice.

A simple predictor explored was the number of template sequence positions to the nearest residue not in a loop. When we trimmed out positions aligning to the middle of long loops, this simple predictor yielded modest improvement, but improvement nonetheless.

A second simple predictor explored was the column log odds: given the probability of the amino acid conditioned on the residues appearing in the column, and given the unconditioned probability of the amino acid, this predictor measures the log likelihood of the conditioned and unconditioned probability for the target amino acid aligned to the column. This predictor yielded modest improvement on good-quality alignments, but failed to yield improvement as the quality of the untrimmed alignments dropped.

The predictors that we found most successful both focus on posterior decoding information. Posterior decoding is a member of the family of near-optimal alignment algorithms: algorithms that estimate alignment information according to many potential alignments of the target sequence and the seed alignment. For a target sequence residue aligned to some column in the alignment, posterior decoding reflects the probability of the residue aligning to the column given all possible alignments of target sequence and seed alignment. One simple predictor we explored was the posterior decoding column cost, the cost with which each target residue was aligned to each column. A simple threshold on this cost yielded a 3% improvement over one of our best alignment methods, and a 15% improvement on our more commonly-used alignment methods. Our second successful predictor was a neural network that accepted as input this cost, plus other information. It yielded consistent results to the posterior decoding column cost, and might merely be a more complex representation of this cost. In analysis of the positions retained after trimming, we saw that the two posterior decoding-based methods removed 65 to 70% of the columns misaligned by more than a few residues and 60% of the over-aligned positions while retaining 85% and 89% of the accurate positions.

We investigated posterior decoding column cost for trimming additional alignment methods – global Viterbi, global posterior decoding, and local posterior decoding – for both Target99 and FSSP seed alignments. It did not appear to make a significant difference on local posterior decoding alignments. In some cases, it yielded a minor improvement; in others, it worsened the alignment somewhat. however, it yielded a reliable improvement to global alignment for both

Viterbi and posterior decoding alignment methods. Many of our best alignment building strategies involve trimming a global alignment according to posterior decoding column cost.

At this time, we have not built a general-purpose predictor. Our focus has been proving if this methodology can work. At this point, we can say that the methodology can work. When we developed separate predictors for different alignment methods, we arrived ultimately at very similar predictors. This finding begs for a general-purpose predictor, which is in the plans.

Chapter 10

Conclusion

The most concise summary of this thesis is as follows: near-optimal sequence alignment information is valuable. This point is apparent in many places within this thesis.

In Chapters 5 and 6, we showed that posterior decoding yields significantly better HMM-based alignments than the standard Viterbi algorithm. The standard algorithm estimates the likelihood of aligning each residue to each column according to the column's amino acid posterior probability distribution. Posterior decoding, in contrast, estimates these likelihoods according to all possible alignments of target sequence and seed alignment: near-optimal sequence alignment information.

In Chapter 5, we demonstrated a simple alignment trimming method that yielded significant improvement in overall alignment quality. That method is to start with two different alignments, built with slightly different methods, and remove the positions where they disagree significantly. This is a simple example of near-optimal alignment information, but an example nonetheless.

In the related work section of Chapter 6, we saw that the literature comparing sequence alignment methods is often contradictory. There are many papers in which the authors report favorable results with their own methods, but other authors cannot always reproduce these results.

The papers by Marcella McClure and co-authors [77, 118, 119] have the major advantage that the authors have not developed an alignment method of their own; therefore, they are more impartial. In addition, they carefully optimize all methods before comparison, and therefore use the methods to their best advantage. In their most recent investigation [77], the method that they found most effective at aligning conserved motifs was PROBE [132], a Gibbs sampling method. Gibbs sampling involves sampling various alignments according to some probability function. Once again, this is an application of near-optimal alignment information.

When considering how to best validate alignments produced by hidden Markov models, we explored pairwise contact potentials. These potentials report the statistical likelihood that two amino acids found in proximity in a predicted structure, as compared to actual protein structures. Pairwise contact potentials have a long history as one component of successful threading algorithms. The HMM-based alignments described in this thesis are built without structural information. For this reason, structural information should be effective for validating them. However, when we studied the amount of significant mutual information in pairwise contacts, the amount past what can be attributed to small sample-size effects, we observed that pairwise contact potentials are significant but weak. We found that the most substantial source of pairwise information was tertiary contacts between β-strand residues, and chose to explore these further. We built a potential function to estimate when two β-strands are in contact given their amino acid sequence. Our potential function showed some signal, but nothing strong enough to warrant application in alignment validation. Even in the best case, we found pairwise contact potentials to be significant yet weak.

In Chapter 9, we explored prediction of the reliability of individual alignment positions. Given an alignment in which some target sequence was aligned to a seed alignment of a template sequence and its homologs, we examined the positions where the target residues align to the columns of the seed alignment. Our predictions were able to identify close to 90% of the accurately-aligned positions, weeding out 70% of the positions misaligned by more than a few residues and 60% of the over-aligned positions. The most significant source of information for this prediction was the posterior decoding column cost, the cost of aligning each target sequence residue to each column in the seed alignment, given all possible alignments of target sequence and seed alignment. Once again, this is an application of near-optimal sequence alignment information.

What makes near-optimal alignment information so effective? According to Gibbs sampling authority Chip Lawrence in a related tutorial [107], when one estimates the overall probability of a sequence alignment, the numbers are large in one sense, small in another. In the first sense, the probability of the alignment might be very significant relative to the null model. In the second sense, the absolute probabilities are still very small numbers; a strong alignment might have a probability on the order of 1%. According to the author's experience in applying probability theory to horse racing, if no horse has more than a 1% chance of winning, you should not bet on one single horse. Instead, you should select a number of horses to wager on, weighing the amount wagered according to the chance that the horse will win. In a biological context, this means that when aligning a sequence to a model, one should look for a way to harness the information of various alignments of the sequence to the model to obtain an overall consensus.

In a biological perspective, why does this approach work? Consider the evolution of a protein, relative to homologous proteins. Certain positions are more conserved in evolution, namely those important to the structure of the protein family, and a good alignment should align these positions accurately. Conservation is not consistent across protein families: there are varied regions, and there are conserved regions corresponding to conserved domains. This is even true with very remote homologs [159]. However, alignment estimation is no trivial task, and is notoriously sensitive to settings of parameters such as gap costs. A slight miscalculation of such parameters can align one position incorrectly. Because amino acids within a protein are sequential, this mistake tends to be propagated to later residues, yielding large misaligned regions. As observed by Vingron and co-authors [188], slight variation of alignment parameters will yield a number of different alignments, with some interesting patterns. There will tend to be regions where most alignments agree: most alignments will place certain target sequence residues in exactly the same column. There will also be regions where alignments disagree, where there is no one consensus position. Finally, there are residues that are not aligned to the same position by an overall consensus, but are aligned to one position most frequently. Recall that whenever a sequence is aligned, the alignment reflects two things: the parameter settings and the biological data. When one looks at various different alignments, the biological data is the same; it is the parameter values that have changed. When one aligns a sequence according to the consensus of many different alignments, the alignment is less sensitive to the parameter settings and more focused on the biological data. Therefore, it aligns with greater probability where the biological signal is strongest: at the conserved domains.

