Structural Modelling of Transmembrane Domains

Sebastian Kelm

Department of Statistics

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy (DPhil)

April 2011

Declaration

All material presented in this thesis is my own work, except where otherwise stated.

Signed: *Date:*
Sebastian Kelm

....

For my parents.

Acknowledgements

First of all, I would like to thank all the members of the Oxford Protein Informatics Group, especially K1, for making life at the office more than just work. Being a part of OPIG has been a lot of fun. I also thank my office mates Habib Saadi, Waqar Ali and Jamie Hill for the daily mayhem that kept life at the office interesting.

Many people helped make this thesis happen. First and foremost my supervisor, Dr Charlotte Deane, whose grim determination has brought about the publication of every single chapter of this thesis, and my industrial supervisor, Dr Jiye Shi (UCB Celltech, Slough), who provided direction and feedback at every stage.

Much of my work was done in collaboration or consultation with others. My program iMembrane is built on the work of Prof Mark Sansom's research group (Biochemistry, Oxford). He and Dr Kathryn Scott have provided much valuable feedback during the early stages of my DPhil, for which I am grateful. My work on Environment-Specific Substitution Tables was published in collaboration with my colleague Jamie Hill. From my 2007 beginnings to Jamie's 2010 summer project and our collaborative 2011 paper, it has grown into something publishable. During my work on protein loop modelling I have repeatedly benefited from the invaluable advice of local loop guru, Yoonjoo Choi. Web wizard Jean-Paul Ebeyer conjured up the web interface for our FREAD loop modelling server, which should be published before the end of the year. These fine people have provided context to my science and have made me feel like I am part of something bigger.

Last, but certainly not least, I thank my family for caring: My grand parents, who kept asking for news about "Studiosus", my little big sister and

of course my parents, without whom I would not be where I am today. Although the journey of the past 8 years has been my own, I might not have made it without them.

There are others who deserve my thanks but, for the sake of brevity, I will not name them all. Know that I haven't forgotten you and if you think I have, feel free to remind me. Cheers, boys and girls.

Now enjoy the read.

Abstract

Membrane proteins represent about one third of all known vertebrate proteins and over half of the current drug targets. Knowledge of their three-dimensional (3D) structure is worth millions of pounds to the pharmaceutical industry. Yet experimental structure elucidation of membrane proteins is a slow and expensive process. In the absence of experimental data, computational modelling tools can be used to close the gap between the numbers of known protein sequences and structures. However, currently available structure prediction tools were developed with globular soluble proteins in mind and perform poorly on membrane proteins. This thesis describes the development of a modelling approach able to predict accurately the structure of transmembrane domains of proteins.

In this thesis we build a template-based modelling framework especially for membrane proteins, which uses membrane protein-specific information to inform the modelling process. Firstly, we develop a tool to accurately determine a given membrane protein structure's orientation within the membrane. We offer an analysis of the preferred substitution patterns within the membrane, as opposed to non-membrane environments, and how these differences influence the structures observed. This information is then used to build a set of tools that produce better sequence alignments of membrane proteins, compared to previously available methods, as well as more accurate predictions of their 3D structures. Each chapter describes one new piece of software or information and uses the tools and knowledge described in previous chapters to build up to a complete accurate model of a transmembrane domain.

Contents

List of Figures

LIST OF FIGURES

LIST OF FIGURES

Chapter 1

Introduction

1.1 The Importance of Protein Structure

1.1.1 Proteins

Proteins are one of the key classes of biological macromolecules. They constitute the largest part of a biological cell's dry mass. Proteins serve as structural building blocks as well as performing most of the cell's molecular functions (Alberts et al., 2002).

1.1.2 Membrane Proteins

Membrane proteins represent around 30% of all known proteins (von Heijne, 2007). They control the communication and exchange of chemicals between the inside and outside of every living cell (see Figure 1.1). Many medically relevant processes involve membrane proteins, e.g. the recognition of foreign substances by the immune system, adhesion of blood cells to the blood vessel walls, electron transport during cellular respiration and the recycling of used neurotransmitters (Müller et al., 2008). It is thus not surprising that the majority of current drug targets are membrane proteins (von Heijne, 2007).

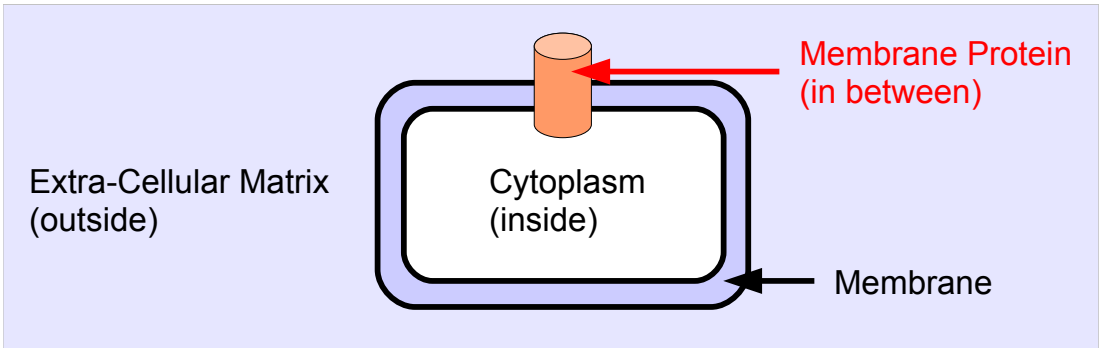


Figure 1.1: The importance of membrane proteins - Membrane proteins are essential for life. They form the bridge between the inside and outside of every biological cell.

1.1.3 Protein Structure

It has been widely accepted since the early '70s that in most cases a protein's sequence directly determines its three-dimensional (3D) structure (Anfinsen, 1973), which in turn determines its function (Alberts et al., 2002), see Figure 1.2. A protein's function can also be investigated without knowledge of its structure (e.g. using binding assays and enzyme kinetics assays). However, knowledge of a protein's structure allows a fuller exploration of how a specific function is obtained.

1.1.3.1 The four levels of protein structure

Protein structure can be broken down into four levels:

1. *Primary structure* refers to the network of covalent bonds between amino acids that holds a protein together. Usually, this is a simple string of peptide bonds, from the protein's N-terminus through to the C-terminus. The primary structure can be more complicated than a simple string if covalent bonds are present between amino acid side chains. In natural proteins this is only the case when disulfide bonds (between cysteine side chains) are present. Disulfide bonds never occur in the reducing environment of the cytosol, or within transmembrane domains, but can occur in secreted or lysosomal proteins or in the extracellular

domains of membrane proteins.

2. *Secondary structure* refers to the local 3D structure taken up by peptide chains. Secondary structure types – helices, strands and coils – are defined by their local hydrogen bonding patterns. *Helices* involve periodic hydrogen bonds between the ith residue and the $i+3$ residue (3_{10} helices), the $i+4$ residue (α-helices) or the $i+5$ residue (π-helices). *β-strands* have an extended conformation and do not form local hydrogen bonds. Instead, they interact with other strands further along in the sequence, to form larger structures, the *β-sheets*. All amino acid sub-sequences not fitting into these two categories are commonly referred to as *random coils* or *loops* and are thought of as linkers between secondary structure elements. Very short loops (3 or 4 residues) describing a 180° turn are sometimes referred to as *turns*, e.g. the β-turns between anti-parallel β-strands.

3. *Tertiary structure* is the global 3D shape taken up by a single protein chain. Secondary structure elements (helices and strands) interact with each other to form compact structures, called "domains". Types of non-covalent interactions observed are hydrogen bonds (polar interactions, electron sharing), salt bridges (charge-charge interactions) and van der Waals forces (hydrophobic interactions).

4. *Quarternary structure* is formed when several protein chains assemble into a larger multimeric complex. Multiple identical chains can form homomers, or can associate with other proteins to form heteromers. Assembly into a complex allows more complicated functions to be carried out, e.g. the transcription of DNA to RNA by the RNA polymerase complex.

1.1.3.2 Accessible and buried residues

The residues on the protein's surface can interact with other proteins and ligands and so may be functionally important. On the other hand, their mutation may not affect the

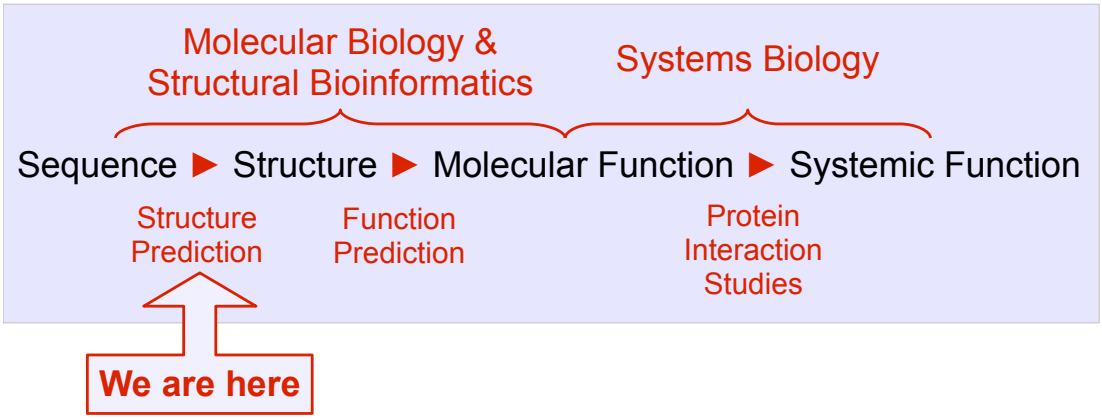


Figure 1.2: The context of protein structure prediction in understanding biology - Structure prediction is an essential step in moving from the sequence of a gene to the understanding of the gene product's (i.e. the corresponding protein's) role in the organism.

stability of the protein's structure. Residues buried in the centre of an approximately spherical protein interact tightly with the surrounding residues to form the compact "core" of the protein. A mutation of their side chains to a chemically or sterically different residue might prevent the protein from folding properly.

1.1.3.3 The structure of soluble proteins

Soluble proteins often adopt a globular conformation (Figure 1.3). In their natural environment they are completely surrounded by water; their structure is usually optimised to have hydrophobic residues in the protein core, while the water-exposed surface contains a higher concentration of polar or charged residues (Berg et al., 2002). The "hydrophobic effect", i.e. the burying of hydrophobic residues inside the protein core, is thought to be a major driving force in the folding of soluble proteins (Berg et al., 2002).

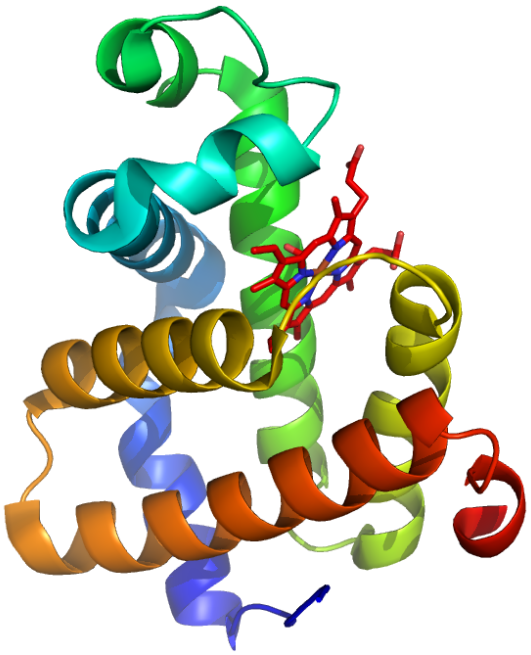


Figure 1.3: A typical globular protein structure - Cartoon representation of a globular water-soluble protein structure, in this case haemoglobin (PDB: 3A9M).

1.1.3.4 The structure of membrane proteins

In comparison, membrane proteins are not completely surrounded by water. Instead, they penetrate a lipid bilayer (a membrane) and thus contain stretches of residues that are exposed to the hydrophobic environment at the core of the membrane. Transmembrane segments usually have one of two structure types: helices or β-strands (Figure 1.4). Helical transmembrane proteins are translocated into the membrane while being translated by the ribosome (Elofsson and von Heijne, 2007). It is thus assumed that they adopt their main fold co-translationally, although rearrangements and multimerisations may take place after translation has finished (Bowie, 2005; DeGrado et al., 2003; Kauko et al., 2010) (see Figure 1.5).

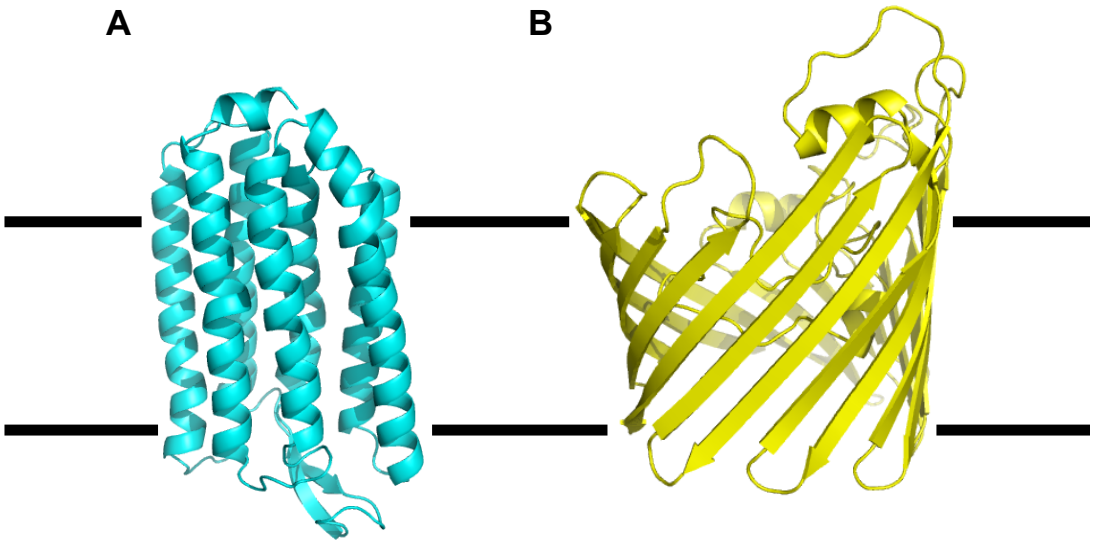


Figure 1.4: Transmembrane protein structure - Cartoon representation of two typical transmembrane (TM) proteins. The black lines show the approximate location of the lipid bilayer. (A) halorhodopsin (PDB: 2JAF), a polytopic helical TM protein; (B) porin (PDB: 2POR), a TM β-barrel constituted of anti-parallel β-strands.

1.1.4 The Concept of Homology

Two proteins are said to be "homologous" if they have a common evolutionary ancestor. Homologous proteins are found to have the same or very similar molecular functions and structures.

1.1.4.1 Alignment

The most common way to quantify how similar two protein sequences are to each other is by "aligning" their sequences. Each protein sequence is represented by a string of letters, each of which encodes a particular type of amino acid (e.g. 'A' for Alanine). If each sequence is one row in a matrix, the goal of sequence alignment is to shift the sequences left or right such that the maximum number of matrix columns contain two similar amino acids. In addition, gaps can be inserted in the middle of a sequence, e.g. if the two sequences are of different length, or if they are only locally similar. Two

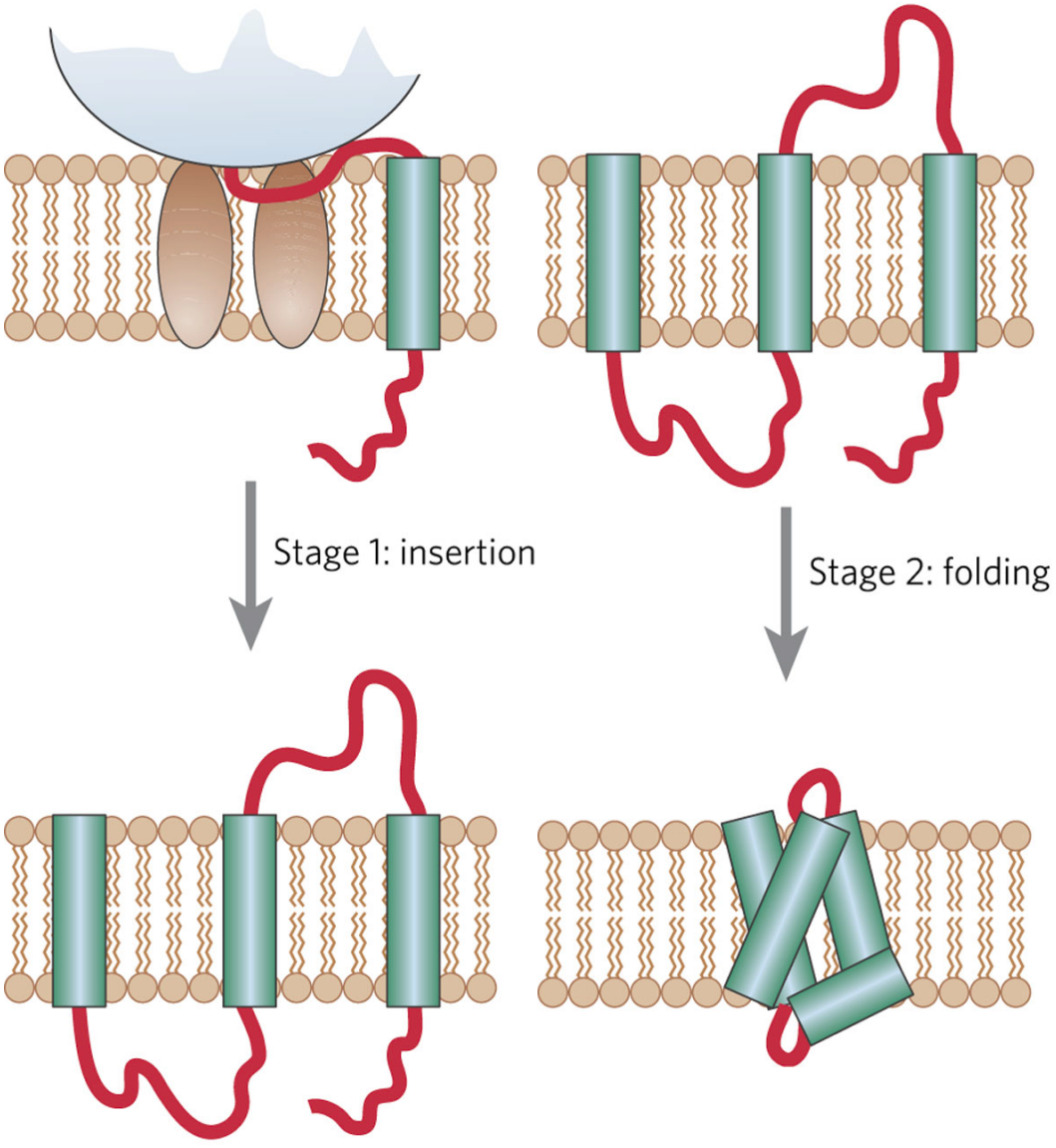


Figure 1.5: Transmembrane protein folding - Diagrammatic representation of the membrane protein folding pathway. Red string: protein flexible chain; green cylinders: TM helices. Left: protein synthesis by the ribosome (white and grey ball) and insertion into the membrane via the translocon complex (brown ellipsoids). Right: folding into the final TM protein structure. Figure adapted from (Bowie, 2005). Reprinted by permission from Macmillan Publishers Ltd: Nature 438, 581-589 (1 December 2005) — doi:10.1038/nature04395, copyright 2005.

components are required for sequence alignment: (a) a measure of how similar each type of amino acid is to the others (i.e. how similar are Alanine and Threonine, compared to Alanine and Histidine – see below); (b) a scoring system to control the location and size of gaps. Once the sequences have been aligned, the similarity between the two sequences can be computed. The most simplistic and most widely used similarity measure is "percentage sequence identity", i.e. the number of columns containing two identical amino acids divided by the total number of columns. An example sequence alignment is shown in Figure 1.6A. Sequence alignment can be generalised to multiple sequence alignment or can involve structural information, e.g. in sequence-to-structure alignment. When the structure of both proteins is known, a pure structure alignment can be performed. More details about the various alignment methods are given in Section 1.4.2.

1.1.4.2 Substitution tables

Substitution tables are typically 20-by-20 matrices, each row and column representing one of the 20 amino acid types. Each value in such a table represents the likelihood (usually as a log-odds score) of observing the substitution from one type of amino acid to another (e.g. from Alanine to Threonine) in a pair of homologous proteins.

The first ever log-odds substitution scoring matrix was PAM (Dayhoff and Schwartz, 1978), which stands for "point accepted mutation". The first PAM matrix, PAM1, was created from the amino acid substitutions observed in families of highly similar protein sequences (1 substitution every 100 amino acids). Higher order matrices represent more dissimilar proteins and are obtained by repeated matrix multiplications of the PAM1 matrix with itself. PAM250 is thus obtained after 250 matrix multiplications and represents the mutation probabilities observed in proteins that have diverged by 250 mutation events within each stretch of 100 amino acids.

The most commonly used type of log-odds matrix is BLOSUM (Henikoff and

A

Name	Sequence
2sli_A	QVGL**L**YEKYDSWSRNELH-LKDILKFEKYSISELTGQA-
gi|109070532	QAPQ**L**YVLYEKGRNHYTE-SISMAKISVYGTL-------
gi|109101506	LFGC**L**YEANDYEEIVFLM-FTLKQAFPAEYLPQ------
gi|170712512	TF--**V**LTSYGYWEKDYNKPYIKSLRVTLKEIDEIVREMV
gi|110674221	NIGL**L**YEGTPSEEMSYIE---MNLKYLESGANK------

B

	G	A	V	L	M	I	F	Y	W	S	T	C	P	N	Q	K	R	H	D	E
G	6	0	-3	-4	-3	-4	-3	-3	-2	0	-2	-3	-2	0	-2	-2	-2	-2	-1	-2
A	0	4	0	-1	-1	-1	-2	-2	-3	1	0	0	-1	-2	-1	-1	-1	-2	-2	-1
V	-3	0	4	1	1	3	-1	-1	-3	-2	0	-1	-2	-3	-2	-2	-3	-3	-3	-2
L	-4	-1	1	4	2	2	0	-1	-2	-2	-1	-1	-3	-3	-2	-2	-2	-3	-4	-3
M	-3	-1	1	2	5	1	0	-1	-1	-1	-1	-1	-2	-2	0	-1	-1	-2	-3	-2
I	-4	-1	3	2	1	4	0	-1	-3	-2	-1	-1	-3	-3	-3	-3	-3	-3	-3	-3
F	-3	-2	-1	0	0	0	6	3	1	-2	-2	-2	-4	-3	-3	-3	-3	-1	-3	-3
Y	-3	-2	-1	-1	-1	-1	3	7	2	-2	-2	-2	-3	-2	-1	-2	-2	2	-3	-2
W	-2	-3	-3	-2	-1	-3	1	2	11	-3	-2	-2	-4	-4	-2	-3	-3	-2	-4	-3
S	0	1	-2	-2	-1	-2	-2	-2	-3	4	1	-1	-1	1	0	0	-1	-1	0	0
T	-2	0	0	-1	-1	-1	-2	-2	-2	1	5	-1	-1	0	-1	-1	-1	-2	-1	-1
C	-3	0	-1	-1	-1	-1	-2	-2	-2	-1	-1	9	-3	-3	-3	-3	-3	-3	-3	-4
P	-2	-1	-2	-3	-2	-3	-4	-3	-4	-1	-1	-3	7	-2	-1	-1	-2	-2	-1	-1
N	0	-2	-3	-3	-2	-3	-3	-2	-4	1	0	-3	-2	6	0	0	0	1	1	0
Q	-2	-1	-2	-2	0	-3	-3	-1	-2	0	-1	-3	-1	0	5	1	1	0	0	2
K	-2	-1	-2	-2	-1	-3	-3	-2	-3	0	-1	-3	-1	0	1	5	2	-1	-1	1
R	-2	-1	-3	-2	-1	-3	-3	-2	-3	-1	-1	-3	-2	0	1	2	5	0	-2	0
H	-2	-2	-3	-3	-2	-3	-1	2	-2	-1	-2	-3	-2	1	0	-1	0	8	-1	0
D	-1	-2	-3	-4	-3	-3	-3	-3	-4	0	-1	-3	-1	1	0	-1	-2	-1	6	2
E	-2	-1	-2	-3	-2	-3	-3	-2	-3	0	-1	-4	-1	0	2	1	0	0	2	5

Figure 1.6: Sequence Alignment - (A) An example sequence alignment of globular proteins (the last 39 residues are shown). One conserved Leucine residue is highlighted in red. One of the sequences shows a Valine (which is similar in size and hydrophobicity) at the same position. (B) The BLOSUM62 substitution matrix. The substitutions from Leucine to Leucine (red) and from Leucine to Valine (yellow) are highlighted. Scores are log-odds scores (computed from substitution frequencies), where higher numbers indicate more favourable substitutions. Substitutions to self (e.g. L to L) indicate how conserved a residue tends to be. Note that BLOSUM matrices are symmetric.

Henikoff, 1992), which stands for "blocks substitution matrix". Matrices of this type are built from the frequencies of amino acid substitutions observed in conserved blocks of sequences in local multiple alignments of closely related proteins. The most popular version is BLOSUM62 (shown in Figure 1.6B). The number in the name signifies the redundancy level of the sequences used to build the table. For BLOSUM62, proteins with no more than 62% sequence identity were used. This is done in order to avoid biasing the score matrix towards common sequences.

BLOSUM-like matrices have been constructed specifically for membrane proteins: JJT (Jones et al., 1994), PHAT (Ng et al., 2000) and SLIM (Müller et al., 2001). These matrices mainly differ from BLOSUM matrices in that they were constructed from datasets containing only the transmembrane domains of membrane proteins.

A further refinement of substitution matrices are "environment-specific substitution tables" (ESST) (Shi et al., 2001). If the structure of a protein is known, then the protein can be partitioned into distinct "structural environments", based on environmental factors. For example, if we consider only the environmental factors "accessibility" and "secondary structure", then one particular structural environment might be defined as "residues on the protein surface and in an alpha helix". Given enough proteins of known structure, one can generate substitution tables for each distinct structural environment. These can be used in sequence alignment and allows for more accurate alignments. The main drawback is that, to use such tables, at least one of the proteins to be aligned must be of known structure.

1.1.4.3 Databases of protein domain families

A protein domain is thought to be the evolutionary and functional unit of a protein (Alberts et al., 2002). Domains are thought to adopt a stable structure on their own and can form the entirety of a protein or appear as part of a multi-domain protein, covalently connected to further domains by (possibly flexible) "linker" regions. There

are databases grouping protein domains into families based on their known or inferred evolutionary distance. Databases such as PFAM (Bateman et al., 2004) are based purely on sequence information. Others, like SCOP (Murzin et al., 1995) and CATH (Orengo et al., 1997) contain domains from proteins of known structure. Both SCOP and CATH are organised hierarchically, with the top level being based solely on secondary structure (e.g. "all alpha" or "all beta") and the bottom level representing domain families with almost identical structure and a high level of sequence similarity. While CATH classification occurs semi-automatically based on sequence and structure similarity, SCOP classification occurs in a more manual fashion.