Chapter 11

Future Work

The work in this thesis demonstrates a proof-of-concept, showing that we can predict the accuracy of alignment positions. When we apply these accuracy predictions, we can yield a significant improvement in overall alignment quality, even when we start with alignments built with the best methods we have available. However, to put this work into practice, a few steps remain.

First, in pursuit of the ultimate goal of a general-purpose predictor, we must train a predictor according to alignments made by a variety of methods, not one method. When we built predictors for two different classes of alignments, the final, optimized predictors were very similar. This bodes well for the goal of building a general-purpose predictor. This predictor will not necessarily generate better results than our best alignment method, because it will not be optimized to beat it. Even so, it should generate useful information on what columns are likely to be misaligned or over-aligned, information that could be very useful to a person or application seeking to build on the alignment.

Second, no method is valuable unless it is used, and a method cannot be used until it is made available. In the short-term, I am planning to put up a web server for use by the larger research community. I look forward to turning my focus to that project, as soon as a certain writing project no longer occupies the center of my life.

Third, removing positions strictly by a score and a threshold is not the right thing to do. Long regions of the alignment might score close to the threshold value, dipping below the threshold briefly or rising above it briefly. This can yield alignments with short aligned regions or small gaps in structurally-unlikely positions, neither of which make much sense biologically. While application of a score and threshold is sufficient for a proof-of-concept, a better approach would be to devise some heuristic to remove large, contiguous regions from the alignment rather than single columns. This heuristic could permit gaps according to the secondary structure of the template sequence or predicted secondary structure of the target sequence, or it could involve a simple gap penalty system. Both approaches have their merits.

Fourth, while we have made progress toward to goal of predicting the reliability of positions in an alignment, there is a larger goal of predicting overall alignment reliability. If the reliability of individual positions of an alignment can be predicted accurately – which they can – then it should be feasible to combine these predictions to form an overall prediction of alignment quality.

Finally, there is work to be done concerning applying the probability of target alignment reliability to the applications of sequence alignment. Most alignment applications treat all positions in the alignment with equal confidence. If we could tell the application what positions are most trustworthy, we should be rewarded with greater application accuracy. The proper way to handle this information will vary with the application. For example, phylogenic analysis and prediction of functional residues depend on proper analysis of the non-conserved positions in the alignment. For such applications, measures of the variation in the alignment might be scaled according to the probability of position reliability. As another example, homology modeling works best with a long alignment, because a portion of the sequence not aligned means a portion not modeled. Therefore, one probably would not want to trim the alignments, but rather to prioritize the positions modeled according to the alignment confidence. By allowing the positions aligned with greatest reliability to assume their preferred romaters, the model would reflect the most accurate regions of the alignment, and predicted alignment reliability would limit the immense homology modeling search space.

Appendix A

Glossary of biological terms

Å: Symbol for angstrom, a unit of measure equal to 10^{-10} meters.

Ab Initio prediction: A family of algorithms that predict the structure of a protein "from scratch", without predicting its relation to some other protein.

Accessibility: See *solvent accessibility*.

Active site: The amino acids of a protein that interact directly with some other molecule when the protein performs its function within the cell.

Alignment: A textual arrangement of two or more proteins, arranged to indicate regions of observed sequence similarity or observed or predicted structural similarity.

Alignment length: The number of columns in the alignment.

α-carbon: The carbon atom within an amino acid that the side chain branches off from. The α-carbon is part of the polypeptide backbone.

α-helix: A secondary structure class in which the amino acids in a contiguous segment of the protein arrange themselves in a right-handed corkscrew-like conformation.

Angstrom: A unit of length defined as 10^{-10} meters. Symbol: Å.

Antiparallel β-sheet: A β-sheet in which the neighboring β-strands point in opposite directions relative to the N- and C-termini.

Backbone: The "spinal column" of a protein, consisting of a contiguous linkage of the nitrogen, α-carbon, and carbonyl carbon atoms of the protein's amino acids.

Backbone atoms: The group of atoms, common to all amino acids, which together form the backbone of a protein: nitrogen, α-carbon, carboxyl carbon, and carboxyl oxygen.

β-carbon: For all amino acids except glycine, the *beta*-carbon is the first atom in the side chain, bonded directly to the α-carbon.

β-sheet: A secondary structure class in which the protein folds back on itself to form a flat or slightly rounded surface, with short segments of the protein chain linking with short segments from other regions of the protein chain.

β-strand: A contiguous segment of protein forming one portion of a β-sheet.

β-turn: A short, 180-degree turn between two antiparallel β-strands close to each other in the protein sequence.

C-terminus: The end of a protein chain.

Chain: The contiguous sequence of amino acids forming one single protein.

Column: Within an alignment, a *column* is a vertical arrangement of amino acids from different proteins, indicating similarity in sequence or structure.

Comparative modeling: See *homology modeling.*

Conserved: Unchanged through evolution.

Contact: Refers to two or more amino acids close enough to interact in the tertiary protein structure.

Contact potential: The statistical likelihood of two or more amino acids to be in contact, given such information as their amino acid type.

Covalent bond: A strong, semi-permanent chemical bond between two atoms.

Coverage: An alignment quality measure describing the proportion of target sequence residues that were aligned to residues in the template sequence.

Deletion: A gap in the alignment of a sequence to another sequence or to a sequence family.

De novo: From scratch.

Disulfide bridge: A covalent bond between the sulfur atoms in two different cysteine amino acids.

Divergent: Differing, or changed through evolution.

Docking: During the functioning of a protein, *docking* refers to the precise interaction between the active site residues of the protein and the atoms of the other compound.

Domain: A compact subunit of the three-dimensional structure of a protein.

Exposure: See *solvent exposure.*

Fold: A protein's structural class or family.

Fold recognition: A class of algorithms that predict the structure of a protein by predicting its relation to some protein of known structure.

FSSP: A database of families of proteins of similar structures [68].

Function: The protein's role within the cell.

Functional residues: The residues involved directly with the protein's function.