1.1.5 Drug Discovery

Drugs attempt to modify a target protein's function. Drug discovery today is primarily driven by high throughput techniques. Large libraries of millions of compounds are screened in the hope of identifying a candidate inhibitor for the target protein (Hertzberg and Pope, 2000). These compounds can be small organic molecules, or short peptides. Peptide ligands can be further evolved and refined using techniques such as phage display (Azzazy and Highsmith, 2002; Yang and Nolan, 2007). More recently, antibodies – large organic molecules that form part of our natural immune system – have become a major focus in drug development (Carter, 2006), for example as anti-cancer drugs (Schrama et al., 2006). Currently, the process from finding a candidate inhibitor to mass-producing the final drug typically takes 10 to 15 years, costs vast sums of money and has a very low success rate.

Knowledge of the target protein's structure can greatly aid in the long process of drug discovery. Firstly, this knowledge enables rational drug design and refinement. A candidate inhibitor can be improved by changing its structure to better fit into the binding cleft of the target protein (Carter, 2006; Heckmann et al., 2008; Ramos and Fernandes, 2006). Secondly, knowing the target's structure allows the application

of computational techniques to predict interactions with potential inhibitors or other proteins or small molecules (Amini et al., 2007). Even low resolution structures can still be useful in this process (Vakser, 1995).

With the development of fast and accurate computational methods, the entire drug production process could become faster and cheaper, while the side effects of drugs could be reduced. Therefore, much money and effort is being invested into developing computational methods to predict protein structure and protein interactions.

In this project, we focus on the production of computational methods for protein structure prediction, especially for the large class of drug targets, membrane proteins.

1.2 Experimental Protein Structure Elucidation

X-ray crystallography provides the highest resolution protein structure models available to date. However, the method requires the purification and crystallisation of the target protein, which is difficult, time consuming and expensive. Even with constant developments of experimental methods, the gap between the number of known sequences (over 11 million entries in the RefSeq database (Sayers et al., 2009) on 1 Jan 2011) and known structures (over 70000 entries in the PDB (Berman et al., 2000) on 1 January 2011) continues to grow exponentially. This problem is even more obvious in the case of membrane proteins, which are considerably harder to purify and crystallise, due to their physico-chemical properties (Müller et al., 2008). Only about 1500 membrane protein structures are currently listed in the PDB_TM database (Tusnády et al., 2005), and most of these are closely related.

Crystallisation of membrane proteins usually requires the modification of the protein, or co-crystallisation together with antibody (F_{ab}) fragments or detergents. The formation of a crystal lattice is achieved by covering hydrophobic patches and providing a framework for polar and charged interactions. This procedure is lengthy and adds to

the difficulty of elucidating membrane protein structures (Müller et al., 2008).

The second most popular experimental method for protein structure elucidation is nuclear magnetic resonance (NMR) spectroscopy (Wuthrich, 1989). About 1/8 of the structures in the PDB (16 April 2011) were obtained via this method. NMR requires the purification of the target protein in milligram quantities and, in the case of a typical solution NMR experiment, its subsequent suspension in liquid. For membrane proteins (Marassi and Opella, 1998), this means having to remove the proteins from their natural hydrophobic membrane environment, which is difficult and causes the proteins to denature (lose shape). Membrane proteins need to be solubilised using detergents, which form micelles around the hydrophobic regions of the protein, or using organic solvents. Producing the correct detergent conditions that cause the membrane protein to refold is problematic. Even if the protein refolds into a stable structure, it is hard to tell if this structure is identical to the protein's native structure. In addition, traditional solution NMR experiments have a limit on the size of the molecule being studied. This boundary has been pushed up to about 50kD, thus enabling the elucidation of proteins of up to about 500 amino acids. However, many membrane proteins natively form large multi-chain complexes that can contain thousands of amino acids.

A more recent approach is solid state NMR (Fu and Cross, 1999; Renault et al., 2010). This allows the elucidation of solid objects that move slowly or not at all. This technique allows the membrane proteins to sit inside larger bilayer vesicles, which more closely resemble their native environment. However, solid state NMR is more technically challenging than solution NMR and is still in its early phases.

1.3 Computational Protein Structure Prediction

Protein structure prediction aims to bridge the gap between the numbers of known protein sequences and available 3D structures. The goal is to produce an accurate representation of a protein's 3D structure given, in the typical case, only its amino acid sequence. Broadly, structure prediction can be divided into two main categories: homology (or template-based) modelling and ab initio (or template-free) modelling (Moult et al., 2009).

Ab initio (i.e. "from first principles") modelling methods aim to simulate the protein folding process in silico. This is generally implemented by encoding the rules of chemistry and physics, combined with empirical knowledge about typical protein structures, in an energy function and then exploring a protein chain's conformational space while minimising its energy. Such methods have, so far, produced useful results only for small proteins (Moult et al., 2009), below about 150 residues in length. The larger the protein, the more degrees of freedom need to be explored. A typical membrane protein is over 300 amino acids long (up to about 1000), with approximately 10 atoms per amino acid. More recently, ab initio methods have begun to incorporate the principle of co-translational folding (Deane et al., 2007; Ellis et al., 2010), which tends to decrease the simulation time until biologically plausible structures are observed (Jefferys et al., 2010). While these approaches show interesting results in terms of their general folding behaviour, their practical application in structure prediction of large proteins is still in its infancy.

Currently, the most successful template-free method is ROSETTA (Das and Baker, 2008). Its algorithm is based on covering the entire length of the protein sequence with small overlapping fragments, representing all possible local 3D conformations. These fragments are combined in many different ways, to form candidate tertiary structures, which are then scored using a statistical energy function based on empirical knowledge

of protein structure. Candidate models are clustered in terms of overall similarity and the program returns one representative per cluster. ROSETTA has consistently performed well (Bradley et al., 2005; Das et al., 2007; Simons et al., 1999). Nevertheless, it cannot beat the accuracy of template-based methods, especially on molecules larger than about 150 residues, and requires large amounts of processing time.

Template-based methods, on the other hand, have been used to successfully predict protein structures for roughly 20 years (the most popular method in existence today, MODELLER, was developed in 1993) (Sali and Blundell, 1993). Homologues are usually identified by the similarity of their amino acid sequence, and it is assumed that their structure and function will therefore also be similar (Anfinsen, 1973). Homology modelling approaches thus begin with a search of the PDB (or any database of known structures) for a protein whose sequence is similar to a given target protein (of unknown structure). Using the known structure as a template, one assumes that the major elements (the "fold") of the structure are conserved between the two proteins, thus greatly reducing the search space to identify the correct overall target structure. The major drawback of homology methods is the complete reliance on a database of known structures. It is possible to model structures similar to those already known, but it is impossible to discover entirely new protein folds in this fashion.

Fifteen years ago, when most of the current homology methods were established, only a few dozen membrane protein structures were known (see PDB website) and their crystallisation was thought to be near-impossible. Thus, none of the traditional homology methods took membrane proteins into account and, for lack of a "correct" test set of membrane protein structures, neither did ab initio methods. Only in recent years have researchers begun to work on computational methods specific to membrane proteins (see Section 2.2).

With the constant advances in experimental technique and the growing interest in membrane proteins as drug targets we are, today, at a stage where homology methods

are becoming applicable to membrane proteins.

1.4 Methods for Template-Based Modelling

1.4.1 Overview

Template-based protein structure prediction is a multi-step process, involving the use of a variety of bioinformatics tools, typically joined together in an automated or semi-automated modelling pipeline. Especially for globular water-soluble proteins, a multitude of programs are available to perform each separate task (Eswar et al., 2007). The input to the entire modelling procedure is a single target sequence, whose structure is to be predicted.

1. **Template identification** (homology detection). In order to perform template-based structure prediction, template structures must first be obtained. This is done by scanning structural databases for homologous proteins of known structure. If proteins with high sequence identity to the target are available, this step can be done using fast sequence alignment methods (Altschul et al., 1990; Pearson, 1990). If the target sequence shares low sequence identity (below 40%) with proteins of known structure, finding homologues becomes increasingly hard. In the lower sequence identity range (10-40%) slower methods such as profile-to-sequence (Altschul et al., 1997; Eddy, 1998), profile-to-profile (Söding, 2005) or sequence-to-structure (Shi et al., 2001) alignment methods perform better. Once possible candidates have been identified, one or more template structures are chosen using scores (e.g. sequence identity, substitution scores, etc.) or previous knowledge.

2. **Alignment.** Many template finding methods are optimised to detect distant evolutionary homologues but do not yield a good quality alignment. Alignment

quality is one of the major contributors to the accuracy of the final model (Sánchez and Sali, 1997). An incorrect alignment will almost always result in an inaccurate model. Therefore, the templates may need to be re-aligned to each other and to the target sequence. Structure alignment methods (Holm and Sander, 1996; Ortiz et al., 2002; Shindyalov and Bourne, 1998; Zhang and Skolnick, 2005a), may be used to produce a multiple structure alignment, which is in turn aligned to the target sequence using sequence-to-structure alignment (a.k.a. "threading") programs (Shi et al., 2001).

3. **Template weighting.** When using multiple templates for modelling, templates may be weighted, e.g. using their estimated evolutionary proximity to the target sequence (e.g. SCORE (Deane et al., 2001)). In addition, clustering methods may be applied to reduce the bias towards groups of closely related protein structures.

4. **Co-ordinate generation.** The final step in modelling is the generation of three dimensional co-ordinates from the alignment between target and template proteins. Three main approaches exist for template-based co-ordinate generation: assembly of rigid bodies (Bates et al., 2001; Deane and Blundell, 2001; Koehl and Delarue, 1995; Petrey et al., 2003; Schwede et al., 2003; Sutcliffe et al., 1987), segment matching (Levitt, 1992) and satisfaction of spatial restraints (Sali and Blundell, 1993). All these approaches result in a 3D model, which may contain gaps (missing residues) and often only the backbone atoms (N, Cα, C, O) of the protein chain. Some methods model all atoms, but may sometimes generate low-quality side-chain co-ordinates, while other programs specialise in re-modelling only the side-chains on a given model (Krivov et al., 2009).

5. **Model quality assessment and model selection.** Usually, co-ordinate generation results in the creation of more than one model. The quality of the models can be assessed using objective functions, which may test for problems such as

steric clashes, unfavourable bond angles and distances, as well as checking the agreement with typical (empirical) local structures. These functions generally return a score, which should ideally correlate with the similarity of a model to the "true" protein structure. Various functions of this type have been implemented in model quality assessment programs (MQAPs), e.g. QMEAN (Benkert et al., 2007). Using MQAP scores, the models may be ranked and/or filtered before returning the result to the user.

6. **Iteration.** In order to improve the final model, steps 2 to 5 may be iterated. This procedure is time-consuming but has been shown to produce superior results as well as being able to correct errors in the initial alignment between target sequence and template structures (Burke et al., 1999).

1.4.2 Sequence Alignment

As discussed in Section 1.1.4.1, sequence alignment is used to identify similar parts of biological sequences. Large segments of such "conserved" regions are thought to indicate that the sequences share a common evolutionary origin. The optimal alignment between two sequences (the one with the highest alignment score) can be computed using dynamic programming algorithms: the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) for global alignment and the Smith-Waterman algorithm (Smith, 1981), a variation of the former, for local alignment. The terms "global" and "local" mean that the entire sequence or only short sub-sequences are assumed to be evolutionarily related. Most currently available alignment methods are based on these algorithms. Various improvements have been made since, such as the use of substitution tables (see Section 1.1.4.2) and an "affine" gap penalty (Gotoh, 1982). This gap penalty has the form:

$$s = eL + o \tag{1.1}$$

where s is the total gap penalty, L is the length of the gap, e is the gap extension penalty and o is the gap opening penalty.

With long input sequences, dynamic programming approaches become increasingly slow, as they typically run at quadratic complexity. Multiple alignments become computationally intractable using this approach (Just, 2001; Wang and Jiang, 1994).

1.4.2.1 BLAST: searching a sequence database

The most widely used method for homology detection is BLAST (Altschul et al., 1990). BLAST performs a fast alignment of one or more query sequences against a given sequence database. This is done using a heuristic k-mer (or "word") search, i.e. the search for shared sequence fragments of length k. Such local matches are then extended using a dynamic programming local alignment algorithm. BLAST produces an E-value for every hit, which represents the number of equally good hits that one would expect to obtain by chance, given the present database.

BLAST's cousin PSI-BLAST (Altschul et al., 1997) performs a simple BLAST search and then builds a position-specific scoring matrix (PSSM) from the hits with an E-value smaller than a given cut-off. A PSSM summarises the information contained in all the sequences found. Each row in the PSSM represents one of the 20 amino acids, each column relates to one position in the query string. One cell in the PSSM thus represents the frequency of observing a particular amino acid at a particular position of the query. This sequence "profile" is then used to search the database again. After every iteration the profile is updated and the procedure repeated until no more sequences are added to the profile or until a cut-off number of iterations is reached. BLAST is excellent at detecting homologs above 40% sequence identity but its power decreases below this. PSI-BLAST is able to find more distant homologs. At low sequence identity neither program produces a very accurate final alignment and thus hits are commonly realigned using other sequence alignment programs (Edgar, 2004a; Notredame, 2000;

Sadreyev et al., 2007; Thompson et al., 1994).

1.4.2.2 MUSCLE: pairwise and multiple sequence alignment

There are many sequence alignment programs. Virtually all of them are based on the Smith-Waterman algorithm for pairwise alignments. However, since the algorithm scales exponentially for multiple sequences, heuristics are needed to align more than 2 or 3 sequences at a time and solutions differ between programs. The CLUSTAL (Chenna et al., 2003) series of programs performs a tree-based multiple sequence alignment (MSA) by first performing all possible pairwise alignments and building a phylogenetic tree from the pairwise sequence similarities. This tree is then used to determine the order of adding sequences to the MSA. Clustal has been improved by additions such as sequence clustering and weighting and position-specific gap penalties (Thompson et al., 1994). However, the method cannot overcome the intrinsic weakness of tree-based methods: once a sequence is added to the MSA, its alignment to the previously added sequences never changes. Thus the accuracy of the MSA depends entirely on the accuracy of each separate pairwise alignment. This problem is partially overcome by iterative methods, such as T-coffee (Notredame, 2000) and MUSCLE (Edgar, 2004a,b), which change the initial MSA by partitioning it into smaller sub-sets of similar sequences, internally re-aligning these and then merging them back into a complete MSA. This procedure is iterated until convergence.

The method used extensively in this work is MUSCLE, which was designed to perform iterative MSA at a speed that allows high throughput application.

Progressive multiple sequence alignment with MUSCLE:

- Multiple sequence alignments (MSAs) are generally used to investigate what regions of a group of related sequences are conserved throughout evolution. MUSCLE and other progressive methods invert this assumption and infer a multiple sequence alignment from an evolutionary tree. This tree is built by hierarchical

clustering of the pairwise sequence similarities/distances.

- Given an evolutionary tree of sequences, the algorithm visits the tree nodes starting from the leaves and moving towards the root. Every leaf is assigned a sequence profile representing a single input sequence.

- As the algorithm progresses towards the root, each internal tree node is assigned a new sequence profile, which represents the alignment of its two child profiles. Then pairwise alignment is performed.

- Every internal node can also be represented as a MSA. The root node represents the final MSA, containing all input sequences.

Sequence profiles. In MUSCLE, sets of aligned sequences are represented as a profile. This profile can be thought of as string of characters, where each character encodes the information contained within a single column of its corresponding MSA, facilitating the application of pairwise alignment methods to multiple sequences.

Pairwise profile alignment. MUSCLE uses a pairwise alignment algorithm similar to CLUSTALW (Thompson et al., 1994). Briefly, this is a speed-optimised dynamic programming algorithm with substitution scores, an affine gap penalty and a half penalty for terminal gaps. The algorithm's complexity is $O(N)$ in space and $O(N^2)$ in time. See Appendix 7.1 for a more complete description of the MUSCLE algorithm.

1.4.3 Structure Alignment

Structure alignment methods superimpose two or more 3D protein structures (see Figure 1.7) and can generate a sequence alignment that reflects this superposition. The sequences of the two proteins need not necessarily be taken into account during this process. Unlike regular sequence alignment, structure alignment has no optimal solution,

as two conflicting parameters are being optimised: the number of residues superimposed should be maximised (usually only the Cα atoms are used), while the root mean square distance (RMSD) between the corresponding atoms should be minimised. A superimposition of a single residue would yield a perfect RMSD of zero, but only a minimal alignment length of one residue. Given that the two proteins are not identical in structure, increasing the number of superimposed residues will in many cases automatically raise the RMSD (Bourne and Weissig, 2003).

Due to these contradictory goals, there is no "best" structure alignment. Implementations vary greatly in their algorithms and speed. Most alignment programs perform rigid body superposition (TM-align (Zhang and Skolnick, 2005a), MAMMOTH (Ortiz et al., 2002), DALI (Holm and Sander, 1996), Combinatorial Extension (Shindyalov and Bourne, 1998)), although some allow kinks and twists (MATT (Menke et al., 2008)) in order to better align more distant homologues.

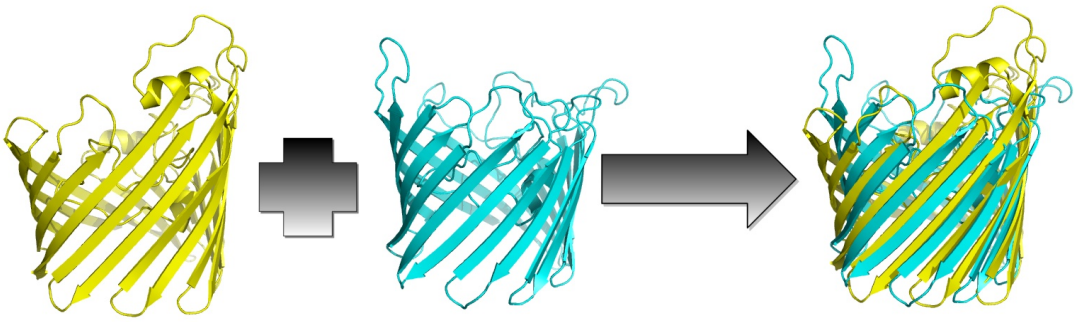


Figure 1.7: Structure Alignment - The structures of two bacterial porins; PDB entries 2POR (left, yellow), 1PRN (middle, cyan) and their superposition (right).

1.4.3.1 TM-align

One of the most widely used pair-wise structure alignment program is TM-align (Zhang and Skolnick, 2005a). It is designed for speed while performing a fairly accurate rigid body superposition of two related protein structures. More recently, a variant called MM-align (Mukherjee and Zhang, 2009) has been developed for the alignment of protein

complexes.

The main difference between TM-align and other structure alignment tools is its use of the TM-score (Zhang and Skolnick, 2004) as an optimisation criterion, instead of more traditional measures of accuracy like the RMSD, the MaxSub score (Siew et al., 2000) or the global distance test (GDT) (Zemla et al., 1999). The RMSD, while being intuitive, has the drawback of being highly sensitive to alignment errors. The other measures compensate for this but are defined by counting the proportion of "well-aligned" residues in the alignment and are thus completely insensitive to the quality of low-accuracy regions in the alignment. In addition, all the above measures have a strong correlation with protein size, which is problematic when comparing the accuracy of alignments between different protein pairs. The TM-score, used by TM-align, is defined as follows:

$$\text{TM–score} = \text{Max}\left[\frac{1}{L_{Target}}\sum_{i=0}^{L_{ali}}\frac{1}{1+\left(\frac{d_i}{d_0(L_{Target})}\right)^2}\right] \tag{1.2}$$

$$d_0(L_{Target}) = 1.24\sqrt[3]{L_{Target}-15} - 1.8 \tag{1.3}$$

where L_{Target} is the length of the target protein that other protein structures are aligned to, L_{ali} is the number of aligned residues, and d_i is the distance between the ith pair of aligned residues. $d_0(L_{Target})$ is a normalisation parameter that renders the average TM-score, for randomly related proteins, independent of protein size; Max is a function that selects the maximum score over all possible alignments.

Given two protein structures, TM-align first performs three initial alignments based on three different heuristics: The first alignment is done by aligning the secondary structure elements of the two proteins by dynamic programming. The second alignment is produced by gapless threading of the shorter protein onto the longer protein. The third alignment is produced by combining the scoring matrices of the first two alignments

and, again, running a dynamic programming algorithm on the combined scoring matrix. In all three cases, the optimal alignment is identified using its TM-score.

These three initial alignments are each fed into an iterative optimisation algorithm. Given an initial alignment, the 3D structures are optimally superimposed using the aligned residues. From this superposition a new scoring matrix is built up. The scoring matrix S has the form

$$S(i,j) = \frac{1}{1 + d_{ij}^2 / d_0(L_{min})^2} \tag{1.4}$$

where d_{ij} is the distance of the ith residue in structure 1 and the jth residue in structure 2, in the current superposition of the two proteins; L_{min} is the length of the shorter protein; d_0 is defined as in equation 1.3, but with respect to L_{min} instead of L_{Target}. As before, the optimal alignment is identified using dynamic programming and using the aligned residues a new superposition can be computed. This process is repeated until convergence, which tends to happen after only 2-3 iterations.

In all the above alignments only gap openings are penalised (except for the gapless threading step).

1.4.4 Sequence to Structure Alignment

Aligning a sequence to a structure differs from other types of alignments, in that both sequence and structure information is used to construct the alignment (see Figure 1.8). This enables higher quality alignments at low sequence identities (<40%).

In a structure prediction context, sequence-to-structure alignment methods are often referred to as "threading" or "fold recognition" methods, as their purpose here is to assign a 3D conformation (or "fold") to an input sequence. This is often done by "threading" an input sequence onto known protein structures in the PDB. A variety of scoring methods have been employed in order to identify the best available 3D conformation for a given sequence. Some methods use the full 3D structure to inform their

scoring function (e.g. using inter-atomic distances), while others first transform the 3D structure into sets of 1D annotation sequences (symbolising, for example, secondary structure).

Sippl and Weitckus (1992) used an empirical pseudo-energy function, based on Cβ-Cβ distances in known protein structures, to score alignments between the input sequence and proteins of known structure. The main drawback of this approach was that it did not support gaps in the alignment. Since then, other methods have used predicted secondary structure (McGuffin and Jones, 2003), solvent accessibility information (Chen and Zhou, 2005) or various combinations of sequence and predicted structural features (Skolnick et al., 2004; Wu and Zhang, 2008; Zhou and Zhou, 2005). The alignment engine itself is most commonly powered by dynamic programming or Hidden Markov Models (Karplus et al., 1998; Söding, 2005). The most accurate methods use a consensus of multiple approaches, for example the 3D-Jury meta-server (Ginalski et al., 2003) available at http://bioinfo.pl.

The program used in this thesis is FUGUE (Shi et al., 2001), which is called as one part of the above-mentioned meta-server. FUGUE uses a dynamic programming algorithm informed by 1D structural annotation. This annotation is directly inferred from the known 3D structure (using the program JOY, see below) and no predicted annotation is relied upon. Each residue in the protein of known structure is thus characterised in terms of its local structure environment. This environment then dictates which one of a set of environment-specific substitution tables (ESSTs) and gap penalties should be used. The two proteins are then aligned as in regular sequence alignment, but using the previously chosen substitution tables and gap penalties for each residue. Further details are given in Section 1.4.4.2 below.

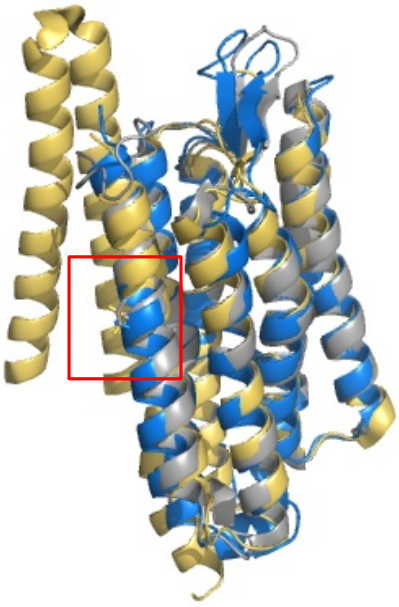
A: Sequence Alignment

```
>1h2s
structure
MVGLTTL-------FW--LGALGMLVGTLAFAWAGRDAGSGERR-YYVTLVGISGIAAVA
YVVM--ALGVGWVPVAERTVFA---------PRYIDWILTTPLIVYFLGLLAGLDSREFGI
VITLNTVVMLAGFAGAMV--PGIERYALFGMGAVAFLGLVYYLVGPMTESASQRSSGIKS
LYVRLRNLTVILWAIYPFIWLLGPPGVALL-TPTVDVALIVYLDLVTKVGFGFIALDAAA
TLRAEHGEGAVFIFVGALTVLFGAIAYGEVTAAAATGDAAAVQEAAVSAILGLIILLGIN
LGLVAATL

>2ei4
structure
QAGFDLLNDGRPETLWLGIGTLLMLIGTFYFIARGWGVTDKEAREYYAITILVPGIASAA
YLAMFFGIGVTEVELASGTVLD-----IYYARYADWLFTTPLLLLDLALLAKVDRVTIGT
LIGVDALMIVTGLIGALSKTP-LARYTWWLFSTIAFLFVLYYLLTSLRSAAAKRSEEVRS
TFNTLTALVAVLWTAYPILWIVGTEGAGVV-GLGIETLAFMVLDVTAKVGFGFVLLRSRA
IL-----------------------------------GET---------------------
--------

>1e12
structure
REN-ALLSS----SLW--VNVALAGIAILVFVYMGRTIRPGRPRLIWGATLMIPLVSISS
YLGLLSGLTVGMIEMPAGHALAGEMVRSQWGRYLTWALSTPMILLALGLLADVDLGSLFT
VIAADIGMCVTGLAAAMTTSALLFRWAFYAISCAFFVVVLSALVTDWAASAS--SAGTAE
IFDTLRVLTVVLWLGYPIVWAVGVEGLALVQSVGATSWAYSVLDVFAKYVFAFILLRWVA
NN-----------------------------------ERTVAV----------------
--------
```

C: Structure Alignment

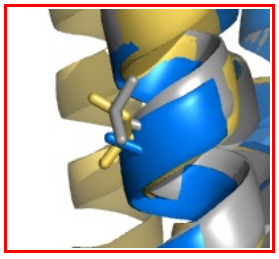


B: Joy Secondary Structure Annotation

```
>1e12
secondary structure and phi angle
CCH-HHHHH----HHH--HHHHHHHHHHHHHHHHCCCCCPHHHHHHHHHHHHHHHHHHH
HHHHHHPCCCEEEECCCPCCCCPCEEEECHHHHHHHHHHHHHHHHHHHHPCCHHHHHH
HHHHHHHHHHHHHHHHCCCCHHHHHHHHHHHHHHHHHHHHHHHCHHHHHH--HCPCHH
HHHHHHHHHHHHHHHHHHHHHCCCCCPCCCCHHHHHHHHHHHHHHCHHHHHHHHHHH
HC-----------------------------------HHHHHC----------------
--------
>1h2s
secondary structure and phi angle
CPCHHHH-------HH--HHHHHHHHHHHHHHHCCCCCCCCHH-HHHHHHHHHHHHHH
HHHH--HCPCPEECCCPCCEEH--------HHHHHHHHHHHHHHHHHHHHPCCHHHHH
HHHHHHHHHHHHHHHHC--CPCHHHHHHHHHHHHHHHHHHHCHHHHHCCCCHHHHH
HHHHHHHHHHHHCCHHHHHHCCCCCPCC-CHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHCPCCHHHHHHHHHHHHHHHHHHHHHHHHHPCHHHHHHHHHHHHHHHHHHH
HHHHHHC
>2ei4
secondary structure and phi angle
CCPCCCCPCPCCCHHHHHHHHHHHHHHHHHHHHHCCCCCHHHHHHHHHHHHHHHHHH
HHHHHPCCCEEEECCCPCEEE-----ECCHHHHHHHHHHHHHHHHHHHHPCCHHHHH
HHHHHHHHHHHHHHHCCCCH-HHHHHHHHHHHHHHHHHHHHHCHHHHHCCCCHHHHH
HHHHHHHHHHHHHHHHHHHHHCCCCCPCC-CHHHHHHHHHHHHHHHHHHHHHHCHH
HC-----------------------------------PPC----------------
--------
```

D: Corresponding Residues

Zooming in on the residues highlighted in the sequence alignment. As shown in the JOY annotation, they are part of a helix.