Gap: When one sequence is aligned to another, a *gap* is a break in the alignment, indicating that the second sequence contains a region of one or more amino acids that do not correspond to anything in the first. Gaps are typically indicated by dashes in the sequence in the columns where it does not align.

Homolog: A protein descended from a common ancestor.

Homology: Similarity and likely evolutionary relation between two or more proteins.

Homology modeling: A class of algorithms that yield very detailed predictions of the structure of a target protein based on its alignment to a template protein and the template protein structure. Homology modeling is usually applied only to cases where there is clear homology between the template and target sequences.

HSSP: A database of families of proteins of similar sequence [158].

Hydrogen bond: A weak bond formed by the attraction of the partially-charged hydrogen of one molecule and some partially-charged atom on another molecule.

Hydrophilic: Describes an amino acid or a portion of a protein structure that can interact with water.

Hydrophobic: Describes an amino acid or a portion of a protein structure that cannot interact with water.

Indel: A break in an alignment, either an insert or a deletion.

Insertion: When one sequence is aligned to another, an *insertion* is a break in the alignment indicating that the first sequence contains one or more amino acids that do not correspond to anything in the second. Insertions are typically indicated by lowercase amino acids in the inserting sequence, and dots in the corresponding columns of the other sequence.

Loop: A secondary structure class in which the amino acids lack regular structure. Loops usually connect two secondary structural elements.

Macromolecule: A large molecule, such as a protein, RNA, or DNA.

Motif: A short region of a protein sequence with previously recognized sequence similarity to segments found in other proteins.

Multiple alignment: An alignment of more than two sequences. Most often, a multiple alignment contains related sequences, arranged to represent common features in their sequence family.

N-terminus: The beginning of a protein chain.

NMR spectroscopy: A method of determining the structure of a protein using Nuclear Magnetic Resonance, by concentrating the protein at slightly-lowered pH, applying RF pulses, and analyzing the emitted radiation.

Optimal superposition: A superposition of the atoms of one protein onto those of another protein that most emphasizes the structural similarity between the two proteins.

Pairwise alignment: An alignment of two protein sequences.

Pairwise contact potential: The statistical likelihood of two amino acids to interact, given such information as their amino acid type.

Parallel β-sheet: A β-sheet in which the β-strands point in the same direction, relative to the N- and C-termini.

PDB: The Protein Data Bank, the world's central repository of protein structure information [13], located at **http://www.rcsb.org/pdb/**.

Peptide: A small protein of approximately 2-40 amino acids.

Peptide bond: The amide linkage between two amino acids in a peptide or protein.

Phylogenic analysis: Prediction of the precise evolutionary relations within a family of related proteins or DNA sequences.

Phylogeny: The evolution of a family of related proteins or DNA sequences.

Polypeptide: A large peptide.

Potential function: An energy function that predicts the conformation of a region of the protein, or the stability of one or more amino acids within a predicted structure.

Primary structure: The sequence of amino acids forming a protein.

Protein: A compound common to all living matter, consisting of multiple amino acids linked in sequence by covalent bonds.

Quaternary structure: The three-dimensional structure of a protein, consisting of multiple protein chains arranged into a single complex.

Random coil: See *loop*.

Residue: General term for a subunit of a macromolecule. In the context of this thesis, *residue* refers to one amino acid within a protein.

RMSD: See *RMS deviation*.

RMS deviation: The root mean-squared deviation, in Å, between the positions of the backbone atoms in two protein structures.

Rotamer: One of many common conformations adopted by an amino acid side chain.

Sequence positions: Amino acids in a protein sequence.

Shift: A measure of alignment error describing the number of sequence positions between where a residue was aligned and where it should have been aligned.

Side chain: The portion of an amino acid that branches off the protein backbone. Each type of amino acid has a different side chain, and the side chain is what gives the amino acid its characteristics.

Secondary structure: The arrangement of amino acids within short regions of a protein chain into specific, regular patterns, as identified by the angles between their backbone atoms.

Secondary structural elements: The sections of a protein which adopt a regular secondary structure such as α-helix or β-strand, as opposed to those which form loops or random coil.

Sequence: The series of amino acids forming a protein.

Sequence alignment: An alignment of two or more protein sequences, arranged to indicate regions of similar sequence.

Solvent: That which dissolves. Within this thesis, solvent refers to the aqueous solution in which most proteins exist.

Solvent accessibility: Describes the extent to which a residue within a protein is exposed to the solvent outside the protein, as computed by the DSSP program [88]

Solvent exposure: Describes the extent to which a residue within a protein is exposed to the solvent outside the protein. Within this thesis, solvent accessibility and solvent exposure describe similar measures computed with different algorithms.

Structure determination: The process of ascertaining the structure of a protein by means of X-ray crystallography or NMR spectroscopy.

Structural alignment: An alignment built by examining the structure of two or more proteins to highlight regions of similar structure.

Structural homolog: Typically refers to a sequence with very similar structure but weak sequence similarity.

Subfamily: A group of protein structures with obvious structural similarity, and generally the same function and the same active site residues.

Superfamily: A group of protein structures consisting of one or more subfamilies. Proteins in the same superfamily have similar structures and similar functions but not necessarily the same active site residues.

Target sequence: In a structure prediction scenario, a protein whose structure is not known, or whose known structure is ignored during the structure prediction process.

Template sequence: In a structure prediction scenario, a protein whose structure is known, and is predicted to be similar to that of the target sequence.

Tertiary structure: The three-dimensional structure of a single protein.

Threading: A class of algorithms that predict the structure of a target protein by predicting relation to some template protein of known structure, and searching for an arrangement of the target protein residues into the template structure such that predicted energy is minimized. In this context, minimizing energy can be viewed as equivalent to maximizing stability.

X-ray crystallography: A method of determining the structure of a protein by growing crystals of the protein, applying X-ray beams to the crystals, and analyzing their diffraction pattern.

Bibliography

[1] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *NAR*, 25:3899–3402, 1997.

[2] Stephen F. Altshul, Warren Gish, Webb Miller, Myers Eugene W., and Lipman David J. Basic Local Alignment Search Tool. *JMB*, 215:403–410, 1990.

[3] M. A. Andrade and C. Sander. Bioinformatics: from genome data to biological knowledge. *Current Opinion in Biotechnology*, 8:675–83, 1997.

[4] C.B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.

[5] A. Apostolico and R. Giancarlo. Sequence alignment in molecular biology. *Jour. Comp. Biol.*, 5(2):173–96, 1998.

[6] Timothy L. Bailey and Charles Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *ISMB-94*, pages 28–36, Menlo Park, CA, 1994. AAAI/MIT Press.

[7] P. Baldi, Y. Chauvin, T. Hunkapillar, and M. McClure. Hidden Markov models of biological primary sequence information. *PNAS*, 91:1059–1063, 1994.

[8] Christian Barrett, Richard Hughey, and Kevin Karplus. Scoring hidden Markov models. *CABIOS*, 13(2):191–199, 1997.

[9] G. Barton. Protein sequence alignment techniques. *Acta Crystallographica. Section D: Biological Crystallography*, 54(1):1139–46, 1998.

[10] G. J. Barton and M. J. Sternberg. A strategy for the rapid multiple alignment of protein sequences. *JMB*, 198(2):327–37, 1987.

[11] P. A. Bates and M. J. E. Sternberg. Model building by comparison at CASP3: using expert knowledge and computer automation. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):47–54, 1999.

[12] A. Baucom, 1997. Personal communication with Albion Baucom of the Gregret laboratory, Chemistry Department, University of California, Santa Cruz.

[13] F.C. Bernstein, T. F. Koetzle, G. J. Williams, E. E. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: a computer-based archival file for macromolecular structures. *JMB*, 112:535–542, 1977.

[14] Christopher M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, 1995.

[15] J. U. Bowie, R. Lüthy, and D. Eisenberg. A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, 253:164–170, 1991.

[16] S. E. Brenner, D. Barken, and M. Levitt. The PRESAGE database for structural genomics. *NAR*, 27(1):251–3, 1999.

[17] S. E. Brenner, C. Chothia, and T. J. P. Hubbard. Assessing sequence comparison methods with reliable structurally identified distant evolutionary relationships. *PNAS*, 95(11):7073–8, 1998.

[18] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing, Algorithms, Architectures and Applications. Proceedings of the NATO Advanced Research Workshop*, pages 227–36. Springer-Verlag, 1990.

[19] P. Briffeuil, G. Baudoux, C. Lambert, X. De Bolle, C. Vinals, E. Feytmans, and E. Depiereux. Comparative analysis of seven multipl protein sequence alignment servers: clues to enhance reliability of predictions. *Bioinformatics*, 14(4):357–66, 1998.

[20] L. Brocchieri and S. Karlin. A symmetric-iterated multiple alignment of protein sequences. *JMB*, 276(1):249–64, 1998.

[21] S. Bryant. Evaluation of Threading Specificity and Accuracy. *Proteins: Structure, Function, and Genetics*, 26(2):172–85, 1996.

[22] S. H. Bryant and C. E. Lawrence. An empirical energy function for threading protein sequence through the folding motif. *Proteins: Structure, Function, and Genetics*, 16(1):92–112, May 1993.

[23] C. Bystroff and D. Baker. Blind Predictions of local protein structure in CASP2 targets using the I-sites library. *Proteins: Structure, Function and Genetics, Suppl.*, 1:167–171, 1997.

[24] From Gene Struction to Function Project on the World Wide Web, 2000. http://s2f.carb.nist.gov/cgi-bin/dbstatus.cgi.

[25] G. Casari and M. J. Sippl. Structure-derived hydrophobic potential, Hydrophobic potential derived from X-ray structures of globular proteins is able to identify native folds. *JMB*, 224(3):725–32, 1992.

[26] C. Chothia. Proteins. One thousand families for the molecular biologist. *nature*, 357(6379):543–4, 1992.

[27] S. Y. Chung and S. Subbiah. A structural explanation for the twilight zone of protein sequence homology. *Structure*, 4(10):11123–7, 1996.

[28] F. Cohen, 1999. Personal communication with Fred Cohen, Departments of Medicine and Pharmacology, University of California, San Francisco.

[29] L. Lo Conte and T. Smith. Visible Volume: a Robust Measure for Protein Structure Characterization. *JMB*, 273(1):338–48, 1997.

[30] A. P. Cootes, P. M. Curmi, R. Cunningham, C. Donnelly, and A. E. Torda. The dependence of amino acid pair correlations on structural environment. *Proteins: Structure, Function, and Genetics*, 32(2):175–89, 1998.

[31] T. M. Cover and J. A. Thomas. *Elements of information theory.* Wiley, 1938.

[32] S. Dalal, S. Balasubramanian, and L. Regan. Protein alchemy: changing beta-sheet into alpha-helix. *Nature Structural Biology*, 4(7):548–52, 1997.

[33] E. Depiereux, G. Badoux, P. Briffeuil, I. Reginster, X. De Bolle, C. Vinals, and E. Feytmans. Match-box_server: a multiple sequence alignment tool placing emphasis on reliability. *CABIOS*, 13(3):249–256, 1997.

[34] K. A. Dill. Dominanat forces in protein folding. *Biochemistry*, 39(31):7133–55, August 1990.

[35] J. Dopazo. A new index to find regions showing an unexpected variability or conservation in sequence alignments. *CABIOS*, 13(2):313–7, 1997.

[36] Inna Dubchak, Ilya Muchnik, and Sung-Hou Kim. Protein Folding Class Predictor for SCOP: Approach Based on Global Descriptors. In *Proceedings, 5th International Conference on Intelligent Systems for Molecular Biology*, pages 104–108, Jun 1997.

[37] R. L. Dunbrack. Comparative modeling of CASP3 targets using PSI-BLAST and SCWRL. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):81–7, 1999.

[38] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[39] S. Eddy. Multiple alignment and multiple sequence-based searches. *Trends Guide to Bioinformatics*, supplement, 1998.

[40] S. R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–63, 1998.

[41] Sean Eddy. Multiple alignment using hidden Markov models. In Christopher Rallings et al., editors, *ISMB-95*, pages 114–120, Menlo Park, CA, July 1995. AAAI/MIT Press.

[42] S.R. Eddy. Hidden Markov models. *Curr. Opin. Struct. Biol.*, 6(3):361–365, 1996.

[43] S.R. Eddy, G. Mitchison, and R. Durbin. Maximum Discrimination Hidden Markov Models of Sequence Consensus. *J. Comput. Biol.*, 2:9–23, 1995.

[44] E. Einstein, G. L. Gilliland, O. Herzberg, J. Moult, J. Orban, R. J. Poljak, L. Banerjei, D. Richardson, and A. J. Howard. Biological function made crystal clear – annontation of hypothetical proteins via structural genomics. *cobiotech*, 11:25–30, 2000.

[45] D. F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.

[46] Daniel Fischer, Christian Barrett, Kevin Bryson, Arne Elofsson, Adam Godzik, David Jones, Kevin Karplus, Lawrence A. Kelley, Robert M. MacCallum, Krzysztof Pawłowski, Burkhard Rost, Leszek Rychlewski, and Michael Sternberg. CAFASP-1: Critical Assessment of Fully

Automated Structure Prediction Methods. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):209–217, 1999.