E: Target sequence

```
>gi|110668951
MSQHLYTRSTNSHCMRMIGPEQLDSAVLQLTQSDVLSQVQNDVLLSSSLWVNIALAGLSILLFVYMGRNITSGRARLIWGATLMIPLVSISSYLGLASGL
TVGFIEMPAGHALAGQEVMSQWGRYLTWALSTPMILLALGVLADVDRGSLFTVIAADIGMCVTGLGAALITSSYLFRWAFYIISCTFFVVVLFALLFEWP
VSAAAAGTDDIFSTLRLLTVVLWIGYPVIWAAGVEGFALIQSVGLTSWGYSGLDILAKYAFSFLLLRWVADNEPTVSSPASNTAESGAVADD
```

Figure 1.8: Sequence to (multiple) structure alignment - A multiple structure alignment (C) and its corresponding multiple sequence alignment (A) are shown. Each structure is annotated using JOY (B, only secondary structure annotation is shown). One set of corresponding residues is highlighted in sequence and structure representations. The target sequence (in FASTA format), for which a 3D model needs to be computed, is shown in E. This sequence needs to be aligned to the pre-aligned structures in C (and A), e.g. using FUGUE.

1.4.4.1 JOY: Annotating Sequence Alignments with Structural Information

JOY (Mizuguchi et al., 1998a) is a post-processor for structure or sequence alignments. It annotates a given input alignment with the local structural environment of every residue within the alignment. The complete annotation provided by JOY is listed in Appendix 7.2. The main annotations are computed as follows.

Solvent accessibility is calculated using the program PSA, which uses an implementation of the algorithm described by Lee and Richards (1971). Each residue is annotated as being either "accessible" or "inaccessible", depending on whether at least 7% of its surface is exposed on the surface of the protein.

Secondary structure and main chain conformation is calculated using SSTRUC, which uses the definitions stated by Kabsch and Sander (1983). Information on dihedral angles is also provided.

Hydrogen bonding of side chain and main chain atoms is predicted by the program HBOND (J. Overington, unpublished). Hydrogen donor/acceptor pairs within 3.5Å of each other are assumed to interact (with some additional restrictions on bond energy and angle). Hydrogen bonds are classified into four types: main chain to main chain; side chain to main chain amide; side chain to main chain carbonyl; side chain to side chain / hetero atom group.

The annotation produced by JOY can be used in the process of creating environment-dependent substitution tables (see Section 3.5.3).

1.4.4.2 FUGUE: Sequence to Structure Alignment

FUGUE (Shi et al., 2001) performs an alignment between a protein sequence and a protein structure. Its main purpose is homology detection. Given only a protein sequence, FUGUE finds homologous protein structures, which are likely to be suitable as templates for homology modelling of the target protein's structure. FUGUE was

trained and tested on subsets of the HOMSTRAD (Mizuguchi et al., 1998b) database of multiple structure alignments of protein families. This database is explained in more detail in Section 3.5.1.2.

FUGUE uses environment-specific substitution tables (ESSTs) produced by the software SUBST. The structural environments used by FUGUE were characterised using JOY.

Consider target protein A, for which only the sequence is known, and match protein B, for which both the sequence and structure are known. In order to obtain a score for the alignment between residue i in protein A and residue j in protein B, we must first choose the correct substitution table. JOY is run on protein B, thus characterising the local structural environment of every residue in protein B. Knowing the environment of residue j, we choose the corresponding substitution table. We obtain the substitution score by picking the cell corresponding to the substitution of the amino acid at position j (in protein B) to the amino acid at position i (in protein A).

Scoring matrix. The rows of the matrix represent the positions in the structure and the columns represent the 21 amino acids (including half-cysteine). M_{pb} is the score of replacing amino acids at position p of the structure with amino acid b. The scoring matrix is thus:

$$M_{pb} = \sum_i S(a_{pi}, E \rightarrow b) \tag{1.5}$$

where i is the ith structure in the alignment, a_{pi} is the amino acid of the ith structure at alignment (or structure) position p and E is the structural environment of amino acid a_{pi}.

Gap penalty matrix. Gap penalties in FUGUE are dependent on the local structure environment. For instance, insertions/deletions in the middle of a secondary structure element (SSE) are penalised.

Sequence-profile search and alignment. FUGUE uses a dynamic programming

algorithm as described by Gotoh (1982). If the length ratio between the query sequence and the structure is greater than 1.5 or less than 2/3, a global-local algorithm is used instead, with zero terminal gap penalties.

A z-score is finally calculated to assess the quality of the match between query sequence and the given structure profile.

FUGUE is one of the most widely used methods for sequence-to-structure alignment and homology detection. According to the authors (Shi et al., 2001), at the time of publication, FUGUE significantly outperformed various other popular methods for homology detection, including PSI-BLAST. At the family level, FUGUE attained a sensitivity of 49% (at 99% specificity) and 62% (at 50% specificity), compared to 42% and 50% for the next best method, a derivative of PSIBLAST. At the superfamily level this difference is smaller as detection levels of all methods sink dramatically. Here, FUGUE's sensitivity reached 4% and 13%, compared to 4% and 10% for the next best method, PSI-BLAST.

1.4.5 Co-ordinate Generation

The classical method for co-ordinate generation is the assembly of rigid bodies (COMPOSER (Sutcliffe et al., 1987), 3D-JIGSAW (Bates et al., 2001), SWISS-MODEL (Schwede et al., 2003)). Here, a "framework" is calculated from the templates by finding conserved regions within the multiple alignment and averaging the Cα co-ordinates across the templates (see Figure 1.9). Main chain atom co-ordinates of each core segment of the target model are then obtained from one of the templates and superimposed onto the framework. Segments which still have no structure are considered to be loops. These are filled in by searching a loop database for a loop with matching anchor regions (structured regions immediately adjacent to the loop) and similar sequence. Side chains are finally added to the model and their conformation optimised using energy minimisation or molecular dynamics.

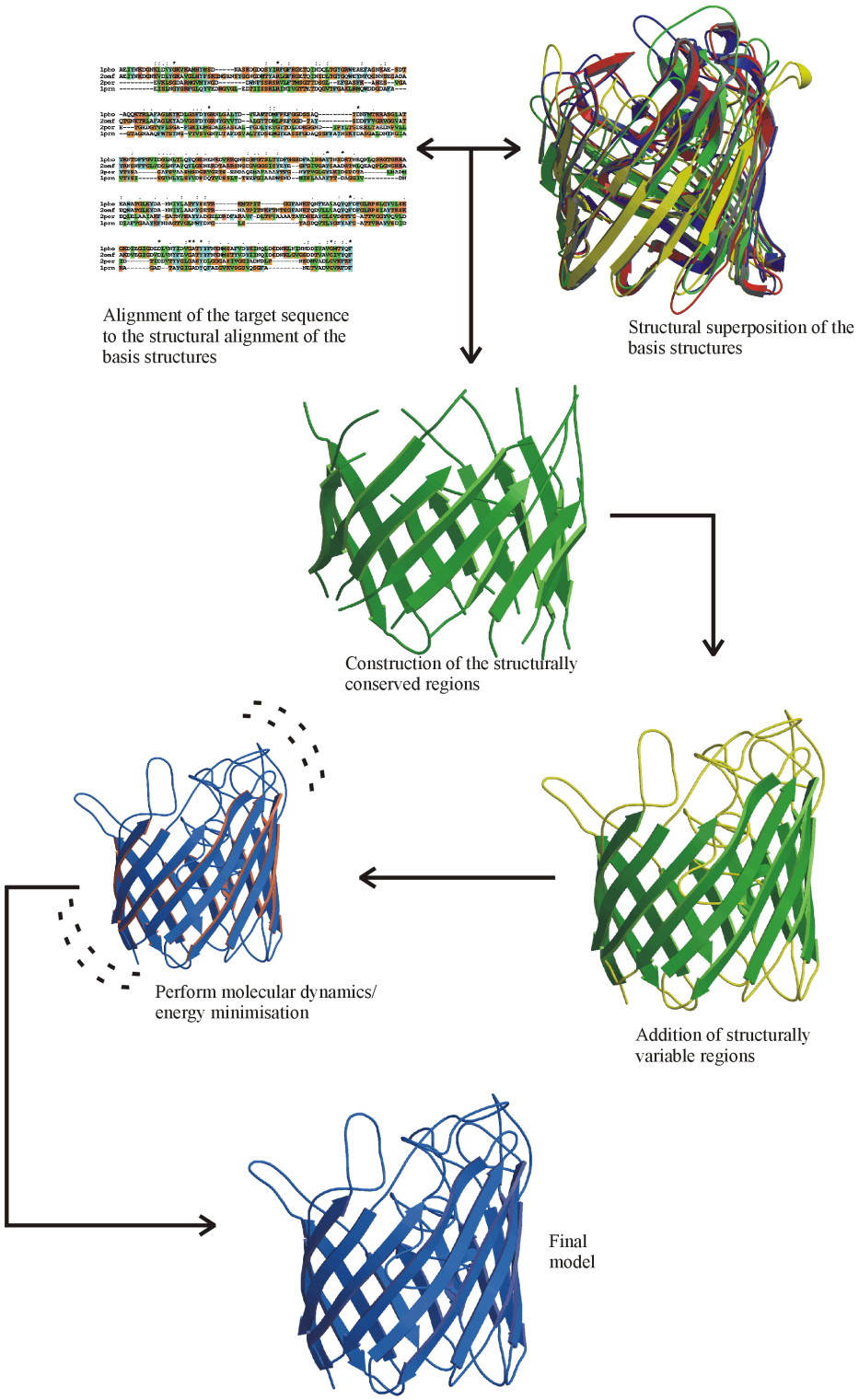


Figure 1.9: Comparative modelling by assembly of rigid bodies - Algorithm used in homology modelling by assembly of rigid bodies (see text for details). Figure taken from (Deane, 2000).

Another type of approach is modelling by satisfaction of spatial restraints (MODELLER (Sali and Blundell, 1993)). First, many constraints on the structure of the target sequence are generated from the initial alignment of the target sequence to the template structures. The assumption made is generally that distances (in 3D space) between corresponding atoms in the alignment are similar in the target and template proteins. Homology-derived restraints are usually combined with stereochemical restraints on bond lengths and angles, as well as non-covalent interactions obtained using a classical molecular mechanics force field. The final model is obtained by an optimisation process that minimises the violations of these restraints.

1.4.5.1 MODELLER

MODELLER implements comparative (template-based) structure modelling by satisfaction of spatial restraints. The program is designed to accept a great variety of restraints, some of which are listed below.

Homology-derived restraints. In the initial step of 3D co-ordinate generation, the 3D co-ordinates of the target protein's residues are derived from a multiple alignment with template proteins of known structure. Typically, spatial restraints thus obtained include main chain dihedral angles and Cα-to-Cα distances, which are correlated in structurally similar proteins (Sali and Blundell, 1993). These relationships are expressed as probability density functions, which can directly be used as spatial restraints.

Stereochemical restraints. MODELLER uses the CHARMM22 force field (MacKerell et al., 1998) to enforce proper stereochemistry.

Experimentally derived restraints. Due to the modular nature of the approach, restraints from any source may be added to the homology-derived restraints. Examples of such are secondary structure packing data (Cohen and Kuntz, 1989) and analyses of hydrophobicity (Aszódi and Taylor, 1994). This allows the easy adaptation of the

MODELLER approach to the protein being targeted for modelling.

Final co-ordinate generation. The spatial and other restraints are combined into an objective function, which depends on the Cartesian co-ordinates of the atoms of the target protein. This objective function is optimised using conjugate gradients and molecular dynamics with simulated annealing (Clore et al., 1986), to yield a new model. The model is improved by slightly varying the initial co-ordinates before optimisation, producing a range of final conformations, from which the "best" one may be selected using either the value of the objective function or by using model quality assessment programs (MQAPs).

1.5 Towards Accurate Prediction of Transmembrane Protein Structure

As illustrated above, there is a multitude of available methods facilitating the modelling of protein structure. Some of these individual methods can be applied directly to membrane proteins, while others need to be adjusted and improved to fit the physico-chemical properties of membrane proteins. This project aims to develop an integrated approach to the accurate modelling of transmembrane proteins from their sequence.

Chapter 2 describes the first step towards this goal: iMembrane, a method for accurate prediction of a protein's location within the membrane. iMembrane combines existing alignment methods with a new simplified membrane model based on a database of coarse-grained molecular dynamics simulations of membrane proteins. This new approach will be used in subsequent chapters to build environment-specific substitution tables of membrane proteins and ultimately to model membrane protein structure.

Chapter 2

iMembrane: Predicting a Protein's Position Within the Membrane

2.1 Context

The overall aim of this work is to accurately predict the structure of a transmembrane (TM) protein from its amino acid sequence. The road from sequence to model structure is long and typically involves a multitude of steps performed by a plethora of programmes. The method described in this chapter, iMembrane, is the first step along this path and is the basis for all subsequent work described in this thesis.

iMembrane predicts a membrane protein's "membrane insertion", i.e. its three-dimensional position within the lipid bilayer. This knowledge is of paramount importance in any structural analysis of membrane proteins and also to the prediction of membrane protein structure.

In Chapter 3, iMembrane is used to partition membrane proteins into sections in contact with the various parts of the lipid bilayer. This partitioning can be used to

describe the molecular evolution occurring in each of these structural environments, in the form of environment-specific substitution tables (ESST).

In Chapter 4, iMembrane is used to annotate the template structure in a homology modelling pipeline. This annotation is then used to select the correct ESST for scoring each part of the target-template alignment, which leads to the generation of higher-accuracy model structures.

The following sections give further motivation for the iMembrane approach, details on its internal workings and, finally, a validation of the method's accuracy.

2.2 Existing Methods for Membrane Proteins

All the approaches described in Chapter 1 have been widely used in the modelling of globular soluble proteins. However, applying them to the modelling of membrane proteins has been less successful (Elofsson and von Heijne, 2007).

Theoretically, considering the physical environment of membrane proteins, the presence of the membrane should impose more restrictions on the proteins' structures, thus making the prediction task easier. In order to exploit these spatial restraints, information regarding the protein's position relative to the membrane is required. This data is not currently available from experiments.

Obtaining an accurate membrane protein structure is difficult and costly. However, even given the structure of a membrane protein, no matter by which method it was obtained, one vital piece of information is still missing: the protein's position within the lipid bilayer. Natural ligands or drugs must be able to access the part of the protein to which they bind. Therefore it is important to be able to distinguish the parts of the protein that are within the lipid bilayer from those that are solvent-accessible. This information is not currently available from experiments. Structures obtained by X-ray crystallography or NMR spectroscopy do not easily allow the elucidation of the

protein's native lipid bilayer environment.

Studies have shown (Elofsson and von Heijne, 2007; Eyre et al., 2004) that the parts of the protein in contact with the membrane have different physico-chemical properties from those that are in contact with water. This is expected, since the hydrophobic tails of membrane lipids constitute a radically different environment from an aqueous solvent, both in terms of hydrophobicity and physical crowding. The ability to distinguish between parts of a protein that are in contact with these different environments should thus be highly informative and advantageous in a structure prediction setting.

One of the most widely used methods to predict the position of transmembrane (TM) helices is TMHMM (Krogh et al., 2001). The program takes as input a single protein sequence and outputs the predicted locations of TM helices within the sequence.

TMHMM is based on the work of Jones et al. (1994), who described a dynamic programming method, which implicitly combines the use of two signals associated with transmembrane helices: hydrophobicity (Argos et al., 1982) and charge bias (von Heijne, 1986, 1994). In addition, a "grammar" is used for the prediction, by considering the protein as a whole: cytoplasmic and extra-cellular loops have to alternate with TM helices. TMHMM uses a hidden Markov model (HMM) (Sonnhammer et al., 1998) to model a membrane protein's topology in a similar way to Jones et al. The positions of TM helices are modelled, as well as whether the N and C termini are inside or outside the cell. In addition, TMHMM uses the width of a typical membrane to constrain the length of the TM helix core.

According to the authors (Krogh et al., 2001), TMHMM finds over 95% of TM helices and correctly distinguishes between transmembrane helix proteins and other proteins in 99% of cases (if no signal peptides are present). On a per-residue level, it reaches (cross-validated) accuracies of 79%. According to the authors, the most common mistake made by TMHMM is to predict signalling peptides as TM helices. It should thus be used in combination with signal peptide prediction programs, such as

SignalP (Bendtsen et al., 2004). In addition, reviews state that the accurate prediction of helix boundaries as well as the identification of half-helices, which do not span the entirety of the membrane, remain problematic (Cuthbertson et al., 2005).

HMM-B2TMR is a HMM-based method to predict the topology of a transmembrane β-barrel from its protein sequence (Martelli et al., 2003). The method is very similar to TMHMM. The method takes in evolutionary information in the form of a multiple sequence alignment of the target sequence with related sequences. The HMM used differs from typical HMMs in that every state is represented not by a single character, but by a vector containing the information obtained from a single column in the multiple alignment. It therefore shares certain features with position-specific scoring matrices. The transition probabilities from a single state depend on the 20 parameters of that state (i.e. the occurrence of the 20 amino acids in the corresponding alignment column).

According to the authors (Martelli et al., 2003), HMM-B2TMR is 83% accurate on a per-residue basis. When distinguishing between β-barrel membrane proteins and other proteins, the method correctly rejects 90% of globular proteins and 90% of all-α membrane proteins.

OPM (Orientation of Proteins in Membranes) is an entirely different style of program. It inserts existing membrane protein structures into a model membrane using a force field (Lomize et al., 2006). The membrane is modelled as a simple hydrophobic slab. Three parameters are optimised: the protein's angle with respect to the membrane plane, its position relative to the membrane and the thickness of the membrane slab (see Figure 2.1). The Z axis in the co-ordinate system represents the membrane normal. The energy ΔG being minimised is defined as follows:

$$\Delta G_{transfer}(\rho, \tau, z_0, d) = \sum_i ASA_i \sigma_i^{W \to M} f(z_i) \tag{2.1}$$

where ASA_i is the accessible surface area of atom i, $\sigma_i^{W\to M}$ is the solvation parameter of atom i, d is the shift of the protein centre along the Z axis, z_0 is half of the membrane hydrophobic thickness, τ is the tilt angle of the protein axis relative to the Z axis and ρ is the rotation angle defining the direction of tilt. The membrane water penetration profile $f(z_i)$ has a Boltzmann sigmoidal form, which fits results of electron paramagnetic resonance (EPR) studies. What it does not allow is any deviation of the input structure (i.e. it remains rigid).

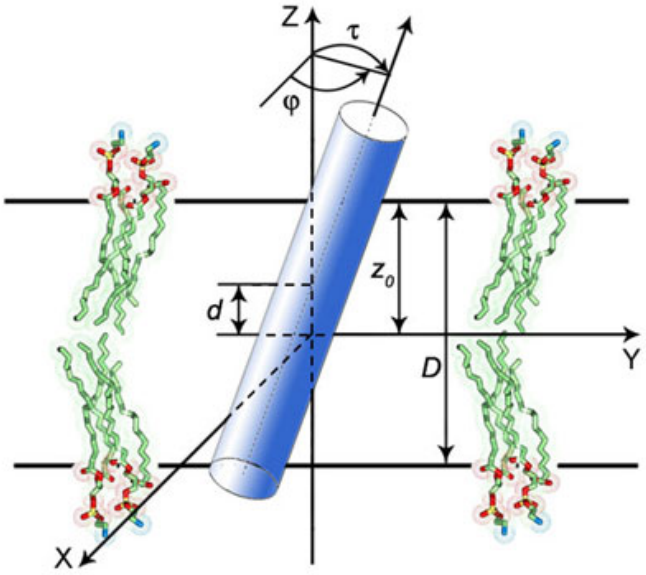


Figure 2.1: A transmembrane helix in a hydrophobic slab - Illustration of the parameters optimised by the OPM algorithm. A hydrophobic slab is tilted (angles τ and ϕ) and its parallel boundary planes moved ([x,y,z] position and distance D), such that the transfer energy $\Delta G_{transfer}$ (see text) is minimised. Figure taken from the OPM website, http://opm.phar.umich.edu, by permission of the author.

Full-blown all-atom molecular dynamics (MD) simulations of membrane proteins provide a higher level of detail than the simple model used in OPM. Unfortunately, running such a simulation takes weeks on a supercomputer. Coarse-grained MD simulations reduce the computational complexity by reducing the number of particles modelled.

The Coarse Grained Database (CGDB) (Scott et al., 2008) represents an

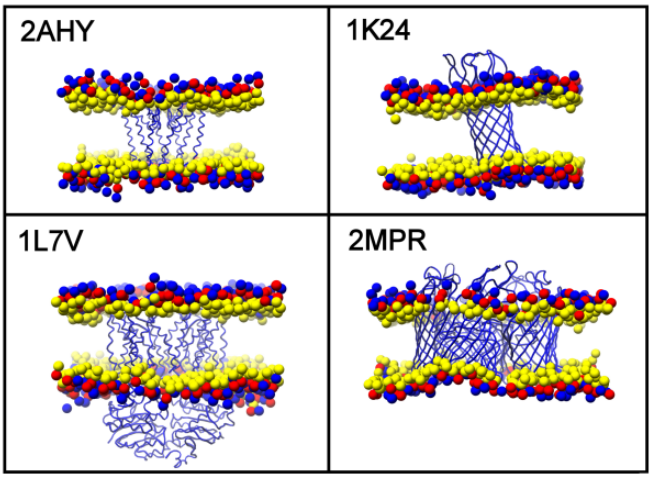


Figure 2.2: Membrane insertion of example proteins from CGDB - Four membrane proteins, after coarse-grained MD simulation. The coloured spheres represent lipid head groups. Lipid tails have been removed for illustration purposes. Figure taken from the CGDB website, http://sbcb.bioch.ox.ac.uk/cgdb/, by permission of the authors.

amino acid by only one main chain atom and zero to three side chain atoms, depending on the residue. Protein structure is maintained through the use of an elastic network model. Prosthetic groups (such as heme groups) are omitted. Membrane lipids are simplified but modelled explicitly, which gives a relatively high level of detail, compared to models such as OPM. The GROMACS code is used for simulations (Lindahl et al., 2001; van der Spoel et al., 2005). See Figure 2.2 for schematics of example simulation results.

CGDB currently (11 Feb 2011) contains 380 membrane proteins from the PDB. Although the database is not non-redundant, in terms of sequence identity, the proteins are manually selected to be representative of the entire set of known membrane protein structures. CGDB is an ongoing effort and should continue to grow, as more membrane protein structures become known.

Many of the proteins in CGDB have been simulated in the presence of different kinds of membrane lipids. Depending on the length of the lipid tails and the charge of the head groups, the results differ slightly between lipid types.

Overview of the CGDB modelling protocol. Firstly, a coarse-grained (CG) model is generated from the original all-atom model. A steepest descent energy minimisation is performed to relax any steric clashes within the protein model. The model is then aligned to the Z axis of the simulation box and surrounded by randomly placed lipid molecules. Coarse grained water particles and Na+ or Cl- ions are added to balance the overall charge of the system to zero. Another round of steepest descent energy minimisation (200 to 400 steps) is now performed on the entire system. After bilayer formation, lipid molecules outside the bilayer are replaced with water molecules. Ten more steps of steepest descent energy minimisation are perfomed, then velocities are reassigned and a 200ns simulation is performed. This simulation is recorded and a summary made available online.

The CGDB database offers analyses of the simulations in the form of tables listing the fractions of time each residue spent in contact with the different parts of the lipid molecules. These interactions can be either with a) the polar head groups of the lipids, b) the hydrophobic tails of the lipids or c) neither (i.e. only with water or other amino acids). These simulation summaries are used extensively in this chapter.

Performing MD simulations – even coarse-grained ones – requires large amounts of time and processing power. The approach described in this chapter, iMembrane, is a fast method to project existing simulation results onto proteins of homologous structure or sequence. These projected results are shown not to vary greatly from those obtained in the original coarse-grained simulations. Where performing an original simulation would take days on a compute server, iMembrane takes mere seconds on a modern desktop computer. This allows the user to take full advantage of the detailed membrane model used in CGDB, even if the protein of interest has not yet been simulated. In addition, iMembrane provides further levels of abstraction from the original CGDB simulation data, presented in a standardised, automatically parsable format. Lastly, iMembrane can be applied to proteins where only sequence information is available.