[47] T. P. Flores, C. A. Orengo, D. S. Moss, and J. M. Thornton. Comparison of conformational characteristics in structurally similar proteins. *Protein Sci.*, 5(11):1811–26, 1993.

[48] V. Di Francesco, J. Garnier, and P.J. Munson. Protein topology recognition from secondary structure sequences: application of the hidden Markov models to the alpha class proteins. *JMB*, 267(2):446–463, 1997.

[49] E. Furuichi and P. Koehl. Influence of protein structure databases on the predictive power of statistical pair potentials. *Proteins: Structure, Function, and Genetics*, 31(2):139–49, 1998.

[50] Mark Gerstein and Michael Levitt. Comprehensive assessment of automatic structural alignment against a manual standard, the SCOP classification of proteins. *Protein Sci.*, 7:445–456, 1998.

[51] J. Gilbrat, T. Madej, and S. Bryant. Surprising similarities in structure comparison. *Curr. Opin. Struct. Biol.*, 6:377–85, 1996.

[52] G. Giribet and W. C. Wheeler. On gaps. *Molecular Phylogenetics and Evolution*, 13(1):132–43, 1999.

[53] A. Godzik, A. Kolinski, and J. Skolnick. Topology fingerprint approach to the inverse protein folding problem. *JMB*, 227(1):227–38, September 1992.

[54] O. Gotoh. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *JMB*, 264(4):823–38, 1996.

[55] O. Gotoh. Multiple sequence alignment: algorithms and applications. *Advances in Biophysics*, 36(1):159–206, 1999.

[56] L. Gregoret and A. Baucom, 1997. Personal communication with Lydia Gregoret and Albion Baucom, Chemistry Board, University of California, Santa Cruz.

[57] Michael Gribskov, Andrew D. McLachlan, and David Eisenberg. Profile Analysis: Detection of Distantly Related Proteins. *PNAS*, 84:4355–4358, July 1987.

[58] T. Grossman, R. Farber, and A. Lapedes. Neural Net Representations of Empirical Protein Potentials. In *Proceedings of Intelligent Systems for Molecular Biology*, pages 154–61. AAAI Press, 1995.

[59] W. N. Grundy, W. Bailey, T. Elkan, and C. Baker. Meta-MEME: Motif-based hidden Markov models of protein families. *CABIOS*, 13(4):397–406, 1997.

[60] M. Hendlich, P. Lackner, S. Weitckus, H. Floeckner, R. Froschauer, K. Gottsbacher, G. Casari, and M. J. Sippl. Identification of native protein folds amongst a large number of incorrect models. The calculation of low energy conformations from potentials of mean force. *JMB*, 216(1):167–80, November 1990.

[61] S. Henikoff, J. G. Henikoff, W. J. Alford, and S. Pietrokovski. Automated construction and graphical presentation of protein blocks from unaligned sequences. *Gene*, 163(2):GC17–26, 1995.

[62] Steven Henikoff and Jorja G. Henikoff. Position-based Sequence Weights. *JMB*, 243(4):574–578, November 1994.

[63] D. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, 1991.

[64] D. G. Higgins and P. M. Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73:237–44, 1988.

[65] D. G. Higgins, J. D. Thompson, and T. J. Gibson. Using CLUSTAL for multiple sequence alignments. *Methods Enzymol.*, 266:383–402, 1996.

[66] Liisa Holm and Chris Sander. Searching protein structure databases has come of age. *Proteins: Structure, Function, and Genetics*, 19(3):165–73, 1994.

[67] Liisa Holm and Chris Sander. The FSSP Database: Fold Classification based on Structure-structure Alignment of Proteins. *NAR*, 24(1):206–209, 1 Jan 1996.

[68] Liisa Holm and Chris Sander. Mapping the protein universe. *Science*, 273(5275):595–603, Aug 2 1996. This is the paper that the FSSP web site requests be cited.

[69] Liisa Holm and Chris Sander. Dali/FSSP classification of three-dimensional protein folds. *NAR*, 25:231–234, 1 Jan 1997.

[70] I. Holmes and R. Durbin. Dynamic programming alignment accuracy. *Jour. Comp. Biol.*, 5(3):493–504, 1998.

[71] E. S. Huang, S. Subbian, J. Tsai, and M. Levitt. Using a hydrophobic contact potential to evaluate native and near-native folds generated by molecular dynamics simulations. *JMB*, 257(3):716–25, April 1996.

[72] X. Huang. On global sequence alignment. *CABIOS*, 10(3):227–35, 1994.

[73] T. Hubbard, A. Murzin, S. Brenner, and C. Chothia. SCOP: a structural classification of proteins database. *NAR*, 25(1):236–9, January 1997.

[74] Tim Hubbard. RMS/Coverage graphs: a qualitative method for comparing three-dimensional protein structure predictions. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):15–21, 1999.

[75] Tim J.P. Hubbard. Use of β-strand Interaction Pseudo-potentials in Protein Structure Prediction and Modeling. In *Proceedings of the 27th Annual Hawaii International Conference on System Sciences*, pages 336–344, 1994.

[76] Tim J.P. Hubbard. Fold recognition and ab initio structure predictions using hidden Markov models and beta-strand pair potentials. *Proteins*, 23(3):398–402, November 1995.

[77] J. Hudak and M. A. McClure. A comparative analysis of computational motif-detection methods. In *Pacific Symposium on Biocomputing*, pages 138–49. World Scientific, 1999.

[78] Richard Hughey and Anders Krogh. Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *CABIOS*, 12(2):95–107, 1996. Information on obtaining SAM is available at `http://www.cse.ucsc.edu/research/compbio/sam.html`.

[79] C. Hung. A comparative study of some competitive learning algorithms for unsupervised training in image classification system. In *International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, volume 2, pages 740–5, Chinese University, Hong Kong, 1994.

[80] E. G. Hutchinson, R. B. Sessions, J. M. Thornton, and D. N. Woolfson. Determinants of strand register in antiparallel beta sheets of proteins. *Protein Sci.*, 7:2287–300, 1998.

[81] F. Jacob. Evolution and tinkering. *Science*, 196:1161–6, 1977.

[82] F. Jeanmougin, J. Thompson, M. Gouy, D. Higgins, and T. Gibson. Multiple sequence alignment with Clustal X. *Trends in Biochemical Sciences*, 23(10):403–5, 1998.

[83] R. L. Jernigan and D. G. Covell. Conformations of folded proteins in restricted spaces. *Biochemistry*, 29(13):3287–94, April 1990.

[84] D. A. Jones, C. A. Orengo, and J. M. Thornton. Protein folds and their recognition from sequence. In M. J. Sternberg, editor, *Protein Structure Prediction*, chapter 8, pages 172–206. IRL Press, 1996.

[85] D. T. Jones. Progress in protein structure prediction. *Curr. Opin. Struct. Biol.*, 7(2):377–87, 1997.

[86] D. T. Jones. GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. *JMB*, 287(4):797–815, 1999.

[87] T. Alwyn Jones and Gerald J. Kleywegt. CASP3 Comparative Modeling Evaluation. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):30–46, 1999.

[88] W. Kabsch and C. Sander. A dictionary of protein secondary structure. *Biopolymers*, 22:2577–637, 1983.

[89] R. Karchin and R. Hughey. Weighting hidden Markov models for maximum discrimination. *Bioinformatics*, 14(9):772–782, 1998.

[90] K. Karplus, 1997. Personal communication with Kevin Karplus, Computer Engineering Board, University of California, Santa Cruz.

[91] Kevin Karplus. Regularizers for Estimating Distributions of Amino Acids from Small Samples. In *ISMB-95*, Menlo Park, CA, July 1995. AAAI/MIT Press.

[92] Kevin Karplus, Christian Barrett, Melissa Cline, Mark Diekhans, Leslie Grate, and Richard Hughey. Predicting Protein Structure using only Sequence Information. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):121–125, 1999.

[93] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden Markov Models for detecting Remote Protein Homologies. *Bioinformatics*, 14(10):846–856, 1998.

[94] Kevin Karplus, Kimmen Sjölander, Christian Barrett, Melissa Cline, David Haussler, Richard Hughey, Liisa Holm, and Chris Sander. Predicting Protein Structure using Hidden Markov Models. *Proteins: Structure, Function, and Genetics*, Suppl. 1:134–139, 1997.

[95] R. D. King, D. A. Clark, J. Shirazi, and M. J. E. Sternberg. On the use of machine learning to indentify topological rules in the packing of beta strands. *Protein Engr.*, 7(11):1295–1303, 1994.

[96] J. P. Kocher, M. J. Rooman, and S. J. Wodak. Factors influencing the ability of knowledge-based potentials to identify native sequence-structure matches. *JMB*, 235(5):1598–613, 1994.

[97] M. Koda. Neural network learning based on stochastic sensitivity analysis. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, 27(1):132–5, February 1997.

[98] W. A. Koppensteiner and M.J. Sippl. Knowledge-based potentials–back to the roots. *Biochemistry*, 63(3):247–52, 1998.

[99] K. K. Koretke, R. B. russell, R. R. Copley, and A. N. Lupas. Fold recognition using sequence and secondary structure information. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):141–8, 1999.

[100] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *JMB*, 235:1501–1531, February 1994.

[101] A. Krogh and S.K. Riis. Prediction of Beta Sheets in Proteins. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 917–923, Cambridge, MA, USA, 1996. MIT Press.

[102] D. Kulp, D. Haussler, M. G. Reese, and F. H. Eeckman. Integrating Database Homology in a Probabilistic Gene Structure Model. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing*, pages 232–244. World Scientific, New York, 1997.

[103] D. Kulp, D. Haussler, M.G. Reese, and F. Eeckman. A generalized hidden Markov model for the recognition of human genes in DNA. In *ISMB-96*, pages 134–142, St. Louis, June 1996. AAAI Press. http://www.cse.ucsc.edu/˜dkulp/cgi-bin/genie.

[104] P. Lackner, W. Koppensteiner, F. Domingues, and M. Sippl. Automated large scale evaluation of protein structure predictions. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):7–14, 1999.

[105] R. Lathrop and T. Smith, 1995. Unpublished investigation in contact potential functions, performed in conjunction with Rick Lathrop and Temple Smith at Boston University.

[106] K. Laurio. Finding remote protein homologs with hidden markov models. Master's thesis, Högskolan Skövde, Computer Science, Högskolan Skövde, 54128 Skövde, Sweden, 1997.

[107] C. Lawrence. Bayesian Bioinformatics. In *ISMB97 Tutorial*, Halkidiki, Greece, June 1997.

[108] C. M. Lemer, M. J. Rooman, and S. J. Wodak. Protein structure prediction by threading methods: evaluation of current techniques. *Proteins: Structure, Function, and Genetics*, 23(3):227,55, 1995.

[109] Michael Levitt. Competetive Assessment of Protein Fold Recognition and Alignment Accuracy. *Proteins: Structure, Function, and Genetics*, Supplement 1(1):92–104, 1997.

[110] S. Lifson and C. Sander. Antiparallel and parallel beta strands differ in amino acid residue potential. *Nature*, 282(5734):109–11, 1979.

[111] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *PNAS*, 86(12):4412–5, 1989.

[112] V. N. Maiorov and G. M. Crippen. Learning about protein folding via potential functions. *Proteins: Structure, Function, and Genetics*, 20(2):167–73, October 1994.

[113] N. Manh and M. Cline. http://www.cse.ucsc.edu/research/compbio/HMM-apps/compare-align.html. World Wide Web, April 1998.

[114] A. Marchler-Bauer, 1999. Personal communication with Aron-Marchler-Bauer, postdoctoral scientist at NCBI.

[115] A. Marchler-Bauer and S. Bryant. Measures of threading specificity and accuracy. *Proteins: Structure, Function, and Genetics*, Supplement 1(1):134–139, 1997.

[116] A Marchler-Bauer, S. Bryant, and C. Hogue. http://www.ncbi.nlm.nih.gov/Structure/casp2/index.html. World Wide Web, June 1998.

[117] A. Marchler-Bauer, M. Levitt, and S. Bryant. A retrospective analysis of CASP2 threading predictions. *Proteins: Structure, Function, and Genetics*, Supplement 1(1):83–91, 1997.

[118] Marcella McClure, Chris Smith, and Pete Elton. Parameterization studies for the SAM and HMMER methods of hidden Markov model generation. In *ISMB-96*, pages 155–164, St. Louis, June 1996. AAAI Press.

[119] Marcella A. McClure, Tatha K. Vasi, and Walter M. Fitch. Comparative Analysis of Multiple Protein Sequence Alignment Methods. *Mol. Biol. Evol.*, 11(4):571–592, 1994.

[120] Heinz Th. Mevissen and Martin Vingron. Quantifying the Local Reliability of a Sequence Alignment. *Protein Engr.*, 9(2):127–32, 1996.

[121] D. L. Minor Jr and P. S. Kim. Context is a major determinant of beta sheet propensity. *Nature*, 371(6494):264–7, 1994.

[122] L. A. Mirny and E. I. Shakhnovich. Protein structure prediction by threading. Why it works and why it does not. *JMB*, 283(2):507–26, 1998.