2.3 Overview of the iMembrane procedure

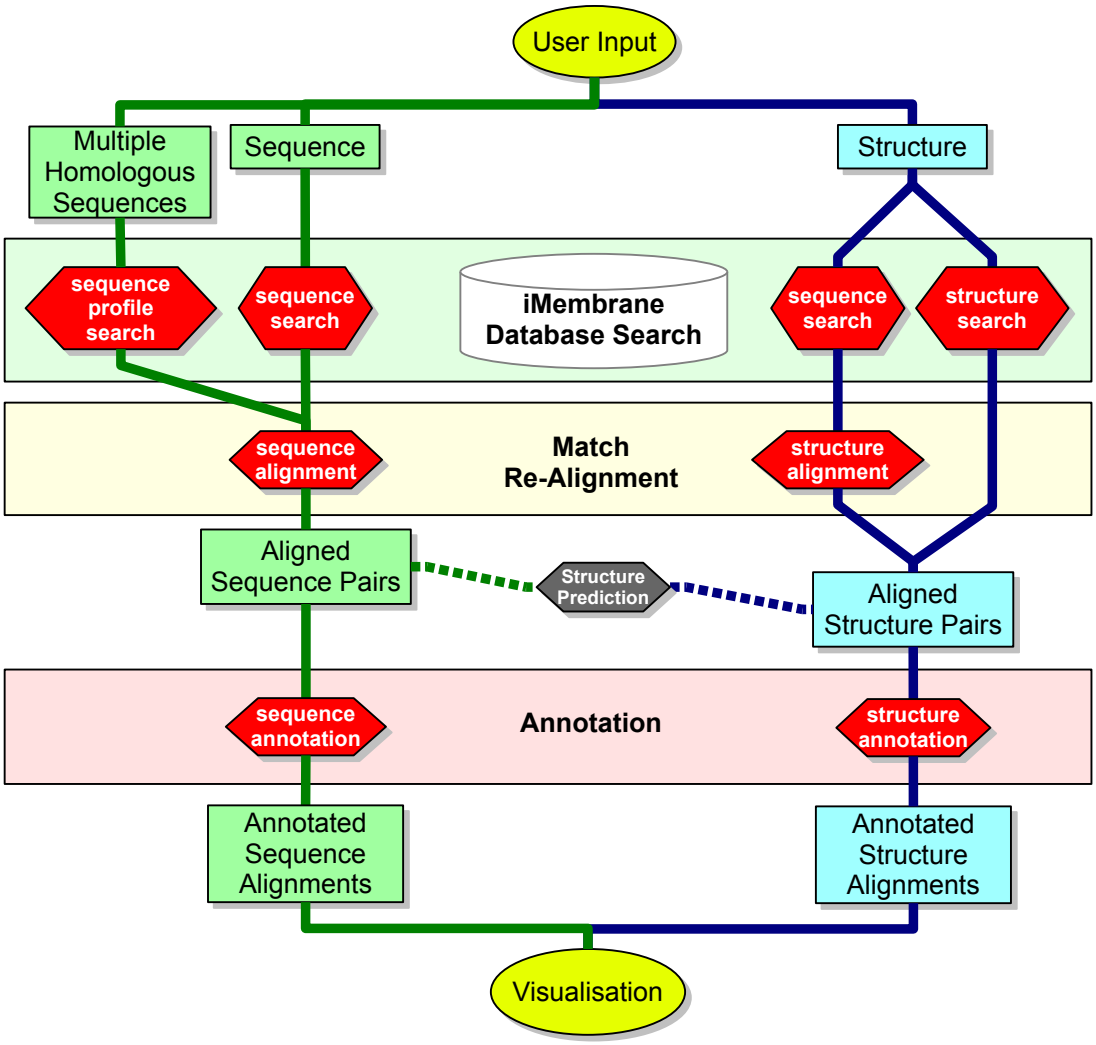


Figure 2.3: Flowchart of iMembrane's main algorithm - The diagram shows each major step in the iMembrane procedure, starting from the user input right down to the visualisation of the programme's output. If iMembrane is run via the web interface (www.imembrane.info), visualisation occurs automatically.

1. The input to iMembrane can either be a sequence, in FASTA (Pearson, 1990) format, or a structure, in PDB (Berman et al., 2000) format.

2. In the case of a structure, its sequence is first extracted from the ATOM records in the structure file.

3. A BLAST (see Section 1.4.2.1) sequence search is then carried out against the iMembrane database of pre-annotated membrane proteins (see Section 2.4), which was built from the CGDB database.

4. Matches are re-aligned to the query using either MUSCLE (see Section 1.4.2.2) sequence alignment or TM-align (see Section 1.4.3.1) structure superposition.

5. Guided by the alignment, iMembrane then projects the database protein's annotation onto the query protein.

A flow chart of this algorithm is shown in figure 2.3. The procedure outlined above is based on the principle of homology. Sequence and structure alignment methods rely on the assumption that proteins of similar sequence share a common evolutionary origin. Similar sequence means similar structure and thus similar function. Analogously, it is assumed here that aligned regions of membrane proteins share similar membrane contact patterns.

We provide two types of annotation of the input sequence and/or structure. Each residue of the input protein is labeled with a single letter: N (not in contact with the membrane), H (in contact with the polar head groups of the membrane lipids) or T (in contact with the hydrophobic tails of the lipids). In the first instance, these letters simply represent a discretised interpretation of the raw simulation results provided by the CGDB database. From now on, we will refer to this type of data as the *"membrane contact"* annotation (see Figure 2.4B).

We also provide a simplified model, which abstracts the membrane as a three-layered slab, with an inner region around the membrane lipids' hydrophobic tails, and two peripheral regions surrounding the membrane lipids' polar head groups. The boundaries of these layers are calculated by fitting planes onto the membrane contact data so as to minimise the number of mis-classified N,H or T contacts. This simplified model allows us to use each residue's three-dimensional co-ordinates to determine, in which layer of

the membrane it resides (or whether it is outside the membrane). Therefore, this type of data shall be referred to as the *"membrane layer"* annotation (see Figure 2.4C).

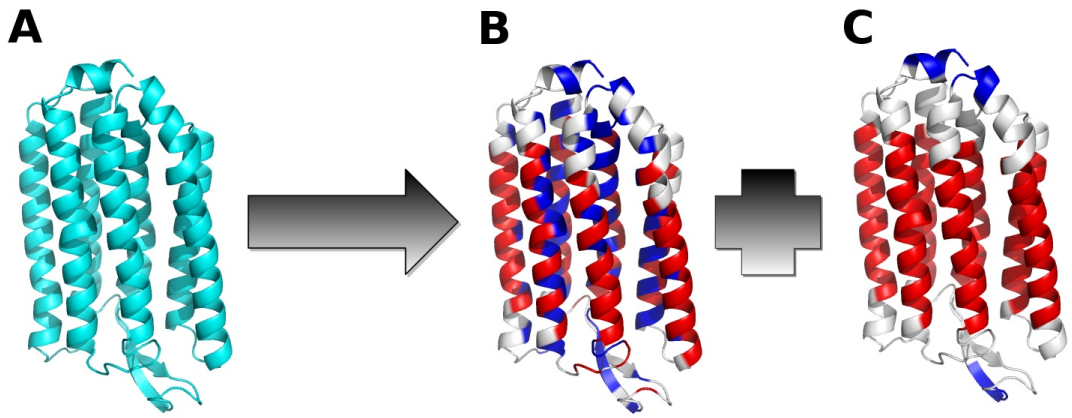


Figure 2.4: The structure of protein 2JAF before and after annotation with iMembrane - (A) 2JAF before annotation with iMembrane; (B) iMembrane's *membrane contact* annotation: red="in contact with lipid Tails", white="in contact with lipid Head groups", blue="Not in contact with the membrane"; (C) iMembrane's *membrane layer* annotation: red="in the lipid-Tail-spanning layer", white="in the lipid-Head-spanning layer", blue="Not in the membrane".

The main conceptual difference between the two types of annotation is that *membrane contact* distinguishes only between residues that are in contact with the head (H) or tail (T) groups of the membrane lipids. It cannot distinguish between residues outside the membrane and those buried inside the protein, both of which are labeled N (not in contact). The *membrane layer* annotation, on the other hand, will annotate residues outside the membrane as N (not in the membrane) and buried residues as either N, H or T, depending on their location with respect to the membrane. The drawback of this type of annotation is that it cannot distinguish between buried and accessible (surface) residues within one membrane layer. However, as residue accessibility can be computed from a protein's 3D co-ordinates (using JOY, see Section 1.4.4.1), this is not a problem. The only remaining issue now is to distinguish between surface residues within the membrane which are (a) in contact with the membrane, or (b) not in contact with the membrane because they are lining an aqueous pore (e.g. the inside of an

ion channel). This conundrum can be solved by combining the knowledge gained from both types of membrane annotation. Thus, by using *membrane contact*, *membrane layer* and the traditional *accessibility* annotation, we now have a complete picture of a protein's position within the membrane.

In the case where the input to our method is a structure, structure alignment to a database protein is performed. Then, each residue can be annotated as part of one of the membrane layers, defined by the aligned database protein, based on its 3D co-ordinates (see Figure 2.7).

In the case of a sequence-only input, the query's three-dimensional information is missing. It is true that, by aligning the input sequence to a database structure, we implicitly assign a fold to the aligned residues. These residues can thus be reliably annotated. However, this does not allow the direct inference of the unaligned residues' 3D co-ordinates (if this were the case, the protein folding problem would be solved). If the unaligned region is long enough, the residues therein could conceivably be in any one of the membrane layers. Therefore we restrict ourselves to annotating only those residues that are aligned to a database protein's residues. In order to completely annotate the input sequence, one would first perform structure prediction and then use iMembrane to annotate the resulting model structure. Structure prediction for membrane proteins is considered in detail in Chapter 4.

2.4 Building the iMembrane database

2.4.1 Database organisation and annotation format

The iMembane database (iMemDB) unifies the information found in the CGDB (see Section 2.2) with the PDB-deposited all-atom X-ray structures. Annotation data is stored in a simple and consistent format, making it easily useable by automated methods.

The CGDB annotation was first simplified into a *membrane contact* annotation (described earlier in this chapter) and saved in the standard JOY format (see Section 1.4.4.1). This allows easy combination with other structural annotation and facilitates the projection of the membrane annotation onto homologous proteins using alignment methods. In addition to the *membrane contact* annotation, a simplified *membrane layer* model was created from the contact data, such that every residue in a protein's structure can be annotated with its position relative to the membrane. Both annotation types, and the way they are constructed, are described in detail below. PDB structures corresponding to the CGDB entries were split into chains and structurally aligned to the coarse-grained CGDB structures using TM-align (see Section 1.4.3.1). The annotation of the CGDB structures was transferred to the PDB proteins, resulting in iMemDB, a new database that is easily parseable, contains all the relevant information from CGDB and is directly compatible with the structures deposited in the PDB.

2.4.2 Membrane contact model

The *membrane contact* model is, in the first instance, simply a discretised form of the information found in the CGDB. It adds no new information but makes existing information more accessible to automatic methods such as iMembrane.

In its simulation summary tables CGDB provides, for each residue, the fraction (from 0 to 1) of simulation time spent in contact with any part of the membrane (F_{total}), or specifically with the polar head groups (F_H) or the hydrophobic tail groups (F_T). F_H and F_T may not sum up to F_{total}, as residues can simultaneously be in contact with both head and tail groups.

Using the above data, each residue is classified as being mostly in contact with lipid head groups (H), lipid tail groups (T) or neither (N), by applying the following cut-off values:

`IF` $F_{total} \leq 0.1$`:`

```
        assign class N
ELSE:
        IF F_H > F_T:
          assign class H
        ELSE:
          assign class T
```

Thus every residue in the protein is assigned to one of the three contact classes N, H, or T (see Figure 2.4 B and Figure 2.5 A). The contact profile of the protein can now be represented as a sequence composed of the letters N, H and T, in a manner analogous to the JOY format.

2.4.3 Membrane layer model

Compared to the *membrane contact* model, the *membrane layer* model is a far more abstract representation of the membrane.

The membrane layer model cuts the entire protein into 5 pieces using 4 parallel planes (see Figure 2.4 C and Figure 2.5 B). The middle section of the protein is in the same layer as the hydrophobic membrane core (lipid tails, T), the two adjacent sections represent the polar outer layer of the membrane (polar head groups, H) and the two outermost sections are surrounded by the aqueous solvent (not in membrane, N).

In addition to the membrane contact time fractions, values for each residue's accessible surface area (ASA) are extracted from the 3D structures of the CGDB-simulated molecules. ASA values are rescaled as fractions between 0 and 1, by dividing each value by the maximum ASA value in the dataset. A residue not in contact with the membrane but having a high ASA is assumed to be in contact with water.

In CGDB, every simulated protein/membrane complex is transformed in Cartesian space such that the Z axis is parallel to the bilayer normal. The angle of the protein with respect to the membrane is thus implicit.

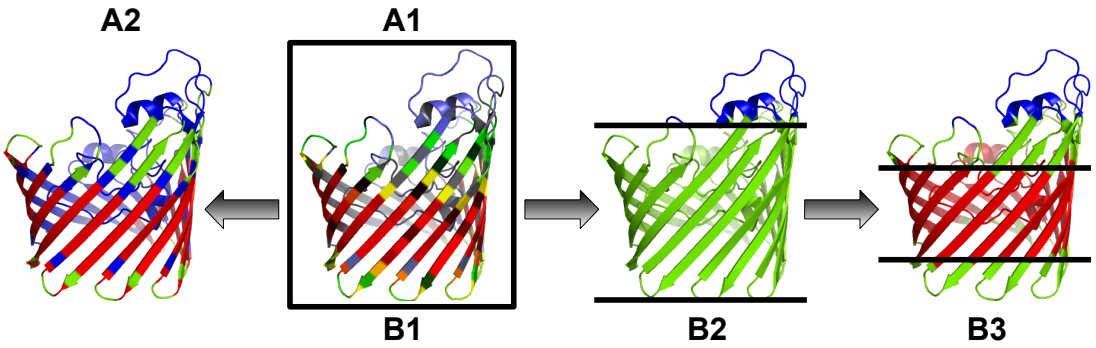


Figure 2.5: Computing the membrane annotation - The protein shown is the CGDB protein 2POR (a porin found in the outer membranes of bacteria). Colours indicate membrane annotation: (A) 1: Raw CGDB contact data; 2: *membrane contact* annotation (3 discreet colours); (B) 1: Raw CGDB contact data; 2: finding a single membrane layer (Head + Tail); 3: finding the membrane Tail layer, resulting in the final *membrane layer* annotation

First, the 3D structure of the protein is cut into discrete, 0.2Å wide slices, all of which are parallel to the membrane (i.e. every slice is penetrated by the bilayer normal). For each of the parameters *fraction of head group contact (H score)*, *fraction of tail group contact (T score)* and *rescaled ASA (N score)*, a profile is now computed (see Figure 2.6A). The X axis of each profile lists the 0.2Å bins, from one side of the membrane to the other. Along the Y axis are the respective parameter values summed over all residues in the current slice.

A single slice is thus assigned three scores (one per profile), which bias it to be classified as being in the membrane tail layer (T), one of the membrane head layers (H) or outside the membrane (N). In order to obtain the best sectioning of the protein, the following linear search procedure is carried out for each class boundary (which is a plane parallel to the membrane):

The input to the linear search function are two profiles (S_{type1} and S_{type2}), which contain the scores for each protein slice being classified into one of two membrane layer types (e.g. *Head layer* and *Tail layer*). The plane defining the boundary between the two layer types is moved, in discrete steps, to every possible position between any two

adjoining protein slices. At each step, a misclassification score is computed. Finally, the position with the minimal misclassification is chosen.

For every possible position j of the boundary plane, the score S_j has the following form:

$$S_j = \sum_{i}^{j} S_{type1} + \sum_{j}^{k} S_{type2} \tag{2.2}$$

where i, j and k are boundaries between protein slices; i is before the first slice to be considered, k is after the last slice to be considered and j is the current boundary being scored. The boundary where S_j is minimal is chosen.

This procedure is applied in a stepwise fashion to every one of the 4 planes:

1. **Find the membrane.** The optimal boundaries between the two water phases and the membrane are computed. The input to the function are the N score profile (water phase) as well as the summed profile of the H and T scores (membrane), both of which span the entire length of the protein structure, along the bilayer normal. This procedure is carried out for both water-membrane boundaries separately (see Figure 2.6B, steps 1&2).

2. **Find the membrane core.** The optimal boundaries between the core membrane layer (tail groups, T) and the two outer layers (head groups, H) are now computed. The profiles used in this step are the H score (outer membrane layer) and the T score (core membrane layer) profiles, both of which are cropped to span only the region between the two previously computed water-membrane boundaries. This procedure is carried out for both head-tail boundaries separately (see Figure 2.6B, steps 3&4).

Like the membrane contact model, the *membrane layer* model can be represented with the three letters H (head group layer), T (tail group layer) and N (non-membrane layer). The *membrane layer* profile of a protein is thus a sequence composed of the characters N, H and T, analogous to the JOY format.

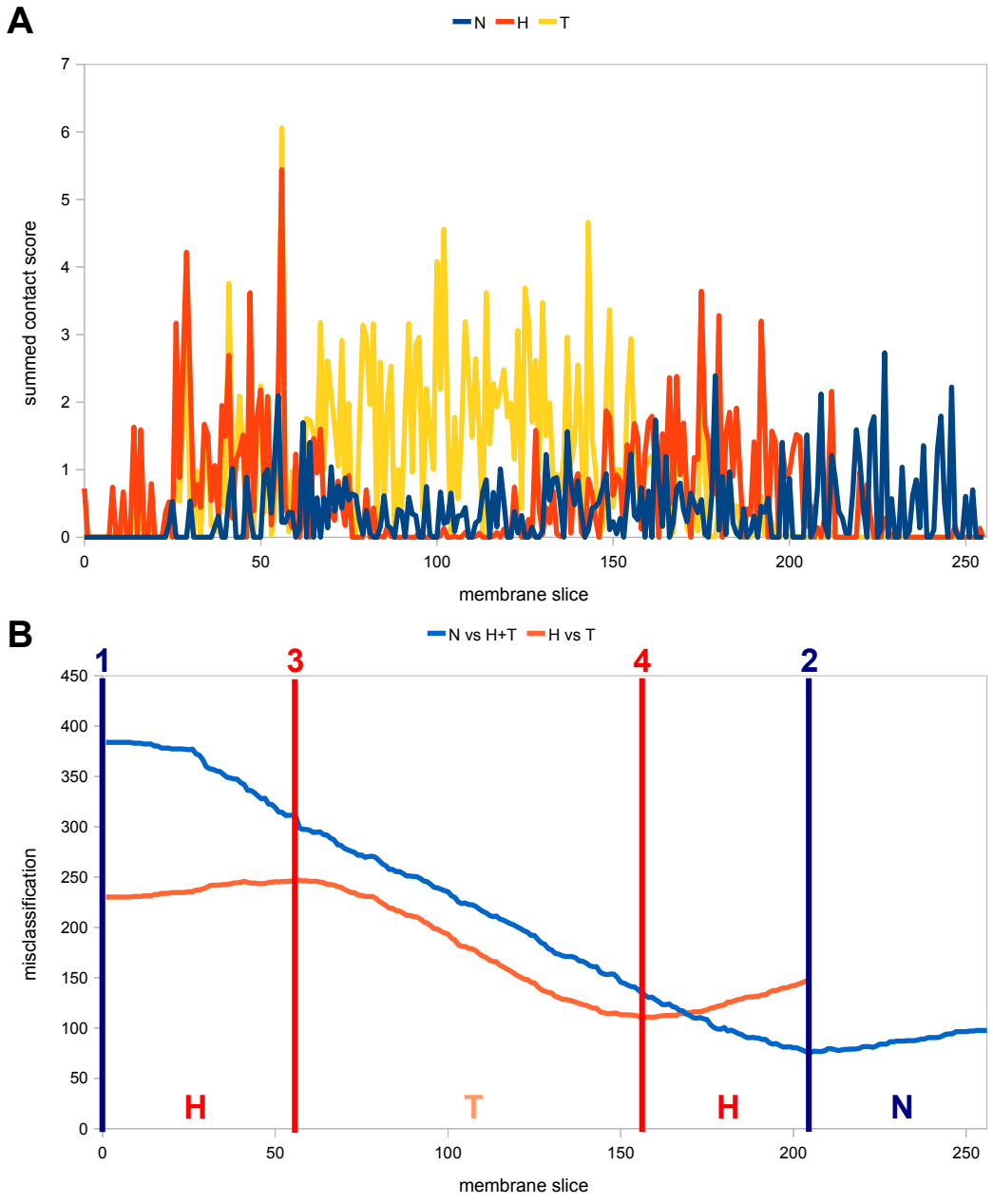


Figure 2.6: Computing the membrane layers from CGDB simulation results - The data shown relates to CGDB protein 2POR. **A:** Profile of the CGDB contact annotation (H and T scores) and normalised solvent accessibility (N score). The X axis lists all 0.2Å wide protein slices perpendicular to the membrane normal. The Y axis shows the total score for all atoms within a slice. **B:** Locating the optimal layer boundaries. X axis as above. The Y axis shows the misclassification score for each slice (see text). *Steps 1&2:* Locate the membrane; *steps 3&4:* locate the membrane core (T layer). This procedure is equivalent to that visualised in Figure 2.5B.

2.5 Homology-based prediction of a protein's membrane insertion

The previous section focused on how the initial iMembrane database was created. This section describes how this database is now used to annotate a given input protein, not present in the CGDB, with the same kind of annotation. The only prerequisite is that the given input protein is homologous to at least one of the database proteins.

2.5.1 Transfer of membrane annotation via structure homology

The process described in the previous section can be extended to annotate any additional homologous protein P, whose structure is known, but for whom no simulation results are available.

We assume that homologous membrane proteins interact with the membrane in a similar fashion. Having computed the *membrane layer* information for a CGDB protein C homologous to protein P, we now structurally align protein P to protein C. This way we move protein P into a virtual membrane, whose co-ordinates were derived from protein C. The *membrane layer* profile of protein P can thus be derived from every residue's Z co-ordinates (along the bilayer normal). All residues can be annotated in this manner – there are no gaps in the annotation (see Figure 2.7).

This procedure is only possible for the *membrane layer* annotation. *Membrane contact* annotation cannot be transferred in the same manner. Instead, an alignment is needed (either an implicit alignment resulting from a structure superposition or one obtained through sequence alignment methods).

2.5.2 Transfer of membrane annotation via sequence homology

If no structure is available, a sequence alignment can be performed between the target and a homologous protein in the iMembrane database. It is then possible to transfer

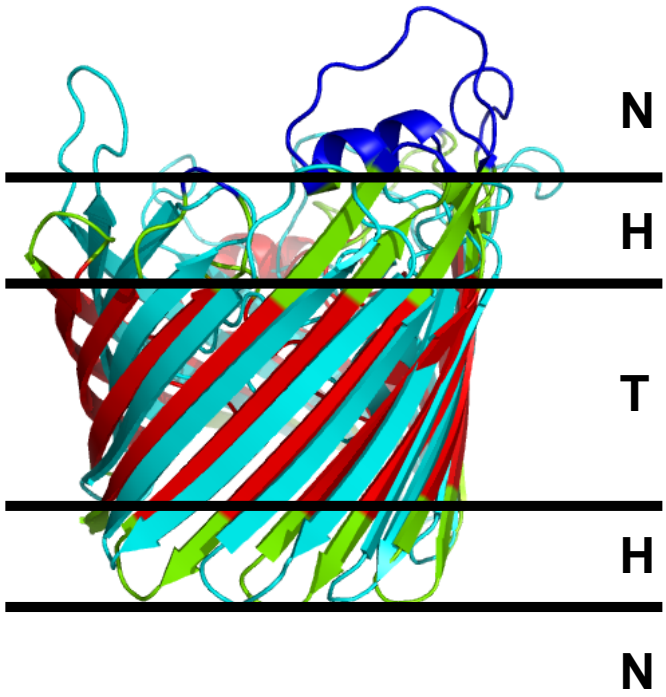


Figure 2.7: Transferring membrane layer annotation via structure homology - The query protein (cyan) has been aligned to its matching CGDB protein 2POR (in red, green and dark blue). The membrane boundaries (black lines) defined using the CGDB annotation of 2POR can now be used to annotate the query protein. Membrane layers are labelled in black letters (N: non-membrane layer, H: head layer, T: tail layer)

Molecule	Data Type	Sequence
1PRN	Query sequence	EISLNGYGRFGLQYVEDRGVGLEDTIISSRLRINIVG
2POR0	CGDB structure	EVKLSGDARMGVMY------NGDDWNFSSRSRVLFTM
2POR0	membrane contact	NNNNNNNNNTTTNT------NHHHTHTNNNNNNNNNN

Figure 2.8: Transferring annotation via sequence alignment - Annotation can be transferred between aligned residues in a sequence alignment. Here, the sequence labelled "2POR0" is an annotated CGDB protein (only the first 37 residues are shown). Membrane contact annotation is shown as a sequence of "N", "H" and "T". Alignment columns are coloured according to this annotation. Unaligned regions of the query sequence cannot be annotated using this method and are thus not coloured. ***Key:*** *T: In contact with lipid Tails (red), H: In contact with polar Head groups (yellow), N: Not in contact with membrane (blue).*

annotation from one protein to the other using the residue equivalences defined by the alignment. This is true for both *membrane layer* and *membrane contact* annotation.

A residue from protein P, aligned to an annotated residue of protein C, simply receives the same annotation as the aligned residue (see Figure 2.8). Unaligned residues receive no annotation.

Even if the input protein's structure is known, *membrane contact* annotation is always transferred using this method.

2.6 Validation

2.6.1 Layer abstraction accuracy

The *membrane contact* annotation provided by iMembrane contains detailed information about each residue's preference to be in contact with either the polar head groups or the hydrophobic tail groups. Residues that are not in direct contact with the lipid tails or head groups are simply labelled as "not in contact". Their location relative to the membrane cannot be identified using this annotation type. In order to allow the annotation of all residues' position within the membrane, we have created the *membrane layer* model, which simplifies the contact information into a 3-layered membrane slab model (one tail layer and two head layers representing the membrane, plus two non-membrane layers).

Figure 2.9 shows the distribution of membrane contacts within the CGDB proteins, as well as the distribution of membrane-contacting residues assigned to the wrong membrane layer. By "wrong membrane layer" we mean that, for the given protein, the assumption that the membrane is shaped like a flat slab does not hold and the residue in question is in fact in contact with a different part of the membrane than its assigned *membrane layer* would indicate. The number of misclassified residues within a single protein thus gives an indication of how strongly this protein distorts the membrane

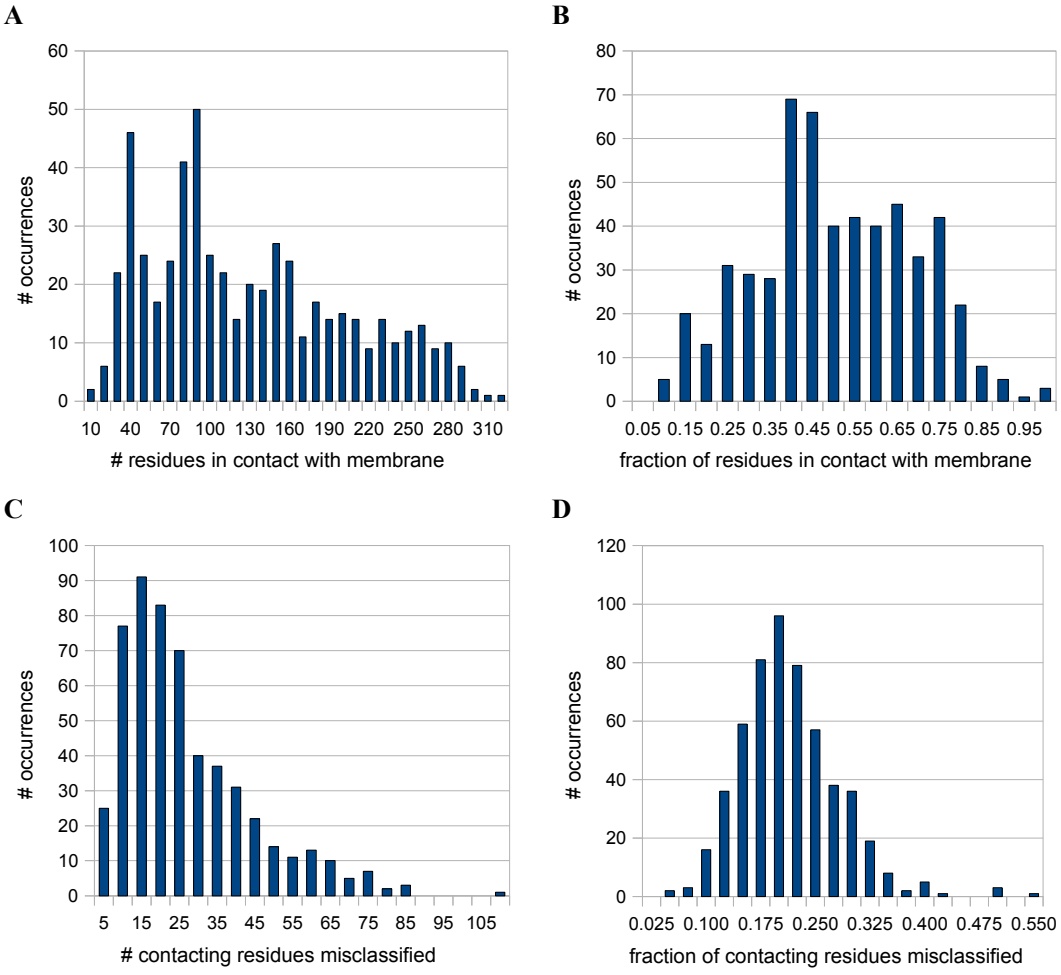


Figure 2.9: Profiles of membrane-contacting residues misclassified by the *membrane layer* annotation - A,B: Residues, per protein chain, in contact with the membrane. **C,D:** Residues, per protein chain, in contact with the membrane and misclassified by the *membrane layer* annotation. The x-axes in **A** and **C** are absolute numbers, **B** is expressed as a fraction of the length of the protein chain, **D** is a fraction of the number of residues in contact with the membrane. The Y axis is always expressed as the absolute number of proteins (within a total set of 542 chains considered).

around it.