[123] S. Miyazawa. A reliable sequence alignment method based on probabilities of residue correspndences. *Protein Engr.*, 8(10):999–1009, 1994.

[124] S. Miyazawa and R. Jernigan. Residue-residue potentials with a favorable contact pair term and unfavorable high packing density term, for simulation and threading. *JMB*, 256(3):623–44, 1996.

[125] B. Morgenstern, K. French, A. Dress, and T. Werner. DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics*, 14(3):290–4, 1998.

[126] J. Moult. Comparison of database potentials and molecular mechanics. *Curr. Opin. Struct. Biol.*, 7(2):194–9, 1997.

[127] J. Moult, T. Hubbard, K. Fidelis, and J. Pedersen. Critical assessment of methods of protein structure prediction (CASP): round II. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):2–6, 1999.

[128] Alexey G. Murzin. Structure Classification-based Assessment of CASP3 Predictions for the Fold Recognition Targets. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):88–103, 1999.

[129] D. Naor and D. Brutlag. On near-optimal alignments of biological sequences. *Jour. Comp. Biol.*, 1(4):349–66, 1994.

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

[131] C. L. Nelsoney and J. W. Kelly. Progress towards understanding β-sheet structure. *Bioorganic and Medicinal Chemistry*, 4(6):739–66, 1996.

[132] A. Neuwald, J. Liu, D. Lipman, and C. E. Lawrence. Extracting protein alignment models from the sequence database. *NAR*, 25:1665–1677, 1997.

[133] C. Notredame, L. Holm, and D. Higgins. COFFEE: an objective function for multiple sequence alignments. *bioinformatics*, 14(5):407–22, June 1998.

[134] Lichtarge O, Bourne H.R., and Cohen F.E. An evolutionary trace method defines binding surfaces common to protein families. *JMB*, 257:342–358, Mar 1996.

[135] C. Orengo and W. Taylor. SSAP: Sequential structure alignment program for protein structure comparison. *Methods Enzymol.*, 266:617–35, 1996.

[136] 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(8):1093–108, August 1997.

[137] C.A. Orengo, J.E. Bray, T. Hubbard, L. LoConte, and I. Sillitoe. Analysis and Assessment of Ab Initio Three-Dimensional Prediction, Secondary Structure, and Contacts Prediction. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):149–170, 1999.

[138] D. E. Otzen and A. R. Fersht. Side-chain determinants of beta-sheet stability. *Biochemistry*, 34(17):5718–24, 1995.

[139] C. N. Pace, B. A. Shirley, M. McNutt, and K. Gajiwala. Forces contributing to the conformational stability of proteins. *Faseb Journal*, 10(1):75–83, 1996.

[140] A. Panchenko, A. Marcher-Bauer, and S. Bryant. Threading with explicit models for evolutionary conservation of structure and sequence. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):133–40, 1999.

[141] B. H. Park, E. S. Huang, and M. Levitt. Factors affecting the ability of energy functions to discriminate correct from incorrect folds. *JMB*, 266(4):831–46, 1997.

[142] J. Park, 1997. Personal communication with Jong Park, postdoctoral researcher at the MRC laboratory of Molecular Biology.

[143] J. Park, A. Bateman, and T. Hubbard. http://www.mrc-lmb.cam.ac.uk/genomes/jong/evalign_paper.html. World Wide Web.

[144] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and C. Chothia. Sequence Comparisons Using Multiple Sequences Detect Three Times as Many Remote Homologues As Pairwise Methods. *JMB*, 284(4):1201–1210, 1998. Paper available at `http://www.mrc-lmb.cam.ac.uk/genomes/jong/assess_paper/assess_paperNov.html`.

[145] J. Park, S. Teichmann, T. Hubbard, and C. Chothia. Intermediate Sequences Increase the Detection of Homology Between Sequences. *JMB*, 273:349–354, 1997.

[146] J. H. Park and T. J. P. Hubbard.
http://www.mrc-lmb.cam.ac.uk/genomes/jong/SC_rate_paper.html. World Wide Web.

[147] S. Pascarella and P. Argos. A data bank merging related protein structures and sequences. *Protein Engr.*, 5(2):121–37, 1992.

[148] W. Pearson and D. Lipman. Improved tools for biological sequence comparison. *PNAS*, 85:2444–2448, 1988.

[149] Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE*, 77(2):257–286, February 1989.

[150] M. D. Richard and R. P Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Computation*, 3(4):461–83, 1991.

[151] B. Rost. TOPITS: threading one-dimensional predictions into three-dimensional structures. In *ISMB-95*, pages 314–21. AAAI, 1995.

[152] B. Rost. PHD: predicting one-dimensional protein structure by profile-based neural networks. *Methods Enzymol.*, 2666:525–39, 1996.

[153] B. Rost. Protein fold recognition by merging 1D structure prediction and sequence alignments. World Wide Web, 1996.
http://www.embl-heidelberg.de/~rost/Var/aqua/aqua.html.

[154] Burkhard Rost. Twilight zone of protein sequence alignments. *Protein Engr.*, 12(2):85–94, February 1999.

[155] R. B. Russell and G. J. Barton. Structural features can be unconserved in proteins with similar folds. An analysis of side-chain to side-chain contacts secondary structure and accessibility. *JMB*, 244(3):332–50, 1994.

[156] R. B. Russell, M. A. Saqi, R. A. Sayle, P. A. Bates, and M. J. Sternberg. Recognition of analogous and homologous protein folds: analysis of sequence and structure conservation. *JMB*, 269(3):423–39, 1997.

[157] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4(4):406–25, 1987.

[158] C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins*, 9(1):56–68, 1991.

[159] S. M. Sauder, J. W. Arthur, and R. L. Dunbrack Jr. Large-scale comparison of protein sequence alignment algorithms with structural alignments. *Proteins: Structure, Function, and Genetics*, To appear.

[160] Reinhard Schneider and Chris Sander. The HSSP database of protein structure-sequence alignments. *NAR*, 24(1):201–205, 1 Jan 1996.

[161] I. N. Shindyalov and P. E. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engr.*, 11(9):739–47, 1998.

[162] M. J. Sippl. Calculation of conformational ensembles from potentials of mean force. An approach to the knowledge-based prediction of local structures in globular proteins. *Journal of Molecular Biology*, 213(4):859–83, June 1990.

[163] Manfred J. Sippl. Boltzmann's Principle, Knowledge-based Mean Fields and Protein Folding: an Approach to the Computational Determination of Protein Structures. *Journal of Computer-Aided Molecular Design*, 7(4):473–501, August 1993.