The test was performed on 542 transmembrane protein chains present in the iMembrane database (database built on 11 Feb 2011). Seventy nine chains in the database are part of a membrane protein complex but do not, themselves, pass through the membrane tail layer and were thus excluded from this test. The vast majority of proteins were found to distort the membrane bilayer by about 10-15 misclassified residues per protein. Expressed as a fraction of the total number of residues in contact with the membrane, this is between 17.5% and 20% per protein chain.

On average, our *membrane layer* model can approximate the membrane's shape. However, quite a few proteins do appear to locally distort the membrane (52 proteins result in more than 50 misclassified contacts). The largest distortion is seen in protein 1SU4A (Sarcoplasmic/endoplasmic reticulum calcium ATPase; 107 membrane contacts misclassified). Most of the misclassifications, in this case, are caused not by the TM helices but by a soluble domain that touches the membrane head groups and "pulls" at them, forming a bulge on the membrane surface. These residues are classified as being in the "non-membrane" layer, even though the *membrane contact* annotation reflects their interaction with the head groups.

When considering the fraction of membrane-contacting residues being misclassified, the largest distortion is seen in the protein 1OCCQ (bovine heart cytochrome C oxidase). Only 32 residues are in direct contact with the membrane but 53% (17) of them are placed in the "wrong" membrane layer. This particular protein chain only has a single TM helix attached to a soluble domain. It is in contact with other protein chains, forming a large TM protein complex. Other chains in the complex contain soluble domains in contact with the membrane (similar to the case of 1SU4A), and many of the TM helices within the complex have differing orientations relative to the membrane. This results in large distortions of the membrane surrounding the entire protein complex, and this distortion is reflected in the misclassified contacts.

In conclusion, the *membrane layer* annotation alone is a good model of the membrane for proteins that do not strongly distort the membrane. There are a number of cases where such distortions do occur, however, and in these cases the *membrane contact* annotation must be consulted in order to accurately describe each residue's physico-chemical environment. Most existing computational methods, such as OPM and PDB_TM (Tusnády et al., 2005), use a simple slab model of the membrane and are thus prone to the same kinds of errors. With iMembrane, we retain the simplicity of the membrane slab model, while effectively circumventing the problem of membrane distortions using the *membrane contact* annotation.

2.6.2 Homology prediction accuracy

The validity of iMembrane's homology-based approach was verified using a leave-one-out cross-validation. The prediction results for each hit were compared to the original *membrane contact* and *membrane layer* annotation generated directly from the corresponding MD simulation result in the CGDB. A Q3 score was calculated, representing the fraction of residues annotated correctly (as either T, H or N; see figure 2.10).

In the case where the input is a structure, iMembrane's accuracy is consistently good above about 20% sequence identity between the input and the database protein. For sequence input, this threshold is at about 35%. (Note that these identities are always calculated from the structure alignment.) Layer annotation is always better conserved than contact annotation, with the difference being most visible for structure input (see Figure 2.10B&D). In Figure 2.10D we consider the *membrane layer* annotation and the input to iMembrane is a structure. In this case iMembrane can make full use of the input protein's 3D co-ordinates and is guaranteed to annotate every residue, as explained earlier (Section 2.4.3).

Independent of the input type (structure or sequence), a sequence identity of more than 35% tends to result in a Q3 accuracy above 70% and a Q2 accuracy of around

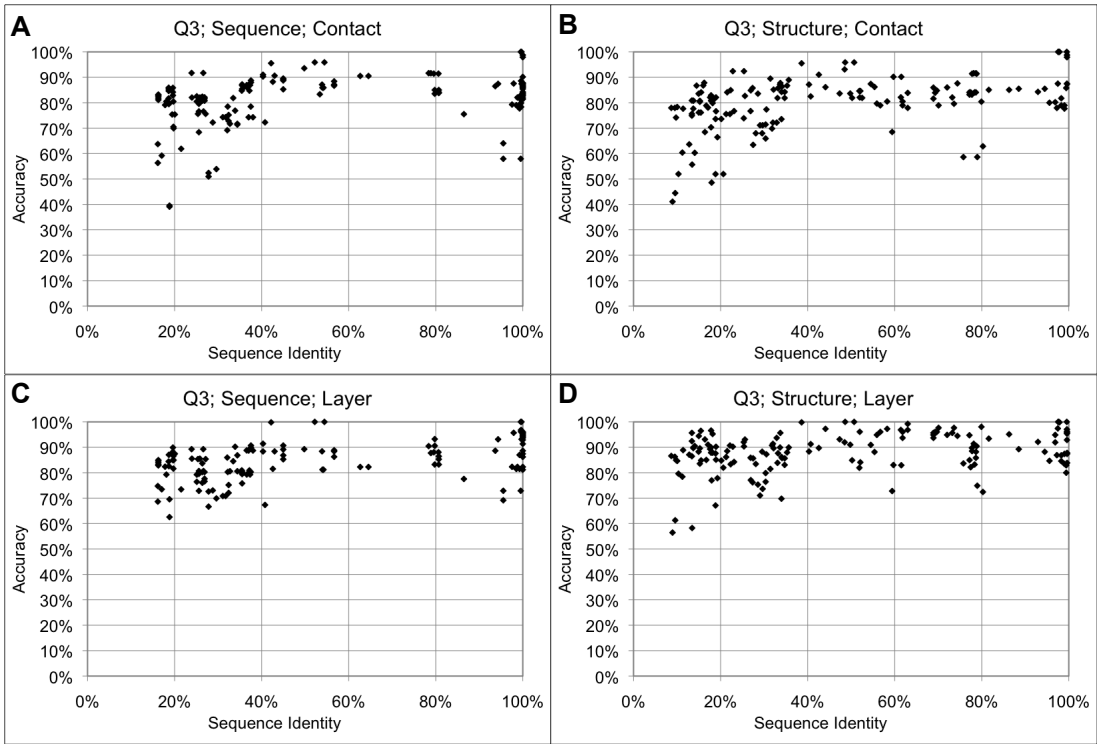


Figure 2.10: iMembrane Q3 accuracy - The X axis shows the percentage sequence identity between the input and database proteins, as calculated from their structure alignment. The Y axis shows the Q3 accuracy, i.e. the percentage of residues annotated correctly by iMembrane as either "N", "H" or "T". **A&B:** *Membrane contact* annotation; **C&D:** *Membrane layer* annotation; **A&C:** Sequence input; **B&D:** Structure input. The Q2 equivalent to this figure can be found in Appendix 7.3.

and above 90% in the membrane layer prediction (see Appendix 7.3 for Q2 accuracy plots). A slight upwards trend can be observed with increasing sequence identity. Below 35% sequence identity, homology detection and sequence alignment quality is known to decline (Rost, 1999). In the case where the input is a sequence, our method depends entirely on the alignment between the query and database proteins; below about 35% identity its accuracy is thus highly variable. For structure input, this boundary is pushed down to 20% sequence identity. The use of improved alignment methods more suitable for distant homologues will benefit the accuracy of iMembrane in future releases. A proof-of-concept implementation of a membrane protein-specific alignment method is described at the end of Chapter 3.

2.7 The iMembrane web application

iMembrane is a valuable source of knowledge for any scientist working with membrane proteins. To make this knowledge as widely accessible as possible, iMembrane is available as a web application with a friendly graphical user interface (www.imembrane.info). A screenshot of the programme's output is shown in Figure 2.11. The iMembrane web server is integrated with the MEDELLER structure prediction server, described in Chapter 4.

2.8 Conclusions

iMembrane is a simple yet powerful new method allowing rapid and easy prediction of a protein's membrane insertion. It is the foundation for the work described in the following chapters, as well as being a useful tool for any biologist or bioinformatician working with membrane proteins. iMembrane has been published in the journal *Bioinformatics* (Kelm et al., 2009). Since its initial publication, some minor changes and updates have been made, which are briefly outlined in a later paper (Kelm et al., 2010).

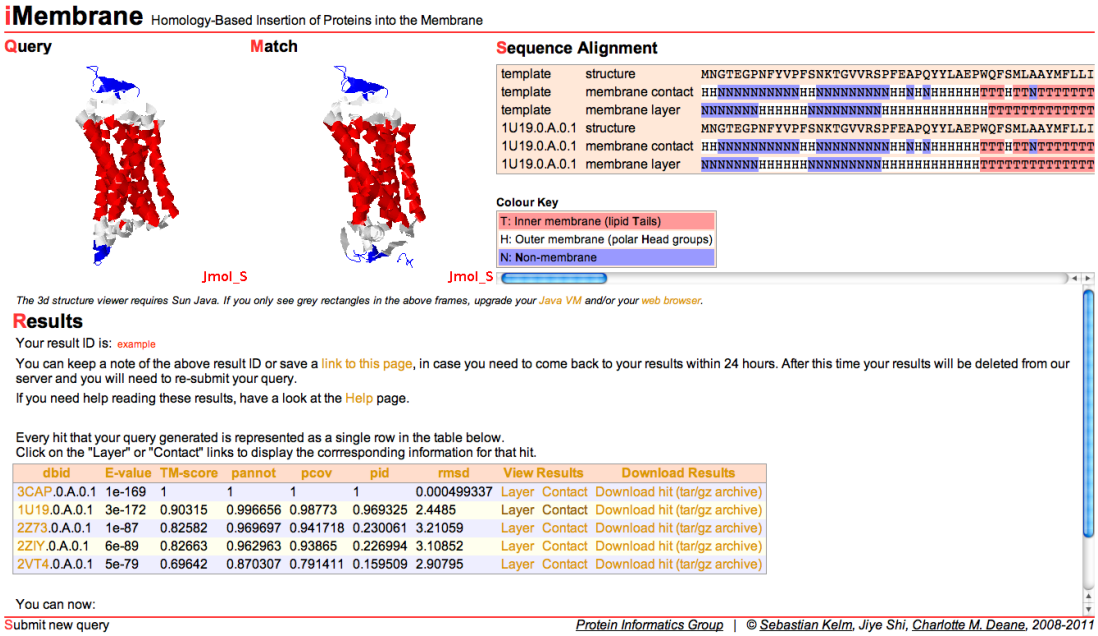


Figure 2.11: Result screen of the iMembrane web application - The query given to iMembrane was the 3D structure of the protein 3CAP, a G-protein coupled receptor (GPCR). Since the protein is in the database, its top database match is identical to the input. For illustration purposes, the second-best hit is shown here, protein 1U19, another GPCR. Top left: cartoon representations of the query and database proteins. Top right: sequence alignment between query (here labelled "template", as this protein will be used as a modelling template later) and database match (labelled by its iMembrane database ID, "1U19.0.A.0.1"). Colours indicate the membrane annotation, as labelled. The lower half of the screen shows the list of top database hits, whose annotation can be viewed by clicking on the corresponding "Layer" or "Contact" link. Links to the corresponding CGDB entries are provided. All results can be downloaded as tar/gzip archives.

The version described here is current at the time of submission of this thesis.

2.9 Future work

iMembrane is built in such a way that its output format is identical to the iMemDB format. It is possible to apply iMembrane to iteratively grow its own database by annotating all proteins homologous to the current database proteins. This could be repeated until some similarity threshold is reached. One must be cautious, however, because the annotation quality is bound to decrease for every new generation of annotated proteins, as the new proteins become more and more distant from the original set of CGDB proteins. Since the effect of such an approach has not yet been tested, iMembrane currently only includes the original set of CGDB proteins in its database.

Future versions of the iMembrane web application may allow the user to browse the iMembrane database, without submitting a query. The database should also be directly downloadable as a single zip file. This would further increase iMembrane's usefulness to the scientific community. In addition, pre-computed annotation may become viewable and downloadable for most membrane proteins of known structure, in a fashion analogous to the OPM and PDB_TM databases.

For sequence input, we currently perform a pairwise sequence alignment with MUSCLE (see Section 1.4.2.2). In future versions we hope to implement a sequence-to-structure alignment method, such as FUGUE (see Section 1.4.4.2), but adapted specifically to membrane proteins. This should raise the annotation quality considerably when the input to iMembrane is a sequence (see Section 2.6.2).

As discussed in Section 2.6.1, membrane proteins can distort the membrane around them, resulting in very specific membrane contact patterns. As the capacity to locally distort the membrane is a property of an entire protein complex, care should be taken when searching iMembrane for single protein chains. A chain that occurs as a

monomer might have a close homolog which is part of a multimeric TM complex. The latter may potentially have very different membrane contact patterns and thus transferring annotation from one to the other may result in annotation errors. Currently, iMembrane only allows searches using a single input chain. In the future, we may implement alternative search strategies, making it possible to search the database for entire protein complexes (e.g. using MM-align, (Mukherjee and Zhang, 2009)), which would essentially eliminate such annotation errors. Furthermore, this would ensure a higher annotation quality, as each chain's annotation would be guaranteed to be compatible with that of the others in the same complex. However, this type of search could result in no database hits being found for some inputs, even if several of the input chains are present in the database. Thus, the current single chain-based annotation scheme should remain available.

2.10 What's next

All the work described in the following chapters makes use of iMembrane. In Chapter 3, iMembrane will be used to investigate the difference in molecular evolution between the various membrane environments (defined by the *membrane contact* and *membrane layer* annotation). The thus identified evolutionary patterns are encoded in environment-specific substitution tables (ESSTs), which are analysed and used for sequence-to-structure alignment of membrane proteins. In Chapter 4, iMembrane and the ESSTs are used in a homology-based structure prediction setting, in order to produce high quality 3D models of transmembrane proteins.

Chapter 3

Environment-Specific Substitution Tables of Membrane Proteins

3.1 Context

There are a multitude of methods that facilitate the structural modelling of soluble proteins but no software has yet been designed specifically to predict membrane protein structure. The previous chapter introduced iMembrane, a method to accurately annotate a membrane protein's membrane insertion. iMembrane allows us to define structural environments within the protein, such as "helix residues in contact with the membrane lipid tails". In this chapter, we use this information to build environment-specific substitution tables (ESSTs). These tables describe the observed patterns of molecular evolution within membrane environments. This information can then be used in membrane-specific alignment methods and, as described in subsequent chapters, for 3D co-ordinate generation.

3.2 Acknowledgements

The work described in this chapter was largely of my own design and, between 2008 and 2009, was performed solely by me. I created an automated pipeline to build the main dataset used in this study and wrote the programme JSUBST to build substitution tables.

In 2010, Jamie R. Hill also began work on ESSTs and membrane protein alignment. The entire analysis and alignment work has been published in (Hill et al., 2011).

Unless otherwise stated, all the work, figures and tables in this chapter are my own. I have included some of Jamie's data for comparison and interest at clearly marked points.

3.3 Motivation

3.3.1 Membrane protein-specific information

Template-based modelling of protein structure is currently the most successful method available. Popular programmes such as MODELLER (see Section 1.4.5.1) use various types of empirical information to restrict the search space of possible structures corresponding to a particular sequence. Generally, the more information available, the more constraints that can be applied and the better the final model.

The first step in template-based modelling is the identification of template proteins of known structure. If more than one template structure is to be used, these first need to be (structurally) aligned to each other. Then, the templates are aligned to the query sequence, using sequence-to-structure alignment (e.g. with FUGUE, described in Section 1.4.4.2). The quality of the initial alignment is a major factor in the accuracy of the final model (Sánchez and Sali, 1997). A detailed description of a typical template-based modelling pipeline was given in Section 1.4.

ESSTs are a widely used way to guide sequence-to-structure alignment. In order

to improve the performance of current modelling methods on membrane proteins, it is essential that we gather the basic information on how substitution behaviour in membrane proteins differs from that in soluble proteins. To this end, we have created membrane protein-specific ESSTs.

3.3.2 Structural environments in membrane proteins

When comparing membrane and soluble proteins, one might assume that buried residues as well as water-accessible residues of both types of proteins should be relatively similar. The main difference should be observable between residues in contact with the membrane, compared to residues in contact with water. However, when using classical "solvent-accessible surface area" annotation, as given by programmes such as JOY (Mizuguchi et al., 1998a) (see Section 1.4.4.1), there is no way to distinguish between these two structural environments. Solvent-accessibility is calculated without using any information about the protein's membrane insertion; it is merely a measure of the surface area of a residue exposed on the outside of the protein. Thus, in membrane proteins, many of the residues labelled "solvent-accessible" (i.e. surface residues) are in fact in contact with the hydrophobic membrane, a radically different physico-chemical environment to the aqueous "solvent".

From the annotation generated by iMembrane (see Chapter 2), it is possible to discriminate between these structural environments. Specifically, we distinguish residues that are predicted to be either: *in contact with the lipid tails (t)*; *in contact with the polar head groups (h)*; *not in contact with the membrane (n)*, i.e. exposed to water. We termed this *"membrane contact"* annotation (Figure 2.4B).

We can also distinguish residues by their position along the bilayer normal, i.e. residues that are: *in the tail layer of the membrane (T)*; *in the head group layer (H)*; *not inside the membrane (N)*. This is called *"membrane layer"* annotation (Figure 2.4C).

Combining those two types of annotation with classical "accessibility" (i.e. which residues are on the surface of the protein), it is also possible to computationally annotate residues that are facing an aqueous pore in the transmembrane protein. All this can be done automatically and requires no human intervention.

In this chapter, we use the annotation produced by iMembrane and JOY to define structural environments specific to membrane proteins. Each such environment is characterised using an ESST. The ESSTs can then be analysed in order to learn more about how amino acid substitution patterns differ between the membrane environments, or between membrane proteins and soluble proteins.

3.3.3 Sequence-to-structure alignment for membrane proteins

Membrane protein ESSTs should not only provide new insights into the molecular evolution of membrane proteins but also have a more immediate practical use. The ultimate aim of this research is to improve structure prediction of membrane proteins. It is widely known that alignment quality is a major factor in determining the quality of template-based models (Sánchez and Sali, 1997). Therefore, we apply our ESSTs to sequence-to-structure alignment of membrane proteins, using the programme FUGUE. We compare alignment performance using our membrane protein-specific tables to the performance of FUGUE's standard ESSTs created from soluble proteins and show that using membrane tables yields markedly improved alignments. In addition, we show the direct effect of a better target-template alignment on the accuracy of the final 3D model of a membrane protein.

3.4 Existing literature

3.4.1 Properties of amino acids in membrane proteins

Membrane protein sequence and structure has been studied for over 35 years. Henderson and Unwin (1975) used electron microscopy to obtain a 3D model of a membrane, containing multiple copies of bacteriorhodopsin (purple membrane protein), a seven-helix bundle, at 7Å resolution. This study was the first concrete proof that transmembrane segments are likely to be helical. Others elaborated on the model (Khorana et al., 1979; Ovchinnikov et al., 1977) with such experiments as protease digestion of water-exposed amino-acids, and by obtaining the protein's primary sequence. Later Henderson et al. (1990) refined the model to 3.5Å resolution, confirming that the protein indeed contained seven largely hydrophobic α-helices.

Deisenhofer et al. (1985) obtained the first high-resolution (3Å) X-ray crystal structure of a membrane protein, that of bacterial photosynthetic reaction centre, containing eleven TM helices. This one high-resolution structure, combined with the growing knowledge of membrane protein sequences, allowed the development of sequence-based methods to predict the location and topology of TM helices. Rees et al. (1989) described the amino acids in the TM region of photosynthetic reaction centre in terms of their hydrophobicity: the interior of the protein was "comparable" to soluble proteins, whereas the bilayer-exposed residues were more hydrophobic and the water-exposed residues more hydrophilic than the interior residues. They proposed a prediction method based on the periodicity of these states in the amino acid sequence. von Heijne (1986, 1992) went on to stipulate the "positive inside rule" which, combined with hydrophobicity information, successfully identified TM helices in 23 out of 24 helical membrane proteins (von Heijne, 1992).

Since then, many more membrane protein structures have been resolved and studied, resulting in a large number of observations. These have been extensively reviewed (von

Heijne, 1994, 1995, 2007).

In 2004, Eyre et al. performed a comprehensive study of all helical transmembrane protein structures known at the time, in order to verify the many previous hypotheses and reevaluate which aspects of membrane protein structure might be useful for prediction purposes. Residue composition and conservation was analysed for a representative set of 23 highly non-redundant (<20% sequence identity) helical transmembrane proteins. The authors used a computational method to automatically place each structure in an artificial membrane, composed of three fixed-width layers (see Figure 3.1). In addition, the study distinguished between surface accessible residues and those buried within the protein (accessible residues could be in contact with either membrane lipids or water). Their method did not allow the computational identification of pore-lining residues and they thus performed this step by manual inspection.

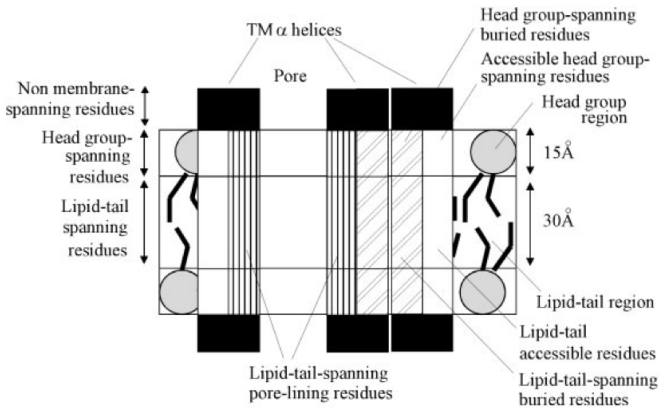


Figure 3.1: Fixed-width membrane model used by Eyre et al. (2004) - The diagram shows three TM helices within a lipid bilayer. The membrane model used here is a simple 3-layered slab, with a central 30Å wide lipid-tail-spanning layer (tail layer) and two peripheral 15Å wide head-group-spanning layers (head layers). The orientation of the protein within the slab is assigned automatically using hydrophobicity of surface exposed side-chains. Pore-lining residues must be identified manually, as a subset of the accessible residues within the membrane tail layer. Figure taken from (Eyre et al., 2004), by permission of Oxford University Press.

Eyre et al. reached a number of conclusions relevant to the work discussed in this chapter, many of which agreed with previous studies on smaller datasets. In the

following paragraphs, the conclusions of Eyre et al. are used as a backbone for a discussion of previous work in the field.

Soluble domains of MPs are similar to globular soluble proteins. Eyre et al. tested this by comparing the average length of secondary structure elements in the soluble domains of MPs to those of globular soluble proteins. The lengths were not significantly different, which was in agreement with previous work (Bowie, 1997; Ulmschneider and Sansom, 2001). In contrast, transmembrane helices were generally found to be longer, in order to be able to span the entire membrane (though occasional exceptions where observed, where a helix only spanned part of the membrane). However, there seemed to be no direct correlation between the length of TM helices and their angle relative to the membrane normal.

As in soluble proteins, buried residues tend to be more conserved than the average residue. It is generally assumed that buried residues tend to be strongly conserved to preserve structural stability (Berg et al., 2002), and membrane proteins seem to be no exception to this rule. This is in agreement with later observations by Mokrab et al. (2010).

Tail layer residues are more conserved than the average residue. Transmembrane domains are subject to stronger evolutionary constraints than soluble domains, requiring specific residue patterns to ensure stability within the particular physico-chemical environment of the membrane. This same trend was observed by Mokrab et al. (2010).

Within the membrane tail layer, large differences are observed between buried and accessible residues. In the tail layer, accessible residues are, on average, significantly more hydrophobic than buried residues. Accessible tail layer residues are also less conserved than buried ones. It is assumed that buried residues in the tail layer tend to be more strongly conserved to preserve structural stability, while lipid tail-contacting residues need to preserve their hydrophobicity to retain favourable in-

teractions with the hydrophobic lipid tails. In addition to the basic hydrophobicity requirement, tail-contacting regions seem to be enriched in aromatic residues, possibly indicating their role in anchoring the protein within the membrane (Yuen et al., 2000).

In the membrane head layer, differences between buried and exposed residues are observed, but are less obvious than in the tail layer. As opposed to the tail layer, accessible residues in the head layer tend to be less hydrophobic than buried residues. Similar to the tail layer, buried residues in the head layer are more conserved than accessible ones, but this difference is less pronounced. Eyre et al. concluded that purely sequence-based methods should therefore focus on predicting only the tail-spanning region.

Pore-lining residues (in the tail layer) show similar patterns to buried residues in the tail layer. Surprisingly, in terms of hydrophobicity and conservation patterns, pore-lining residues appeared similar to buried tail layer residues. Concerning amino acid composition there was only a slight enrichment in polar and charged residues in the pore (possibly for functional reasons) and a reduction in the number of aromatic residues (possibly due to steric constraints). Eyre et al. ventured that the hydrophobicity of the pore would enable efficient flow of polar substrates through it and prevent strong interactions that could hinder flow. They commented that the strong similarity to buried residues would make it very hard to predict pore-lining residues from sequence alone.

Buried residues in the tail layer are involved in helix-helix interactions. In soluble proteins, there are standard patterns that aid helix-helix packing, such as the leucine zipper (Crick, 1953; Langosch and Heringa, 1998). Such patterns are also observed in membrane proteins. In addition, transmembrane helices interact via another mechanism, rarely seen in soluble proteins (Eilers et al., 2002), which involves patches of small polar amino acids (Eilers et al., 2000; Javadpour et al., 1999; Senes et al., 2000; Ulmschneider and Sansom, 2001).

Polar and charged residues do appear on the lipid-exposed surface (in the tail layer). Many of these lipid-exposed residues were observed "snorkelling" their polar groups, usually at the end of a long side chain, towards the head layer. This was thought to allow the formation of energetically favourable hydrogen bonds (H-bonds) with head groups or water, while leaving only a fairly apolar hydrocarbon chain exposed to lipid tails. Other charged residues were seen to form H-bonds with polar (not necessarily charged) residues one helix turn away. About one quarter of charged surface residues remained unsatisfied and apparently in contact with the membrane lipid tails. This observation may be due to classification errors due to the use of a fixed-width membrane slab model. In a later study, Mokrab et al. (2010) observed a similar pattern on a larger dataset, although their membrane model was essentially identical to that of Eyre et al. and thus prone to the same errors.

3.4.2 Substitution tables of membrane proteins

An introduction to substitution tables has been given in Section 1.1.4.2. As mentioned there, a refinement of the concept of substitution tables are ESSTs. A protein is partitioned such that each residue belongs to one of a predefined set of structural environments. For membrane proteins, these environments might be defined by using a membrane model such as that of Eyre et al. (Figure 3.1). For each such environment a single substitution table is created.

The sequence-to-structure alignment program FUGUE was described in Section 1.4.4.2. It uses a set of 64 ESSTs built from a dataset of soluble protein structures. The 64 ESSTs correspond to 64 structural environments defined by their solvent accessibility, secondary structure and various hydrogen bonding patterns.

In a later study, Mizuguchi et al. (2007) used ESSTs as a means of analysing thermophilic procaryotes and identified two different mechanisms by which thermophilic archaea and thermophilic eubacteria might have evolved their temperature resistance.

A recent study by Mokrab et al. (2010) used a dataset of 42 families of helical membrane protein structures, each of which contained at least one structure and a number of homologous sequences. The families were aligned (one multiple sequence alignment per family) and amino acid substitutions were counted, in order to build ESSTs. The method used to count substitutions is similar to the one used in this chapter (see Section 3.5.2) and the authors mentioned possible applications in sequence-to-structure alignment.

3.5 Materials and Methods

3.5.1 Creating the JSUBST datasets

3.5.1.1 Membrane protein dataset

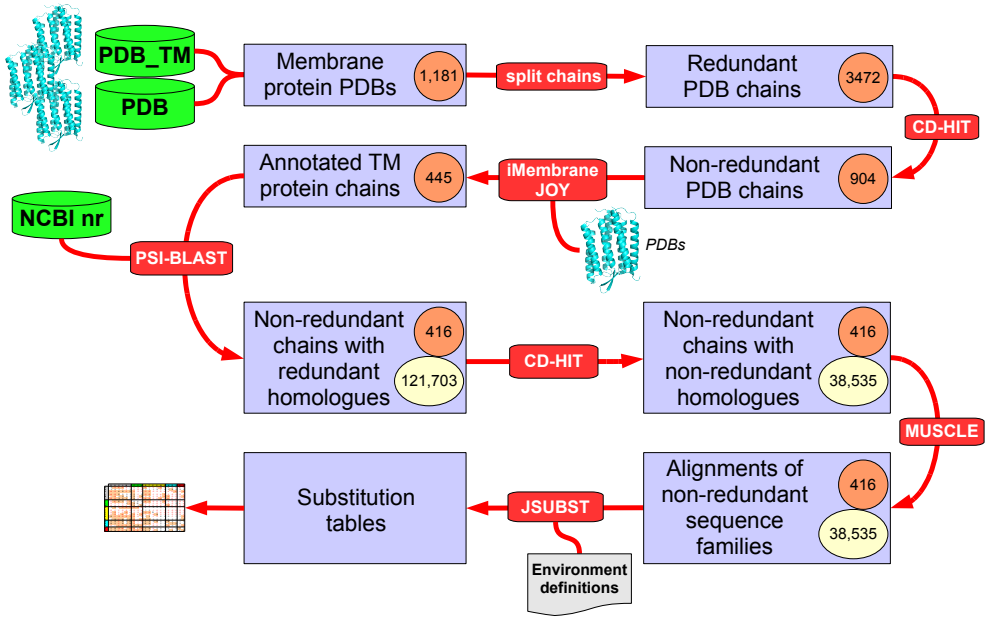


Figure 3.2: Creation of the dataset for calculating substitution tables - Flowchart of the procedure used to create the dataset for substitution table generation. The orange circles show the number of CGDB protein sequences (or sequence families) kept in each step. The light yellow circles show the total number of sequences in the dataset, including homologs from the NCBI NR sequence database.

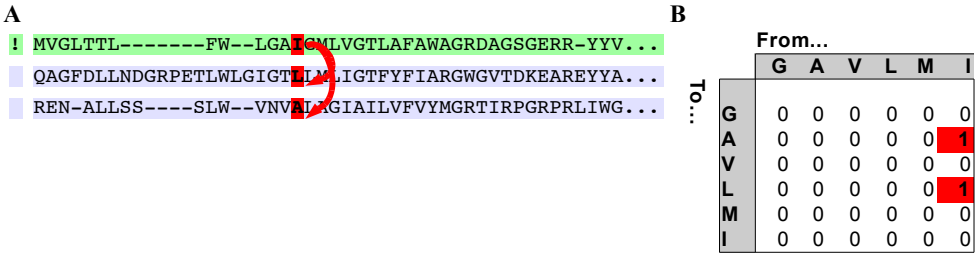


Figure 3.3: Counting substitutions in a multiple alignment - In JSUBST, substitutions can be counted in a multiple sequence alignment between one master sequence (whose structure is known) and any number of sequence homologues (whose structure may not be known). As an example, one substitution is highlighted in A and is counted in B (no other substitutions are counted here, for illustration purposes). **A:** Multiple sequence alignment. The first sequence (marked with a ! sign) is the master sequence, whose structure is known. The other sequences are homologues. **B:** Substitution count table. Only aliphatic residues are shown. Note that such count tables are asymmetric.

Transmembrane protein structures were identified using the PDB_TM database (Tusnády et al., 2005) and downloaded from the PDB (Berman et al., 2000). Each was then split into its component protein chains. The dataset was made non-redundant at the 80% sequence identity level using CD-HIT (Li and Godzik, 2006).

Each protein chain in the dataset was annotated using JOY (*accessibility* and *secondary structure* annotation) and iMembrane (*membrane contact* and *membrane layer* annotation). Proteins without any iMembrane hit (i.e. those not structurally similar to any protein in the iMembrane database) were removed from the dataset. This had the desirable side-effect of eliminating any protein chains not part of a biological membrane protein complex, such as antibody (F_{ab}) fragments, which are commonly used to facilitate the crystallisation of membrane proteins.

For each chain, related sequences were obtained from 5 iterations of PSI-BLAST (Altschul et al., 1997) runs against the NR database (Sayers et al., 2009). NR is a non-redundant database of known protein sequences. The E-value thresholds used for the PSI-BLAST search were 10^{-3} for keeping a hit, and 10^{-5} for including a hit in the sequence profile of the next iteration. Protein chains gaining no PSI-BLAST hits were discarded (mostly single helices of <30 amino acids).

Every "family" of homologous sequences was made non-redundant at the 80% sequence identity level using CD-HIT. Then, sequences were aligned to their corresponding structure using the MUSCLE (Edgar, 2004a) multiple sequence alignment method.

Finally, the remaining 416 annotated multiple alignments were fed into JSUBST to count amino acid substitutions. An overview of the entire procedure up to this point is shown in Figure 3.2.

3.5.1.2 Soluble protein dataset

Using the same method, an equivalent dataset was built for soluble proteins. It contains 423 soluble protein structures, taken from the HOMSTRAD (Mizuguchi et al., 1998b) database, each aligned to a family of homologous sequences.

HOMSTRAD contains structure alignments of proteins from homologous families. Known protein structures are collected from the Protein Data Bank (PDB) (Berman et al., 2000) on the basis of their quality and resolution and aligned using the COMPARER (Sali and Blundell, 1990) structure alignment method. The alignments are then subjected to manual refinement and annotated in terms of each protein's structural environment using the JOY (Mizuguchi et al., 1998a) program. The protein families in HOMSTRAD are manually selected by human curators, such that all family members share a common evolutionary origin. This is done on the basis of sequence and structure similarity.

3.5.2 Counting substitutions with JSUBST

JSUBST calculates substitution tables from an annotated sequence alignment as produced by the programme JOY. JSUBST can handle (annotated) structure input as well as (unannotated) sequence input.

Counting substitutions. The input to JSUBST is a multiple sequence alignment, where the structure of at least one of the proteins is known. In addition, JSUBST ex-

pects a definition of the possible structural environments, for each of which an ESST will be built (e.g. accessible/inaccessible and helix/sheet/coil/positive phi; resulting in $2 \times 4 = 8$ possible environments, and thus 8 ESSTs).

For each column within the alignment, substitutions are counted between each protein of known structure and all other proteins (see Figure 3.3). Let A_{ab}^{E} be the number of observations of amino acid a being substituted by amino acid b in structural environment E. If a and b in their respective environments $E(a)$ and $E(b)$ are observed in the same alignment column, then both counts $A_{ab}^{E(a)}$ and $A_{ba}^{E(b)}$ are increased by one.

As a special case, if b comes from a protein where only sequence information (and no structure) is known, $E(b)$ is undefined and only $A_{ab}^{E(a)}$ is incremented. This is generally the case for our datasets, where only one of the aligned sequences has a known structure. Therefore, tables created this way are asymmetric.

A set of 20×20 matrices, each describing the raw (unweighted) substitution counts observed in one environment in the input alignment, are created. The sum of all environment matrices is called the environment-independent (or TOTAL) counts matrix and describes the observed substitution counts within the entire protein.

If more than one input alignment is given, this entire procedure is repeated for each alignment and the counts in each environment are summed over all input alignments.

Clustering and weighting. The procedure described above treats every protein within an alignment equally. If large groups of highly similar proteins are present within the alignment, this will bias the collected information towards these groups of proteins. To avoid such bias, alignments are clustered before counting substitutions. Using a standard single linkage algorithm, the alignment is broken up into clusters of proteins with pairwise sequence identities less than a user-adjustable cut-off value (<60%, by default). This procedure is similar to that used in the calculation of BLOSUM matrices (Henikoff and Henikoff, 1992). Then, substitutions are counted between pairs of clusters, rather than pairs of sequences, and each individual substitution is weighted

by a factor of $\frac{1}{nm}$, where n and m are the numbers of sequences in the two clusters.

Calculating observed substitution frequencies. Raw (weighted) substitution counts are converted to substitution frequencies as follows:

$$P(b|a,E) = \frac{A^E_{ab}}{\sum_c A^E_{ac}} \tag{3.1}$$

where $P(b|a,E)$ is the frequency of amino acid a in environment E being substituted by amino acid b. Note that $\sum_c A^E_{ac}$ is the sum of substitution counts from a to all possible amino acids, in environment E.

Calculating log-odds scores. Frequency matrices are converted to log-odds matrices as follows:

$$s(a, E \rightarrow b) = log\frac{P(b|a,E)}{q_b} \tag{3.2}$$

$$q_b = \frac{\sum_{a,E} A^E_{ab}}{\sum_{a,c,E} A^E_{ac}} \tag{3.3}$$

where $s(a, E \rightarrow b)$ is the log-odds score for amino acid a in environment E being substituted by amino acid b, and q_b is the background probability of observing amino acid b. Log odds scores are multiplied by a scaling factor of $3/log(2)$ and rounded, as in the original SUBST. (Scaling factors vary between different tables, e.g. BLOSOM62, as used in NCBI BLAST, uses a factor of $log(2)/2$.)

In Shi et al. (2001) a prior distribution of amino acid substitution probabilities is combined into a weighted sum with the frequencies observed from the input alignment. Then a smoothing algorithm is run. This smoothing procedure was originally developed to deal with sparse data. However, we found that its effect was negligible, when using a large enough dataset. The only (small) visible effect of the smoothing procedure seemed to be on tables relating to the "positive phi angle" secondary structure class,

which is naturally sparse and mainly involves glycine-to-glycine substitutions. On our datasets, activating smoothing appeared to be slightly detrimental to alignment quality (Jamie R. Hill, personal communication). We therefore opted not to use this smoothing procedure.

3.5.3 Definition of structural environments

ESSTs differ from traditional environment-independent tables in that they do not treat the entire protein as a single homogenous entity. Instead, the protein is partitioned into several smaller "structural environments", whose amino acid substitution patterns are characterised independently of each other. A simple example would be to separate residues by their *accessibility*, i.e. they are either on the protein surface (accessible) or buried within the protein (inaccessible). In this example, we use only a single environment factor (annotation type), namely *accessibility*, to define our structural environments. We would thus create two ESSTs, one for "accessible" residues and one for "inaccessible" residues. More complex environments can be constructed, and larger sets of ESSTs created, by combining several environment factors. An overview of the environment factors used in this work, and their possible values, is shown below:

- Membrane contact
  - in contact with lipid tails (t)
  - in contact with head-group (h)
  - not in contact with membrane (n)
- Membrane layer
  - membrane tail layer (T)
  - membrane head-group layers (H)
  - non-membrane layers (N)
- Secondary structure

  - α-helix (X)
  - extended β-sheet (E)
  - coil / loop (C)
  - positive phi angle (P)
- Accessibility
  - accessible (A)
  - inaccessible / buried (a)

For all possible combinations of environment factors, one set of ESSTs is created. That is, the *membrane layer* environment factor by itself, *membrane layer* combined with *membrane contact*, *membrane layer* combined with *membrane contact* and *secondary structure*, etc. The number of possible combinations of environment factors is thus defined by the term:

$$C_{factors} = \sum_{k=1}^{n} \binom{n}{k} = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!} \tag{3.4}$$

where n is the number of total environment factors (in this case 4) and k is the number of environment factors used to build a single set of ESSTs. We thus build 15 sets of ESSTs:

$$C_{factors} = \sum_{k=1}^{4} \binom{4}{k} = \binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 4 + 6 + 4 + 1 = 15 \tag{3.5}$$

The number of ESSTs in a single such set depends on the number of values each of its defining environment factors can take. For example, a set of tables created using the factors *membrane contact* and *secondary structure*, would contain 12 ESSTs (3 possible values for *membrane contact* $\times$ 4 possible values for *secondary structure*). Each ESST is labeled with the values of the environment factors that define it, e.g. one of the ESST would be labeled "tX", which stands for "residues in contact with the lipid tail

groups and inside a helix". In total, I have created 239 ESSTs for membrane proteins, each based on a different structural environment.

For soluble proteins, ESSTs were created analogously (using only *secondary structure* and *accessibility* annotation, as iMembrane annotation is not applicable for soluble proteins). In accordance with Equation 3.4 this resulted in $2+1=3$ sets of tables (just *secondary structure*, just *accessibility* and both combined) totalling $4+2+(4\times 2)=14$ ESSTs for soluble proteins.

3.5.4 Comparing the Substitution Tables

I use Kolmogorov-Smirnov (KS) tests to compare the substitution profiles (i.e. the ESSTs) of two structural environments. For each substitution, a KS test is carried out. This determines whether the distribution of substitution scores (where one score comes from each protein family) for a particular substitution is significantly different in the two environments being compared. By choosing a p-value cut-off, the difference between two matrices can be quantified as the fraction of substitutions with a significantly different score distribution. The procedure is outlined below:

1. The input is a dataset with one multiple sequence alignment (MSA) per protein family (see Figure 3.2). Every multiple alignment contains one master protein, whose structure is known, as well as a series of sequence homologues. Every MSA is annotated with JOY (and iMembrane, in the case of membrane proteins).

2. Based on a given definition of structural environments, create a separate set of substitution tables for every annotated MSA. Note that this is different from the usual way of creating ESSTs, where the information of all MSAs is summarised into a single set of ESSTs.

3. "Overlay" all tables for a specific environment such that every cell has a distribution of values, with each protein family providing one value for each possible

substitution.

4. To compare the substitution of amino acid a to b in two arbitrary structural environments E_1 and E_2, run a two-sample KS test, giving the vector of values of $s(a, E_1 \to b)$ and the vector of values of $s(a, E_2 \to b)$ as the two arguments.

5. A significance level of 0.05 is used to determine whether a particular substitution has a significantly different distribution in the two environments. As this is a multiple testing scenario, the Bonferroni correction (Dunn, 1961) is used to avoid over-estimating the number of significant substitutions, i.e. the p-value cut-off is divided by the number of tests performed (20 x 20 amino acids = 400 possible substitution types = 400 tests).

6. The distance between two tables is defined as the fraction of significantly different substitution types out of all 400 possible ones.

Using this approach, we perform a large number of pairwise comparisons of structural environments. The resulting distances are saved in a distance matrix, which can then be fed into a hierarchical clustering method and visualised as a binary tree. Here, Ward's hierarchical clustering method (Ward, 1963) is used, implemented in the "agnes" function of the R statistical programming package. The resulting binary tree is visualised using the FigTree program[1] which gives a global idea of how the substitution tables are related.

However, before comparing all 253 structural environments (239 in membrane proteins + 14 in soluble proteins) to each other, some filtering was required as several environments make little biological sense and thus contain virtually no observed substitutions (e.g. residues in contact with the membrane tails and in the "not-in-membrane" layer). To simplify matters all tables were compared, using the procedure described above, to the environment-independent "TOTAL" substitution tables. We discarded

[1] FigTree homepage: http://tree.bio.ed.ac.uk/software/figtree

those ESSTs with less than 20 substitutions (i.e. 5% of the possible 400) significantly different from the TOTAL tables. This step filters out tables containing very little data, as well as tables with substitution distributions very similar to the environment-independent tables. After this filtering step, 152 ESSTs for membrane proteins remained (out of the initial 239). For soluble proteins, filtering did not reduce the number of tables, i.e. all 14 soluble tables contain enough data and are significantly different from the TOTAL soluble table.

All 166 remaining ESSTs (including both membrane and soluble protein tables) were then compared to each other. This resulted in a 166×166 distance matrix, which could be visualised using hierarchical clustering methods (see Section 3.6.1).

In addition to such large scale comparisons, it is possible to compare two individual structural environments. First, one ESST (a single table of 20×20 log-odds scores) is subtracted from the other to form a difference matrix (of 20×20 log-odds differences). Then, each cell is shaded in red if its KS test produced a p-value below the chosen cut-off. Rather than simply summarising the difference between two environments in a single number, this approach allows an analysis of *which substitutions* contribute to the difference. An example of this type of comparison is discussed in Section 3.6.2.

3.5.5 Testing ESSTs in sequence-to-structure alignment

The above sections described how we defined a list of structural environments, created ESSTs to describe their evolutionary properties, and compared these ESSTs to each other. Here, we go on to apply one particular set of ESSTs to a real bioinformatics problem: sequence-to-structure alignment. We used our membrane ESSTs as an input to the alignment program FUGUE (described in Section 1.4.4.2), replacing its default ESSTs created from soluble proteins. We then re-optimised gap penalties on a small training set (of membrane protein alignments) and measured the improvement in alignment accuracy on a separate test set. This work was based on ESSTs created from the

same dataset described in Section 3.5.1 but was performed, as part of a separate study (Hill et al., 2011), in large parts by Jamie R. Hill.

3.5.5.1 ESST environment definition for purposes of the FUGUE alignment test

As described in Section 3.5.3, four environment factors were used to define the structural environments for our ESSTs. As each environment factor can take one of several possible values (*membrane contact* [t,h,n]; *membrane layer* [T,H,N]; *secondary structure* [X,E,C,P]; *accessibility* [a,A]), this results in a total of $3 \times 3 \times 4 \times 2 = 72$ environments. As some of these environments make little biological sense and are thus extremely sparsely populated, we combined several of them, resulting in a final 26 well-populated structural environments, each of which is labeled with a 3-letter code, as shown in Figure 3.4. The membrane-related annotations (*membrane contact* and *membrane layer*) are summarised in the first letter of each environment label, by ignoring the *membrane contact* annotation apart from two cases: Pore-lining residues (non-contact, tail layer, accessible) and the "head-tail Interface", where contact and layer annotations disagree ([head contact, tail layer] or [tail contact, head layer]). A visual representation of this summarised membrane annotation is shown in Figure 3.5.

First letter (layer)	**Second letter (secondary structure)**	**Third letter (accessibility)**
H – lipid head	X – helix	A – accessible
N – not in membrane	E – beta strand	a – inaccessible
T – lipid tail	C – coil	
P – pore-lining	P – positive ϕ	
I – interface region		

Figure 3.4: Glossary of environment descriptions for the FUGUE alignment test - All combinations of letters are possible with the exception that Pore-lining and Interface-regions cannot be inaccessible. This is because the definition of these "layers" requires them to have contacts with either the membrane or solvent. Figure adapted from (Hill et al., 2011), by permission of Oxford University Press. "Helix" has been relabelled as 'X' instead of the original 'H', for consistency.

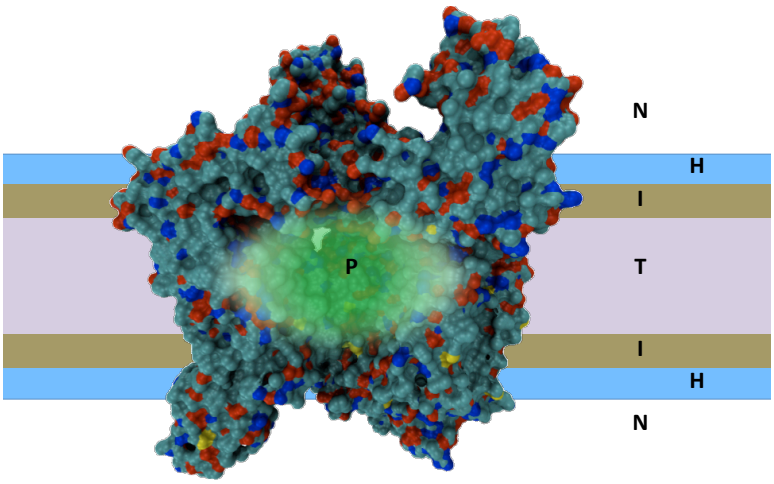


Figure 3.5: Membrane environments used for the FUGUE alignment test - A schematic slice through a membrane protein (1YEW) in the membrane indicating the layer types used. 'N' is the region outside the membrane, 'T' and 'H' span the tail and head groups of the membrane lipids respectively, 'P' is the area lining the pore, and 'I' is the interface region between the tail and head groups, on the protein surface. Figure created by Jamie R. Hill, taken from (Hill et al., 2011), by permission of Oxford University Press.

3.5.5.2 Alignment test set

We collated a set of pairs of homologous membrane proteins both with known structure. These pairs of proteins were structurally aligned using TM-align (Zhang and Skolnick, 2005b). An implicit sequence alignment was extracted from the superposed structures which served as the "correct" alignment.

Our dataset was based on the MEDELLER dataset (explained in detail in Section 4.3.10). Briefly, target proteins were those identified as membrane proteins by SCOP (Murzin et al., 1995), PDB_TM (Tusnády et al., 2005), OPM (Lomize et al., 2006) and CGDB (Scott et al., 2008). All structures were taken from the PDB and filtered to include only X-ray structures with resolution $\leq$3Å. Structures were split into chains and made non-redundant at the level of 80% sequence identity, using CD-HIT (Li and Godzik, 2006). We then searched the iMembrane database for all membrane proteins of similar structure (TM-score $>$ 0.50) to each target. Target proteins, for which no

suitable template was found, were discarded. In addition, cases where the target and template were one and the same (same PDB code and chain identifier) were discarded. This resulted in the MEDELLER dataset, containing 616 target-template pairs of membrane proteins. For the purposes of this alignment test, we performed an additional filtering step using CD-HIT such that, for any single target, no two templates shared more than 80% sequence identity. This left 408 target-template pairs including both α-helical and β-barrel proteins.

We designated one protein, from each pair, to be the "template" of known structure and the other to be the "target" of unknown structure. We discarded all information about the target's structure. Each residue in the "template" protein was annotated with respect to its structural environment using iMembrane (Chapter 2) and JOY (Section 1.4.4.1).

3.5.5.3 Alignment programs and accuracy test procedure

Several alignment programs were run on each of the 408 target-template pairs:

- FUGUE, using its own built-in ESSTs based on soluble proteins ("default FUGUE")
- FUGUE, using our new membrane protein ESSTs ("membrane FUGUE")
- FUGUE, using a bipartite scheme that applies the PHAT (Ng et al., 2000) table, for residues in the tail layer (including pore-lining residues), and the BLOSUM62 (Henikoff and Henikoff, 1992) table, for residues in the head or non-membrane layers
- MUSCLE, a sequence-to-sequence alignment program, given only the target and template sequences as input (with no structural information at all)

The sequence alignment generated by each of these methods was compared to the "correct" alignment generated by TM-align. Residues in the template protein were considered one at a time. If a residue is aligned to the same target residue in both

the "correct" (TM-align) alignment and the alignment being assessed (e.g. membrane FUGUE), it is considered "correctly aligned". Being aligned to a gap in both alignments also counts as "correctly aligned". Gaps within the template protein are counted in the same way as regular residues. Terminal gaps in the template sequence (where the target 'overhangs' the template) are ignored. A schematic of this procedure is shown in Figure 3.6.

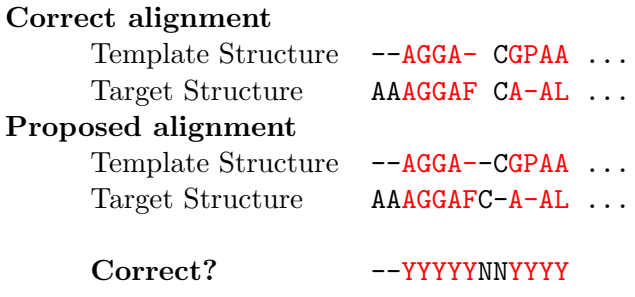


Figure 3.6: Measuring target-template alignment accuracy - The "correct" alignment refers to the structure-to-structure alignment created by TM-align; the "proposed" alignment may be made by any of the FUGUE variants or MUSCLE. In this example 9 residues are correctly aligned, out of a possible maximum of 10 (9 template residues + one internal gap in the "correct" alignment). For clarity, a space was inserted in the "correct" alignment, to make the template sequences line up with each other. Correctly aligned residues are highlighted in red. Note that only the N-terminal end of a larger alignment is shown, and that terminal overhangs are being ignored when assessing correctness. Figure adapted from (Hill et al., 2011), by permission of Oxford University Press.

The number of "correctly aligned" residues is summed over the entire alignment to assess its accuracy. In addition, this measure can be summed over the entire dataset of alignments, or over a subset of alignments in a given sequence identity range (e.g. all alignments between 30% and 35% sequence identity), in order to allow large-scale comparisons between alignment methods.

The methods were compared on a test set of 336 target-template alignments (the total 408, minus the 72 alignments used for gap penalty optimisation).

3.5.5.4 FUGUE gap penalty determination

In any standard sequence alignment algorithm, there is at least one gap penalty (see Section 1.1.4.1). FUGUE uses an "affine gap penalty" scheme (Gotoh, 1982), i.e. it distinguishes between gap opening and gap extension penalties, where the opening penalty is generally more negative than the extension penalty. This reflects the assumption that a single large insertion/deletion event is more likely to occur than many small insertions/deletions. FUGUE has four types of gap penalties, which reflect a residue's position relative to the nearest secondary structure element:

1. Gap within a secondary structure element (H)
2. Gap at the end of a secondary structure element (L)
3. Gap in a loop region (VL)
4. Gap at a terminus (VVL)

Each of these penalties has an opening and an extension variant, making a total of eight penalties, all of which can be modified using command-line arguments.

The value of these gap penalties can have a large impact on alignment quality. As the default values are optimised for use with the built-in default ESSTs, leaving the penalties unchanged may not be a fair comparison. We thus re-optimised all gap penalties for each FUGUE variant, on a small training set of 72 pairs of membrane proteins randomly chosen from our dataset of 408 target-template pairs. The remaining 336 pairs were used as the test set.

During the optimisation procedure, each penalty was perturbed from its default value in integer steps of size 1-5, depending on the size of the initial value. The combination of penalties yielding the highest number of correctly aligned residues, summed over the entire training set, was chosen for each FUGUE variant. For further details, see Hill et al. (2011).