[164] T. Sivaraman, T. K. Kumar, Y. T. Tu, W. Wang, H. M. Chen, and C. Yu. Secondary structure formation is the earliest structural event in the refolding of an all beta-sheet protein. *Biochemical and Biophysical Research Communications*, 260(1):284–8, 1999.

[165] K. Sjolander. Automated domain identification in proteins using HMMs. Presented at the TIGR Genome Sequencing and Analysis Conference (GSAC), September 1999.

[166] K. Sjölander, K. Karplus, M. P. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet Mixtures: A Method for Improving Detection of Weak but Significant Protein Sequence Homology. *CABIOS*, 12(4):327–345, August 1996.

[167] Kimmen Sjölander. Phylogenetic inference in protein superfamilies: analysis of SH2 domains. In *Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology*, June 1998.

[168] C. K. Smith, J. M Withka, and L. Regan. A thermodynamic scale for the beta-sheet forming tendencies of the amino acids. *Biochemistry*, 33(18):5510–17, 1994.

[169] R. F. Smith and T. F. Smith. Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modelling. *Protein Engr.*, 5(1):35–41, 1992.

[170] T. F. Smith. The art of matchmaking: sequence alignment methods and their structural implications. *Structure with Folding and Design*, 7(1):"R7–R12", 1999.

[171] T. F. Smith and M. S. Waterman. Comparison of bio-sequences. *Adv. Appl. Math*, 2:482–489, 1981.

[172] E.L.L Sonnhammer, S.R. Eddy, and R. Durbin. Pfam: A Comprehensive Database of Protein Families Based on Seed Alignments. *Proteins*, 28:405–420, 1997.

[173] N. Srinivasan, K. Guruprasad, and T. Blundell. Comparative modelling of proteins. In M. J. Sternberg, editor, *Protein Structure Prediction*, chapter 1, pages 1–30. IRL Press, 1996.

[174] M. J. E. Sternberg. Protein structure prediction — principles and approaches. In M. J. Sternberg, editor, *Protein Structure Prediction*, chapter 1, pages 1–30. IRL Press, 1996.

[175] S. R. Sunyaev, F. Eisenhaber, P. Argos, E. N. Kuznetsov, and V. G. Tumanyan. Are knowledge-based potentials derived from protein structure sets discriminative with respect to amino acid types? *Proteins: Structure, Function, and Genetics*, 31(3):225–46, 1998.

[176] Roman L. Tatusov, Stephen F. Altschul, and Eugen V. Koonin. Detection of Conserved Segments in Proteins: Iterative Scanning of Sequence Databases with Alignment Blocks. *PNAS*, 91:12091–12095, December 1994.

[177] W. R. Taylor. Multiple sequence threading: an analysis of alignment quality and stability. *JMB*, 269(5):902–43, 1997.

[178] W. R. Taylor. Dynamic sequence databank searching with templates and multiple alignment. *JMB*, 280(3):375–406, 1998.

[179] P. D. Thomas and K. A. Dill. Statistical potentials extracted from protein structures: how accurate are they? *JMB*, 257(2):457–69, March 1996.

[180] Julie D. Thompson, Desmond G. Higgins, and Toby J. Gibson. CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-specific Gap Penalties, and Weight Matrix Choice. *NAR*, 22(22):4673–4680, 1994.

[181] Julie D. Thompson, Frederick Plewniak, and Oliver Poch. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1):87–8, 1999.

[182] Julie D. Thompson, Frederick Plewniak, and Oliver Poch. A comprehensive comparison of multiple sequence alignment programs. *NAR*, 27(13):2682–90, 1999. Additional detailed results at `http://www-igbmc.u-strasbg.fr/BioInfo/BAliBASE/prog_scores.html`.

[183] J. Thorne, H. Kishino, and J. Felsenstein. Inching toward reality: an improved likelihood model of sequence evolution. *Journal of Molecular Evolution*, 34(1):3–16, January 1992.

[184] A. Torda. Perspectives in protein-fold recognition. *Curr. Opin. Struct. Biol.*, 7(2):200–5, 1997.

[185] E. C. Uberbacher and R. J. Mural. Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. *PNAS*, 88:11261–11265, 1991.

[186] S. Vajda, M. Sippl, and J. Novotny. Empirical potentials and functions for protein folding and binding. *Curr. Opin. Struct. Biol.*, 7(2):222–8, 1997.

[187] M. Vendruscolo and E. Domany. Pairwise contact potentials are unsuitable for protein folding. *Journal of Chemical Physics*, 109(24):11101–8, 1998.

[188] M. Vingron. Near-optimal sequence alignment. *Curr. Opin. Struct. Biol.*, 6(3):346–52, 1996.

[189] M. Vingron and P. Argos. Determination of reliable regions in protein sequence alignments. *Protein Engr.*, 3(7):565–9, 1990.

[190] T. Webster, R. Nambudripad, and L. Buturovic, 1997. Personal communication with members of the Temple Smith laboratory, Boston University.

[191] J. V. White, I. Muchnik, and T. F. Smith. Modeling protein cores with Markov random fields. *Mathematical Biosciences*, 124(4):149–79, December 1994.

[192] D. H. Wolpert and D. R. Wolf. Estimating functions of probability distributions from a finite set of samples. *Physical Review E*, 52(6):6841–54, December 1995.

[193] An-Suei Yang and Barry Honig. Free energy determinants of secondary structure formation: II. antiparallel β-sheets. *JMB*, 252(3):366–76, 1995.

[194] L. Yu and T. Smith. Positional statistical significance in sequence alignment. *Jour. Comp. Biol.*, 6(2):253–259, 1999.

[195] L. Zhang and J. Skolnick. How do potentials derived from structural databases relate to ”true” potentials? *Protein Sci.*, 7(1):112–22, 1998.

[196] Z. Zhang, P. Berman, T. Wiehe, and W. Miller. Post-processing long pairwise alignments. *Bioinformatics*, 15(12):1012–9, 1999.

[197] H. Zhu and W. Braun. Sequence specificity, statistical potentials, and three-dimensional structure prediction with selccorrecting distance geometry calculations of beta-sheet formation in proteins. *Protein Sci.*, 8(2):326–42, 1999.

[198] M. Q. Zjang and T. G. Marr. Alignment of molecular sequences seen as random path analysis. *Journal of Theoretical Biology*, 174(2):119–29, May 1995.

[199] F. Zu-Keng and M. Sippl. Optimum superposition of protein structures: ambiguities and implications. *Folding and Design*, 1(2):123–32, 1996.

[200] M. Zuker. Suboptimal sequence alignment in molecular biology. Alignment with error analysis. *JMB*, 221(2):403–20, September 1991.