3.5.6 Testing alignments in 3D structure prediction

It is widely known that alignment quality is the major contributor to the quality of template-based models (Sánchez and Sali, 1997). Building models based on our improved membrane protein alignments should thus yield 3D models of higher accuracy. To test this hypothesis I ran my co-ordinate generation program MEDELLER (discussed in Chapter 4) with the various alignments generated as input.

The three alignment methods compared were membrane FUGUE, default FUGUE and the structure-based "correct" alignments created by TM-align. MEDELLER provides a variety of options to prioritise either model quality or coverage, but these had little effect on the relative accuracy of the three methods.

3.6 Analysis

3.6.1 Large-scale comparison of membrane and soluble protein environments

The automated comparison method explained in Section 3.5.4 was applied to all possible pairs of ESSTs of membrane and soluble proteins, yielding a large (166×166) matrix of pairwise distances. The matrix can be used to visualise the clustering of a large number of structural environments (i.e. the ESSTs built to describe these environments). In addition, it is possible to "zoom in" and investigate each pairwise comparison in detail, by examining which substitutions have a significantly different distribution (across protein families) in the two environments being considered. This section deals with the overall clustering of structural environments. An example comparison of two individual matrices is discussed in Section 3.6.2.

Ward's clustering method was run on the matrix of pairwise environment distances. The resulting hierarchical tree is shown in Figure 3.7. From the figure, it is clear that the structural environments (across membrane and soluble proteins) primarily cluster

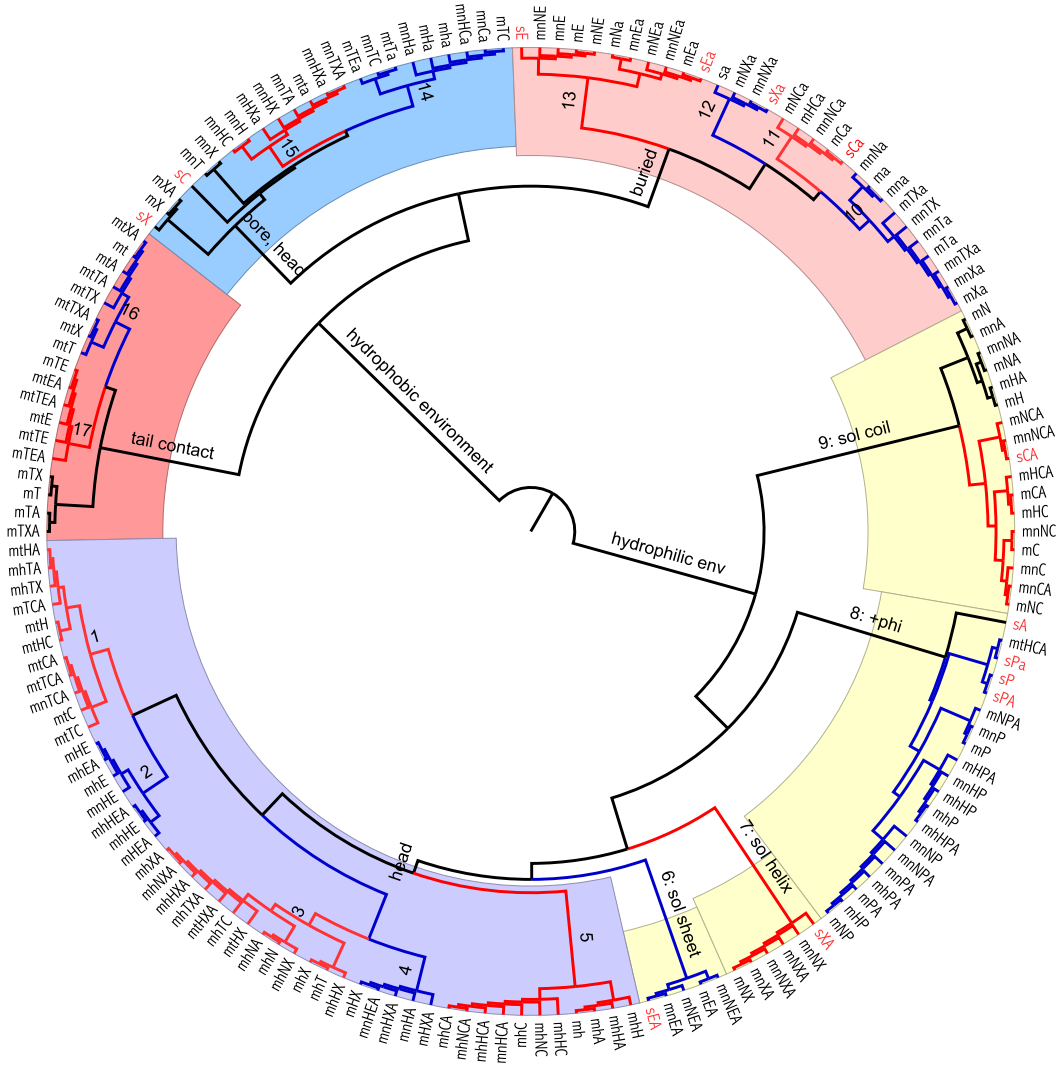


Figure 3.7: Comparison of structural environments in membrane and soluble proteins - Hierarchical clustering tree (Ward's method), based on the KS test distance measure defined in the text. Background colours give a rough grouping of environments, clade colours sub-divide these groupings further. Clade numbers correspond to those used in the text. Black clades contain only very general environments and are thus less interesting. The names of soluble protein environments are coloured red.

into two distinct groups: one made up of strongly hydrophilic, surface-exposed environments, the other comprising mainly hydrophobic lipid-exposed or buried environments along with some mixed environments. These two main clades can be further sub-divided into 5 large groups or 17 smaller clades, as shown below. The clade numbers below correspond to those used in the figure. Throughout the next section, we will refer to the work of Hill et al. (2011), comparing the results between the analysis presented here and that from the paper. Included in this chapter are adapted versions of the PCA plots (Figure 3.8) and dendrogram (3.9) from Hill et al., for comparison purposes.

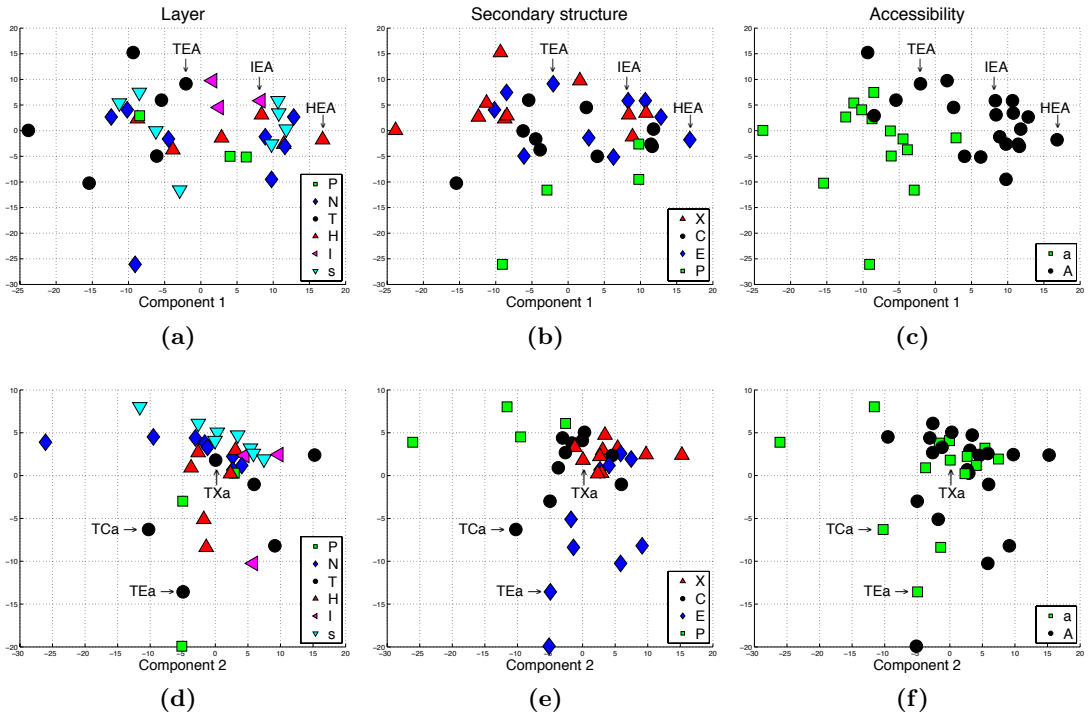


Figure 3.8: Principal component analysis of ESSTs by Hill et al. - The top row and the bottom row are views of the same data along different principal components. The columns colour-code the data-points by layer type, secondary structure, and accessibility respectively. This allows the three-letter table code of each point to be read off from left to right. The labelled tables are ordered by secondary structure in the second principal component – reading panel (e) from left to right we first encounter TCa, then TEa, then TXa. A similar ordering holds for other layer and accessibility types. Figure adapted from (Hill et al., 2011), by permission of Oxford University Press. "Helix" was relabelled 'X', for consistency, instead of the original 'H'.

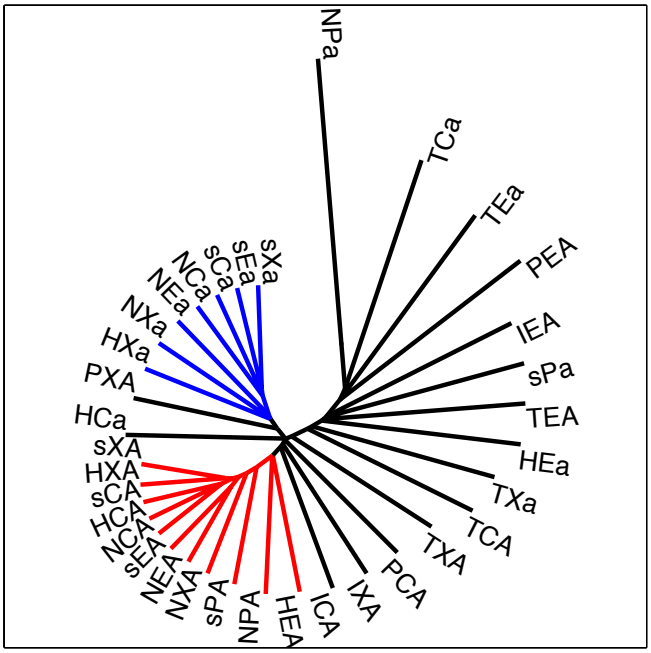


Figure 3.9: Dendrogram of ESSTs by Hill et al. - A split is seen between accessible (red) and inaccessible (blue) environments. Tail-layer environments (T**) appear not to cluster. Figure adapted from (Hill et al., 2011), by permission of Oxford University Press. "Helix" was relabelled 'X', for consistency, instead of the original 'H'.

3.6.1.1 Strongly hydrophilic environments

Surface-exposed residues near the membrane head groups

1. *Head-tail interface and accessible coils nearer the centre of the membrane*

   These residues are positioned at the border between the head and tail layer of the membrane, where the environment is a mixture of hydrophobic and hydrophilic groups. Hydrophobic tail groups are present, and polar side-chains may "snorkel", reaching "up" towards the hydrophilic head groups, possibly pulling them down in turn. Such "snorkeling" was observed in the study of Eyre et al. (2004) for a quarter of hydrophilic residues in the membrane tail layer. This clade contains coil/loop residues in the tail layer of the membrane, which are likely to be at the ends of transmembrane regions and "snorkeling" polar side chains. Thus, this clade contains residues likely to be in contact with polar groups (either head groups or water), but near the hydrophobic lipid tails. In accordance with this interpretation, the PCA plots by Hill et al. (Figure 3.8) indicate that, in terms of hydrophobicity, "interface" residues seem to lie between water or head group accessible residues and tail-accessible residues.

2. *Head contact sheet*

   Beta sheet residues in contact with the head groups cluster with various environments that include them. Hill et al. (2011) showed that this environment seems markedly less hydrophilic than other head layer environments (Figure 3.8).

3. *Head contact helix and head-tail interface helix*

   Head-contacting helix residues (irrespective of layer) dominate this clade. They cluster with membrane-contacting helix residues slightly further towards the centre of the membrane, located at the head-tail interface, which again indicates the occurrence of "snorkeling". Notably, although not unexpectedly, this clade also contains residues in the non-membrane layer, but in contact with the head

groups. Residues outside the membrane and not in contact with the membrane are in a separate clade. This is discussed in more detail further below.

4. *Head layer water-accessible helix*

   Water-accessible helix residues in the head layer (possibly facing an aqueous pore) cluster separately from head-accessible helix residues. In addition, the more general environment "head layer, helix, accessible" [mHXA] clusters here. This is the exact equivalent to the [HXA] environment of Hill et al. (2011), which, in that analysis, was observed to cluster closely with accessible residues in soluble proteins [e.g. sHA] (Figure 3.9). We do not observe this in the present analysis. This is undoubtedly due to the different methods of comparing ESSTs to each other. This is discussed further at the end of this section.

5. *Accesible head layer coil*

   The head layer is a frequent location of coils in transmembrane proteins (the non-membrane layer is more frequent, but may include loops from soluble domains). There seems to be little distinction between coil residues in contact with head groups and those in contact with water, but still within the head layer (these residues could be pore-lining or on the water-exposed face of a large transmembrane domain). This is in contrast with helix residues within the same environment (discussed above), which showed a more distinct separation between head group-accessible and water-accessible residues.

Water-exposed environments outside the membrane

Accessible environments outside the membrane are always grouped with their equivalent in soluble proteins. These findings confirm the often-made previous assumption that soluble domains of membrane proteins behave essentially identically to globular soluble proteins (Bowie, 1997; Eyre et al., 2004; Mokrab et al., 2010; Ulmschneider and Sansom, 2001). This pattern was also observed in Hill et al. (2011), but the separation from

head-accessible environments was less visible (Figure 3.9) because, in that study, the "not-in-membrane" layer also included some head-accessible residues. The present results show that, by excluding such residues, the expected clustering of water-accessible environments emerges even more clearly.

6. *Soluble accessible sheet*

   Water-accessible beta sheet residues outside the membrane are grouped with their counter-parts in soluble proteins.

7. *Soluble accessible helix*

   Water-accessible helix residues outside the membrane are grouped with their counter-parts in soluble proteins.

8. *Positive phi angle (anywhere in the protein)*

   All positive phi tables are grouped together in this single clade. Positive phi tables are generally dominated by substitutions from Glycine to Glycine. Environments in soluble proteins form a separate sub-clade from the membrane protein environments, indicating that, in this case, the origin of the protein itself may be more important than any of the environment factors. This was not observed in Hill et al. (2011), where positive phi tables did not cluster in an easily interpretable fashion (Figure 3.9).

   No positive phi environments are shown for the membrane tail layer, or any buried environment in membrane proteins. This is simply an indication of the scarcity of residues fitting these criteria, as having a positive phi angle means breaking any surrounding α-helix or β-sheet.

   In addition, this clade contains an unexpected environment: accessible coil residues at the head-tail interface. This may be an artefact, due to the low population of this environment. However, one could imagine that, since this is usually the location where TM helices end, there may be an abundance of helix-terminating

Glycine residues (with a positive phi angle) here, dominating this substitution table. It does not explain, however, why this environment clusters with soluble proteins rather than the neighbouring environments in membrane proteins, increasing the chance that this is, in fact, merely noise.

9. *Soluble accessible coil*

   Water-accessible coil residues outside the membrane [mnNCA] are grouped with their counter-parts in soluble proteins [sCA]. A separate sub-clade contains accessible loops in the head layer [mHCA] (which may be in contact with the head groups or water), and several other less specific environments dominated by water-accessible coils (such as "coils not in contact with the membrane", or simply "coils").

3.6.1.2 Hydrophobic environments (buried or in the tail layer)

Buried environments

Buried environments (divided by their secondary structure type) in soluble proteins generally cluster with their equivalents in the soluble domains of membrane proteins. Again, this confirms previous assumptions. The same pattern was observed by (Hill et al., 2011) for helices and coils, although buried sheets did not appear to cluster with their soluble counterparts (Figure 3.9), possibly due to the simpler method (Gong et al., 2009) used for ESST comparison.

10. *Buried tail layer helix*

    Buried helix residues in the tail layer of the membrane do not cluster with buried helix residues outside the membrane (which are located in clade 12). Furthermore, buried residues in the tail layer cluster separately from tail-contacting residues (clade 16) – differences between these two environments have been discussed at length in previous work, in terms of hydrophobicity, amino acid composition, and

conservation (Eyre et al., 2004), as well as individual substitution preferences (Mokrab et al., 2010).

11. *Buried soluble or head layer coil*

    Buried coil residues (no matter in which layer) are grouped with their counter-parts in soluble proteins. A similar pattern was seen in Hill et al. (2011) (Figure 3.9). Buried coils in the membrane tail layer do not appear as a separate environment at all, due to their scarcity (in Hill et al. these do not cluster).

12. *Buried soluble helix*

    Buried helix residues outside the membrane are grouped with their counter-parts in soluble proteins.

13. *Buried soluble sheet*

    Buried beta sheet residues outside the membrane are grouped with their counter-parts in soluble proteins. They are grouped with buried sheets in general, indicating that this type of residue is most common in soluble domains of membrane proteins, rather than inside the membrane (beta barrels contain very few buried residues). Buried sheets in the head layer do not appear as a separate environment, due to their scarcity. Buried sheets in the tail layer appear in clade 14 and are discussed there. Tail-accessible beta sheets appear separately in clade 17.

    A separate sub-clade contains the general "beta sheet" environment for membrane proteins as well as the equivalent for soluble proteins.

Pore-lining or buried in the head layer

14. *Buried coils in the head layer, buried tail layer sheets*

    This clade is dominated by buried coil (loop) residues in the head layer [mnHCa], and less specific environments that overlap with this definition (e.g. buried residues in the head layer [mHa], or tail layer coils [mTC]). These are distinct

from buried residues in other layers (clades 10 and 11) and resemble more closely the pore-lining residues (in clade 15). The head layer contains relatively high concentrations of water. Assuming that these loops have some degree of flexibility, water may occasionally be able to form hydrogen bonds with these residues. In addition, polar side chains may interact with each other. This agrees with the PCA plots by Hill et al. (2011) (Figure 3.8), which show buried coils (in all layers, in fact) to be more hydrophilic than their helix counterparts, and slightly more so than buried coils in soluble proteins.

Buried β-sheet residues in the tail layer also appear in this clade, along with two less specific environments. Such residues are rare, due to the organisation of a typical beta barrel protein, where most residues are surface accessible. Furthermore, buried residues are often in an environment near the aqueous pore at the centre of the protein and are usually of structural or functional importance. They are thus expected to show substitution preferences atypical of other buried or surface residues. In terms of clade organisation, we find a similar pattern as in Hill et al. (2011). Buried sheets in the tail layer cluster close to pore-lining sheet residues (in the neighbouring clade 15) in both the dendrogram (Figure 3.9) and PCA plots (Figure 3.8). They are distinct from tail-exposed sheet residues (clade 17).

15. *Pore-lining helix, head-layer helix*

    Pore-lining helix residues are grouped with various more general environments that include them. In addition, they cluster with buried helix residues in the head layer. This, once again, indicates that polar interactions are more common in the head layer, even for residues annotated as "buried".

    Further to this, we provide additional observations: Pore-lining helix residues cluster in the hydrophobic half of our dendrogram, but not immediately adjacent to the buried tail layer residues. There may thus be important differences in

amino acid substitution patterns between these environments. This is a new discovery, first shown but not discussed in Hill et al. (2011), warranting further investigation. A detailed comparison between these two environments is discussed in Section 3.6.2.2.

Tail contacting

16. *Tail contact helix*

    Helix residues in contact with the membrane tails dominate this clade. They cluster with various more general environments that include them. This agrees with previous studies (Eyre et al., 2004; Mokrab et al., 2010) which showed that tail-contacting helix residues tended to be more hydrophobic but less conserved than buried residues in the tail layer, but still more conserved than residues outside the membrane. A comparison between this environment and water-accessible residues in soluble proteins is discussed in Section 3.6.2.1.

17. *Tail contact sheet*

    Beta sheet residues in contact with the membrane tails dominate this clade. They cluster with various more general environments that include them. This is a new result indicating that findings distinguishing lipid-exposed and buried residues in helical proteins (Eyre et al., 2004) also seem to hold for β-barrel proteins.

3.6.1.3 Summary

As discussed above, our ESSTs, created to represent the various structural environments in membrane and soluble proteins, cluster in a biologically interpretable way. Large differences can be observed between tables that represent chemically distinct environments and the structure of the clustering tree reflects much of the knowledge gained from previous studies. In addition, several new observations of biological interest were made. In summary, one can make the following statements:

- In general, environments in membrane proteins but outside the membrane cluster with their equivalents in soluble proteins. In other words, soluble domains of membrane proteins behave similarly to soluble globular proteins.

- Residues with a positive phi angle seem to behave similarly, independent of their environment, although a subtle distinction can be made based on the type of protein (membrane or soluble).

- Environments within the membrane show very different substitution patterns, which are also clearly distinct from substitution patterns in soluble proteins. They can be divided as shown previously using the four environment factors *membrane contact*, *membrane layer*, *secondary structure* and *accessibility*.

The analysis presented here is based on the same data as that of Hill et al. (2011). However, the latter study was based on a single set of ESSTs built from a slightly reduced set of structural environments. Several environments were combined into one, e.g. the "tail contact, head layer" and "head contact, tail layer" environments were combined into a single head-tail "interface" environment (Figure 3.4). This was done in order to increase the amount of data used to build each ESST, without making the environment definitions too unintuitive.

In contrast, the analysis presented here takes a more exhaustive, automatable approach. All possible combinations of environment factors were used to build ESSTs, and no environments were manually recombined. The clustering presented here was manually interpreted but, in principle, there is no need to do so if the sole goal is to build a reduced set of ESSTs. Conceptually, this table comparison approach could be automated and environments chosen based on the clade structure of the tree. New ESSTs would then be built and formatted appropriately to serve as input to any sequence-to-structure alignment programme, such as FUGUE. This approach would not be restricted to membrane proteins, but could be applied to any pre-annotated set

of proteins.

In terms of results, the present analysis and that of Hill et al. (2011) are in agreement, overall. Several details have become clearer, such as the consistent clustering of environments in soluble proteins with their equivalent non-membrane environments in membrane proteins. Hill et al. observed a general grouping of accessible non-membrane and head layer residues, whereas the present analysis shows a much clearer separation between them. This is due to the simplified set of environments used by Hill et al., which combined all accessible residues in the non-membrane layer, even if they were, in fact, in contact with the membrane head groups. This difference is visible when comparing ESST clustering results, but is unlikely to have a large impact on the alignment results presented by Hill et al., due to the relatively low fraction of residues affected (<5%; Jamie R. Hill, personal communication).

3.6.2 Example environment comparisons

3.6.2.1 Helices in contact with the lipid tails vs accessible helices in soluble proteins

In this section, we compare two example environments which, intuitively, should be very different from each other. Accessible helix residues in soluble proteins (Figure 3.10A) are in contact with the polar solvent, whereas tail-contacting residues within the membrane (Figure 3.10B) are touching strongly hydrophobic hydrocarbon chains.

Looking at each table individually, one can already make out different substitution patterns. On the surface of soluble proteins we observe relatively subtle preferences to conserve amino acid type, and somewhat stronger preferences to conserve polar or charged amino acids. In the tail-accessible environment in membrane proteins, the whole block of aliphatic-to-aliphatic substitutions is favoured, apart from substitutions to glycine. This agrees with and confirms previous observations (Eyre et al., 2004) that the tail-accessible environment is especially hydrophobic and the main evolutionary

3. ENVIRONMENT-SPECIFIC SUBSTITUTION TABLES OF MEMBRANE PROTEINS

A: sXA – accessible helix in soluble proteins

	G	A	V	L	M	I	F	Y	W	S	T	C	P	N	Q	K	R	H	D	E
G	5	-2	-5	-5	-5	-6	-5	-5	-6	-2	-4	-2	-4	-3	-4	-4	-5	-4	-4	-4
A	1	4	1	-1	-1	0	-2	-2	-2	1	1	1	0	0	0	0	0	0	0	0
V	-4	-2	4	0	-1	2	-1	-3	-2	-3	-2	-2	-3	-5	-4	-4	-4	-4	-6	-4
L	-3	-2	2	5	3	3	1	-1	0	-3	-2	-1	-3	-4	-3	-3	-2	-3	-5	-4
M	-2	-1	0	3	7	2	1	-1	0	-2	0	-1	-3	-2	-2	-2	-2	-2	-3	-3
I	-5	-3	3	2	0	5	0	-2	-1	-4	-2	-2	-4	-4	-5	-4	-4	-4	-7	-5
F	-5	-4	-1	0	0	-1	8	4	2	-4	-3	-2	-4	-5	-5	-5	-5	-2	-6	-6
Y	-3	-3	-1	-1	0	-1	5	10	3	-3	-3	-1	-4	-3	-3	-3	-3	2	-5	-4
W	-3	-4	-3	-1	-2	-1	3	3	13	-4	-4	-4	-5	-5	-6	-5	-4	-2	-5	-5
S	1	1	-2	-3	-2	-3	-3	-3	-4	4	1	0	0	0	0	-1	-1	-1	-1	-1
T	-1	-1	-1	-2	-2	-2	-3	-3	-4	0	5	-1	-2	-1	-1	-2	-2	-2	-2	-2
C	-3	-2	-1	-3	-2	-3	-3	-4	-4	-3	-3	12	-4	-4	-5	-6	-5	-4	-6	-7
P	-3	-2	-3	-4	-3	-4	-3	-4	-2	-1	-3	-6	9	-3	-4	-3	-3	-2	-2	-2
N	0	-1	-3	-3	-2	-3	-3	-2	-4	1	1	-2	-2	6	0	0	-1	1	1	0
Q	1	2	-1	0	1	0	-2	-2	-1	2	1	0	0	2	6	2	2	2	2	3
K	1	1	0	0	0	0	-2	-2	-2	2	1	0	0	2	3	6	4	1	1	2
R	0	1	-1	0	1	0	-2	-1	-1	1	0	0	-1	1	2	4	7	2	0	1
H	-2	-1	-1	-2	-1	-2	0	3	-1	-1	-1	-2	-2	1	0	-1	0	9	-1	-1
D	1	0	-3	-4	-2	-4	-4	-3	-4	1	0	-2	0	2	1	0	-1	0	6	3
E	1	2	0	-1	0	-1	-2	-1	-1	2	1	-1	1	2	3	2	1	1	4	6

B: mtTXA – tail-contacting helix

	G	A	V	L	M	I	F	Y	W	S	T	C	P	N	Q	K	R	H	D	E
G	5	0	-3	-4	-4	-3	-3	-5	-5	0	-2	-1	-4	-3	-5	-3	-6	-2	-5	-5
A	2	4	0	-1	0	-1	-1	-3	-2	2	1	1	-2	-1	-2	-2	-4	-4	-1	-2
V	0	2	5	2	2	4	1	-1	-1	0	2	2	-2	-5	-3	-4	-4	-4	-6	-2
L	1	2	3	5	4	3	3	1	1	1	1	2	-2	-3	-1	-2	0	-3	-5	-2
M	-1	1	2	2	8	2	2	0	0	1	1	2	-4	1	0	-1	-1	-3	-3	-3
I	1	2	5	4	3	6	2	0	0	1	2	2	-1	-2	-3	-2	-4	-3	-5	-4
F	0	1	2	3	2	2	8	5	3	0	1	2	-1	-5	-3	-3	-2	-3	-5	-7
Y	-5	-3	-3	-3	-3	-4	0	9	2	-2	-4	-2	-5	-1	-4	-2	-2	4	-2	-6
W	-2	-1	-2	0	0	-1	3	7	13	-1	-2	-4	-2	-1	-3	-1	0	-2	-6	-8
S	-1	-1	-4	-4	-3	-4	-5	-5	-5	4	1	0	-5	1	-2	-1	-4	-5	-1	0
T	-2	-1	-1	-3	-1	-3	-3	-4	-3	2	5	0	-4	-2	-1	-2	-3	-7	-1	-1
C	0	2	1	0	1	1	0	-2	-3	1	3	11	-4	-4	-6	-5	-4	-3	-11	1
P	-3	-5	-6	-6	-8	-8	-6	-7	-3	-3	-7	-6	11	-5	-3	-1	-3	-8	-4	-5
N	-7	-7	-10	-9	-7	-10	-10	-7	-8	-5	-6	-6	-5	7	-2	0	-2	-1	1	0
Q	-5	-5	-8	-8	-5	-10	-10	-6	-8	-4	-6	-9	-4	2	8	2	1	0	3	3
K	-7	-6	-9	-9	-7	-11	-10	-5	-4	-4	-5	-14	-3	2	4	8	3	-1	4	3
R	-7	-8	-10	-8	-8	-11	-8	-4	-5	-4	-7	-10	-3	1	2	3	9	-2	0	1
H	-7	-6	-10	-8	-6	-11	-7	-2	-6	-5	-7	-12	-7	2	-1	0	-1	13	2	-2
D	-7	-7	-12	-12	-10	-12	-15	-11	-5	-5	-5	-13	-9	1	-1	-2	-2	-3	7	3
E	-6	-6	-10	-10	-9	-11	-12	-8	-10	-5	-5	-12	-11	3	3	1	-1	-4	4	8

C: difference A-B

	G	A	V	L	M	I	F	Y	W	S	T	C	P	N	Q	K	R	H	D	E
G	0	-2	-2	-1	-1	-3	-2	0	-1	-2	-2	-1	0	0	1	-1	1	-2	1	1
A	-1	0	1	0	-1	1	-1	1	0	-1	0	0	2	1	2	2	4	4	1	2
V	-4	-4	-1	-2	-3	-2	-2	-2	-1	-3	-4	-4	-1	0	-1	0	0	0	0	-2
L	-4	-4	-1	0	-1	0	-2	-2	-1	-4	-3	-3	-1	-1	-2	-1	-2	0	0	-2
M	-1	-2	-2	1	-1	0	-1	-1	0	-3	-1	-3	1	-3	-2	-1	-1	1	0	0
I	-6	-5	-2	-2	-3	-1	-2	-2	-1	-5	-4	-4	-3	-2	-2	-2	0	-1	-2	-1
F	-5	-5	-3	-3	-2	-3	0	-1	-1	-4	-4	-4	-3	0	-2	-2	-3	1	-1	1
Y	2	0	2	2	3	3	5	1	1	-1	1	1	1	-2	1	-1	-1	-2	-3	2
W	-1	-3	-1	-1	-2	0	0	-4	0	-3	-2	0	-3	-4	-3	-4	-4	0	1	3
S	2	2	2	1	1	1	2	2	1	0	0	0	5	-1	2	0	3	4	0	-1
T	1	0	0	1	-1	1	0	1	-1	-2	0	-1	2	1	0	0	1	5	-1	-1
C	-3	-4	-2	-3	-3	-4	-3	-2	-1	-4	-6	1	0	0	1	-1	-1	-1	5	-8
P	0	3	3	2	5	4	3	3	1	2	4	0	-2	2	-1	-2	0	6	2	3
N	7	6	7	6	5	7	7	5	4	6	7	4	3	-1	2	0	1	2	0	0
Q	6	7	7	8	6	10	8	4	7	6	7	9	4	0	-2	0	1	2	-1	0
K	8	7	9	9	7	11	8	3	2	6	6	14	3	0	-1	-2	1	2	-3	-1
R	7	9	9	8	9	11	6	3	4	5	7	10	2	0	0	1	-2	4	0	0
H	5	5	9	6	5	9	7	5	5	4	6	10	5	-1	1	-1	1	-4	-3	1
D	8	7	9	8	8	8	11	8	1	6	5	11	9	1	2	2	1	3	-1	0
E	7	8	10	9	9	10	10	7	9	7	6	11	12	-1	0	1	2	5	0	-2

Figure 3.10: Comparison of two ESSTs: tail-contacting vs soluble accessible - **A:** ESST of the environment "tail-contact, tail layer, helix, accessible" in membrane proteins [mtTXA]; **B:** ESST of the environment "helix, accessible" in soluble proteins [sXA]; **C:** Difference matrix of A-B. Log-odds scores are rounded to the nearest integer, negative numbers are coloured in red. Amino acids are grouped by type as follows: aliphatic (grey), aromatic (green), polar uncharged (yellow), positively charged (blue), negatively charged (red). Individual substitutions are shaded in colour if their respective KS test indicates a difference in log-odds score distributions, across protein families, between the two environments. Yellow shading, $p < 0.05$ (uncorrected); orange shading, $p < 0.05/400$ (Bonferroni corrected for large-scale analysis).

constraint appears to be the conservation of that hydrophobicity.

Figure 3.10C shows a difference matrix. The general trend in this environment comparison is immediately visible by observing the clustering of cells shaded in orange and yellow, for which a KS test indicated differing log-odds score distributions between the two environments. Most of the differences are located in the aliphatic-to-anything section (left) of the matrix. Aliphatic-to-aliphatic substitutions (top left) are clearly favoured by the membrane environment, whereas aliphatic-to-polar/charged substitutions (bottom left) are strongly disfavoured. In addition, the substitution of several polar or charged residues to aliphatic residues (top right) are favoured, when compared to the surface of soluble proteins.

Some concrete example substitutions include:

Valine (V, aliphatic) to Glutamate (E, negatively charged). This substitution is neutral (score 0) in soluble, accessible helix residues. On the other hand, introducing an unpaired negative charge deep inside the hydrophobic membrane is highly unfavourable (score -10). This difference is highly significant ($p = 3.5 \times 10^{-8}$).

Threonine (T, polar uncharged) to Isoleucine (I, aliphatic). This substitution is unfavourable in soluble, accessible helix residues (score -2) but is favourable in the middle of the membrane (score $+2$). This difference is also highly significant ($p = 2.9 \times 10^{-15}$).

3.6.2.2 Buried helices in the tail layer vs pore-lining helices

Eyre et al. (2004) observed that pore-lining residues shared similar properties to buried residues within the membrane tail layer, in terms of hydrophobicity, conservation and amino acid composition. Only a small enrichment of polar and charged residues was observed. The most common residues within the pore were isoleucine, alanine and glycine. Alanine was observed to overall favour being buried over being accessible, but the other hydrophobic residues had no particular preference. Other groups had

3. ENVIRONMENT-SPECIFIC SUBSTITUTION TABLES OF MEMBRANE PROTEINS

A: mnTXa – buried helix in tail layer

	G	A	V	L	M	I	F	Y	W	S	T	C	P	N	Q	K	R	H	D	E
G	9	1	-6	-7	-7	-7	-10	-9	-9	1	-3	-2	-6	-3	-10	-5	-11	-9	-9	-5
A	2	7	1	-3	-1	-2	-4	-2	-4	4	3	2	-2	-1	-1	-8	-4	-5	-5	-2
V	-6	-1	7	1	1	4	-1	-3	-8	-3	0	-1	-5	-4	-4	-4	-4	-7	-8	-4
L	-8	-4	1	7	4	2	1	-3	-1	-6	-2	-3	-9	-7	-1	-2	-6	-5	-10	-4
M	-5	0	2	5	11	5	0	0	0	-3	0	-1	-9	-4	1	0	-4	-1	-4	-3
I	-7	-2	4	3	2	8	1	-4	-3	-4	-1	-1	-9	-6	-4	-2	-8	-6	-10	0
F	-7	-5	-2	1	1	-2	10	4	4	-5	-4	-2	-12	-4	-4	-5	-9	-2	-10	-7
Y	-11	-7	-6	-3	-4	-4	3	12	0	-7	-6	-6	-14	-4	-9	-8	-10	1	-8	-5
W	-7	-10	-4	-5	-5	-2	0	2	15	-9	-10	-13	-31	-1	-11	-14	-9	-6	-11	-25
S	-1	3	-3	-6	-3	-3	-6	-7	-4	7	3	3	-2	1	-2	-5	-6	-4	-3	-2
T	-4	-1	1	-3	-2	-1	-3	-7	-5	2	7	1	-2	-1	0	-10	-6	-3	-5	-4
C	-1	2	1	-1	-2	-1	-3	-1	-6	4	3	13	1	-2	0	-6	-5	1	-3	-1
P	-6	-5	-5	-7	-6	-8	-14	-19	-13	-2	-5	-12	12	-3	-1	-18	-9	6	-8	-18
N	-5	-5	-7	-8	-7	-6	-7	-9	-15	-4	-3	2	-7	10	-5	-3	-4	-3	0	-5
Q	-10	-10	-10	-6	-3	-8	-8	-9	-2	-6	-6	-5	-8	0	11	4	-2	-1	-3	3
K	-13	-15	-16	-15	-14	-17	-18	-14	-6	-12	-10	-15	-15	-13	-5	11	3	-13	-10	-13
R	-15	-11	-19	-12	-22	-16	-13	-9	0	-14	-13	-1	-13	-3	-5	1	11	-1	-9	-9
H	-11	-11	-10	-8	-8	-8	-6	-1	-4	-7	-3	-8	-10	-3	1	0	-4	14	3	-10
D	-11	-12	-12	-15	-13	-15	-13	-11	-14	-9	-6	-11	-4	0	-4	-7	-9	-12	11	-2
E	-13	-10	-13	-11	-11	-11	-15	-14	-16	-9	-10	-16	-12	-3	0	-8	-6	-9	2	11

B: mnTXA – pore-lining helix

	G	A	V	L	M	I	F	Y	W	S	T	C	P	N	Q	K	R	H	D	E
G	8	-1	-6	-6	-7	-7	-7	-8	-9	1	-3	-4	-3	-5	-4	-6	-6	-4	-5	-5
A	1	6	0	-2	-3	-1	-4	-5	-4	2	0	0	0	-4	-2	-2	-3	-6	-3	-3
V	-3	1	6	1	0	3	-1	-3	-7	-4	1	-1	-1	-3	-3	-6	-6	-11	-7	-4
L	-4	0	2	6	3	3	2	-2	-3	-4	-1	-3	0	-4	-3	-3	-3	-9	-8	-5
M	-3	1	3	4	10	2	3	-3	-3	-2	0	-1	-2	1	0	0	-2	-9	-6	-5
I	-4	0	4	3	3	7	1	-3	-5	-5	0	-2	-1	-4	-3	-2	-5	-12	-8	-5
F	-3	-2	-1	3	0	0	9	4	2	-4	-2	-3	-8	-2	-5	-3	-6	-6	-9	-6
Y	-4	-7	-6	-3	-3	-3	2	11	2	-6	-6	-5	-6	0	-3	-2	-6	-5	-7	-6
W	-7	-7	-1	-5	0	-2	0	2	15	-10	-4	-3	-7	-8	-8	-5	-7	-8	-11	-10
S	1	2	-3	-5	-4	-3	-5	-4	-6	7	2	1	-1	0	0	-1	-2	-3	-2	-2
T	-3	0	0	-3	-1	-2	-4	-4	-7	1	7	0	-1	0	0	-1	-4	-7	-3	-3
C	1	1	2	1	0	-3	-2	-1	-5	1	3	17	-8	-6	-3	-8	-7	-3	-13	-11
P	-6	-4	-5	-4	-4	-6	-9	-17	-7	-6	-6	-22	10	-4	-5	-10	-9	-10	-3	-5
N	-4	-3	-5	-6	-2	-6	-7	-4	-8	-1	-2	-9	-6	9	0	1	0	-1	1	-1
Q	-4	-2	-6	-3	-3	-5	-3	-2	-5	-1	-1	-11	-4	1	9	3	3	1	3	5
K	-8	-5	-8	-6	-7	-8	-8	-7	-5	-4	-6	-21	-8	0	1	7	2	-5	-1	-1
R	-8	-4	-10	-6	-8	-8	-9	-9	-6	-3	-7	-5	-6	2	0	4	10	0	-2	-1
H	-10	-7	-9	-6	-6	-7	-5	0	-5	-6	-5	-5	-7	1	2	-1	-1	15	-1	-3
D	-7	-5	-6	-11	-9	-8	-11	-9	-5	-2	-6	-11	-6	-1	1	1	-3	-6	9	4
E	-5	-4	-6	-7	-6	-4	-9	-7	-9	-2	-4	-13	-7	-1	3	2	-1	-5	6	9

C: difference A-B

	G	A	V	L	M	I	F	Y	W	S	T	C	P	N	Q	K	R	H	D	E
G	1	2	0	-1	0	0	-3	-1	0	0	0	2	-3	2	-6	1	-5	-5	-4	0
A	1	1	1	-1	2	-1	0	3	0	2	3	2	-2	3	1	-6	-1	1	-2	1
V	-3	-2	1	0	1	1	0	0	-1	1	-1	0	-4	-1	-1	2	2	4	-1	0
L	-4	-4	-1	1	1	-1	-1	-1	2	-2	-1	0	-9	-3	2	1	-3	4	-2	1
M	-2	-1	-1	1	1	3	-3	3	3	-1	0	0	-7	-5	1	0	-2	8	2	2
I	-3	-2	0	0	-1	1	0	-1	2	1	-1	1	-8	-2	-1	0	-3	6	-2	5
F	-4	-3	-1	-2	1	-2	1	0	2	-1	-2	1	-4	-2	1	-2	-3	4	-1	-1
Y	-7	0	0	0	-1	-1	1	1	-2	-1	0	-1	-8	-4	-6	-6	-4	6	-1	1
W	0	-3	-3	0	-5	0	0	0	0	1	-6	-10	-24	7	-3	-9	-2	2	0	-15
S	-2	1	0	-1	1	0	-1	-3	2	0	1	2	-1	1	-2	-4	-4	-1	-1	0
T	-1	-1	1	0	-1	1	1	-3	2	1	0	1	-1	-1	0	-9	-2	4	-2	-1
C	-2	1	-1	-2	-2	2	-1	0	-1	3	0	-4	9	4	3	2	2	4	10	10
P	0	-1	0	-3	-2	-2	-5	-2	-6	4	1	10	2	1	4	-8	0	16	-5	-13
N	-1	-2	-2	-2	-5	0	0	-5	-7	-3	-1	11	-1	1	-5	-4	-4	-2	-1	-4
Q	-6	-8	-4	-3	0	-3	-5	-7	3	-5	-5	6	-4	-1	2	1	-5	-2	-6	-2
K	-5	-10	-8	-9	-7	-9	-10	-7	-1	-8	-4	6	-7	-13	-6	4	1	-8	-9	-12
R	-7	-7	-9	-6	-14	-8	-4	0	6	-11	-6	4	-7	-5	-5	-3	1	-1	-7	-8
H	-1	-4	-1	-2	-2	-1	-1	-1	1	-1	2	-3	-3	-4	-1	1	-3	-1	4	-7
D	-4	-7	-6	-4	-4	-7	-2	-2	-9	-7	0	0	2	1	-5	-8	-6	-6	2	-6
E	-8	-6	-7	-4	-5	-7	-6	-7	-7	-7	-6	-3	-5	-2	-3	-10	-5	-4	-4	2

Figure 3.11: Comparison of two ESSTs: tail-layer buried vs pore-lining - A: ESST of the environment "non-contact, tail layer, helix, buried" [mnTXa]; **B:** ESST of the environment "non-contact, tail layer, helix, accessible" [mnTXA]; **C:** Difference matrix of A-B. Log-odds scores are rounded to the nearest integer, negative numbers are coloured in red. Amino acids are grouped by type as follows: aliphatic (grey), aromatic (green), polar uncharged (yellow), positively charged (blue), negatively charged (red). Individual substitutions are shaded in colour if their respective KS test indicates a difference in log-odds score distributions, across protein families, between the two environments. Yellow shading, $p < 0.05$ (uncorrected); orange shading, $p < 0.05/400$ (Bonferroni corrected for large-scale analysis).

previously reported the importance of small residues in TM helix packing, especially alanine, glycine, serine and threonine (Eilers et al., 2000; Javadpour et al., 1999; Senes et al., 2000; Ulmschneider and Sansom, 2001).

We compared the ESSTs corresponding to these two structural environments (Figure 3.11) and found few differences. Both matrices show a similar overall pattern, with a strongly positive diagonal and otherwise mostly negative scores. In the buried environment, non-conservative substitutions (away from the diagonal) are more strongly disfavoured than in the pore-lining environment. Conversely, the conservation (score on the diagonal) of several polar and charged residues is slightly (but significantly, $p < 0.05$) stronger in the buried environment. This may simply be an indication of the stronger evolutionary pressure to avoid inserting unpaired charges in the middle of the protein and thus destabilising its structure.

Off the diagonal, the biggest statistically significant difference seems to be in the substitutions away from glycine or alanine (the first two columns), where scores tend to be less favourable in the buried environment. This is in agreement with the above-mentioned roles of these residues in TM helix packing.

3.7 Applications

3.7.1 Sequence-to-structure alignment

We tested the effect of using membrane-specific tables in sequence-to-structure alignment using the program FUGUE. By default, FUGUE uses a set of ESSTs for soluble proteins, created using SUBST (based on the SUB177 dataset). We replaced these tables with membrane protein-specific ones, re-optimised gap penalties and compared our alignment accuracy.

FUGUE was run, using both table types ("default" and "membrane"), on a test set of 336 protein pairs. In addition we tested a bipartite scheme using the PHAT and

BLOSUM62 tables, as well as a pure sequence-to-sequence alignment method MUSCLE. All alignments were compared to the "correct" structure-to-structure alignment made using TM-align.

As is shown in Figures 3.12 and 3.13, membrane FUGUE clearly outperformed default FUGUE in the critical "twilight zone" of sequence identity (around 20% on our scale). Above 35% identity both approaches did roughly equally well and below 10% both approaches performed poorly.

% Identity	Number of Alignments	*membrane FUGUE vs* default FUGUE		MUSCLE		PHAT/ BLOSUM62	
		Win	Loss	Win	Loss	Win	Loss
0–5	53	12	5	14	4	11	3
5–10	78	32	10	29	8	30	6
10–15	49	36	4	43	1	33	6
15–20	20	10	2	14	0	11	2
20–25	30	8	0	18	0	12	3
25–30	26	4	1	14	0	6	1
30–35	14	1	0	6	0	3	1
> 35	66	1	2	3	1	1	0
Total	336	104	24	141	14	107	22

Figure 3.12: Alignment quality of membrane ESSTs vs other methods - For each sequence identity range, the number of alignments where membrane FUGUE correctly aligns at least 10 more (Win) or 10 fewer (Loss) residues than the named alternative method. For example, in the 10 - 15% sequence identity range membrane FUGUE correctly aligns at least 10 more residues in 36 out of 49 alignments. Figure adapted from (Hill et al., 2011), by permission of Oxford University Press.

3.7.2 3D co-ordinate generation

Reasonable alignments are only achieved from 15% sequence identity upwards, and above 35% alignments differ little between methods (see Figures 3.12 and 3.13). In the 15 - 35% sequence identity range the average root mean square deviations are: 3.4Å (membrane FUGUE), 4.1Å (default FUGUE), 2.0Å (TM-align). The mean sequence identity is 24%. A similar trend is observed on GDT_TS scores. Thus, the better

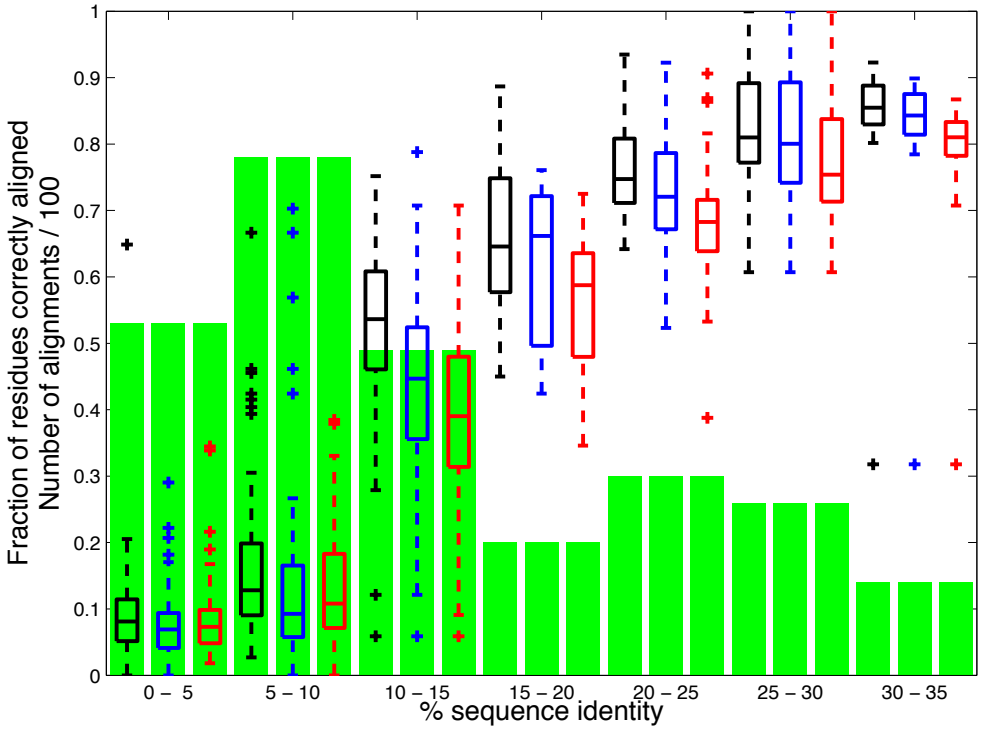


Figure 3.13: Alignment accuracy on membrane proteins of various alignment methods - Box plots of the fraction of residues aligned correctly as sequence identity increases. There are three boxes at each sequence identity, from left to right corresponding to membrane FUGUE (black), default FUGUE (blue), and MUSCLE (red). The green bars show the number of alignments divided by 100. For example, there are 78 alignments in the 5 - 10% sequence identity range. PHAT/BLOSUM62 always performed similarly to default FUGUE and was thus omitted, for clarity. Figure taken from (Hill et al., 2011), by permission of Oxford University Press.

alignments created by membrane FUGUE result in an increase of model quality. These findings were reported in (Hill et al., 2011).

3.8 Conclusion

We have developed a new approach for generating substitution tables of membrane proteins. This required the definition of a useful set of structural environments specific to membrane proteins. Tables representing the molecular evolution within these environments were shown to differ from each other. We have proposed several biologically meaningful interpretations, which confirm and supplement observations in the published literature.

Substitution profiles for membrane environments differ significantly from those observed outside the membrane or in soluble proteins. Molecular evolution thus seems to differ inside and outside the membrane, as well as in the membrane core compared to the outer layers of the membrane.

Using these specialised membrane substitution tables for the structural prediction of membrane proteins should thus improve modelling accuracy.

Guided by the structure of the clustering tree used in our analysis of ESSTs, one may now choose to combine several of the structural environments, in order to reduce the total number of ESSTs and increase the amount of data used to build each of them. This would increase the quality of each individual ESST, while keeping the maximum possible differences between tables. This was done manually in Hill et al. (2011), based on biological intuition and measures of table sparsity and quality. Using the approach presented here, the process could potentially be automated and made applicable to any set of ESSTs.

In addition to the insights gained from our analysis of membrane protein-specific ESSTs, we have demonstrated that the use of such ESSTs in alignment programs can

improve alignment accuracy. Furthermore, the use of such improved alignments in template-based structure prediction directly results in more accurate 3D models.

These positive results are encouraging, especially since many improvements could still be made to our definitions of structural environments, the exact method used to build ESSTs and the alignment algorithm itself. In order to improve the quality of the ESSTs we may, in the future, replace MUSCLE with a sequence-to-structure alignment method, when creating the initial multiple sequence alignments from which ESSTs are built. As sequence-to-structure alignment requires ESSTs to work, one could imagine an iterative alignment and table building procedure, which is continued until convergence.

3.9 What's next

The next chapter will move away from the alignment problem and focus, instead, on the final critical step in a template-based structure prediction pipeline: 3D co-ordinate generation. Given a target-template alignment (which will remain unchanged), membrane protein-specific ESSTs will be used as part of a co-ordinate generation method, MEDELLER, to make decisions about which co-ordinates may be copied from a given template structure. This approach results in the generation of improved 3D models of membrane proteins.

Chapter 4

MEDELLER: Co-ordinate Generation for Membrane Proteins

4.1 Context

In the previous chapters we introduced iMembrane, a method to annotate membrane proteins in terms of their membrane insertion. We defined structural environments unique to membrane proteins based on each residue's contact with the membrane head or tail groups and its position along the membrane normal. Environment-specific substitution tables (ESSTs) were created, encoding the amino acid substitution patterns observed within each environment. We showed that ESSTs of membrane environments differ significantly from those of soluble proteins. Finally, we demonstrated that the use of such membrane tables improves sequence-to-structure alignment of membrane proteins. In this chapter, we use the knowledge from the previous chapters and address the next step in our template-based modelling pipeline for membrane proteins. We present MEDELLER, a co-ordinate generation method specifically designed for mem-

brane proteins. This work has been published in the journal *Bioinformatics* (Kelm et al., 2010).

4.2 Motivation

Physically, membrane proteins differ significantly from water-soluble proteins (Schulz, 2002; Stevens and Arkin, 1999; Ulmschneider and Sansom, 2001). Soluble proteins often adopt a globular conformation, with their hydrophobic residues mainly in the protein core and their polar and charged residues predominantly on the water-exposed surface. Membrane proteins, on the other hand, sit in a lipid bilayer and thus contain stretches of residues that are exposed to the hydrophobic environment of the membrane (Eyre et al., 2004). Such transmembrane (TM) segments usually have one of two structure types: α-helices or β-strands. These structural differences between membrane and soluble proteins have been used in various computational methods (Punta et al., 2007), for example to identify membrane proteins from sequence alone (Gromiha and Suwa, 2005; Wallin and von Heijne, 1998).

A typical (template-based) homology modelling pipeline was described in Section 1.4. The final step in such a pipeline is co-ordinate generation based on the alignment between template protein structures and the target sequence. The three main approaches to this problem are (a) assembly of rigid bodies (Bates et al., 2001; Deane and Blundell, 2001; Koehl and Delarue, 1995; Petrey et al., 2003; Schwede et al., 2003), (b) segment matching (Levitt, 1992) and (c) satisfaction of spatial restraints (Sali and Blundell, 1993).

The current computational structure prediction methods may not be ideal for transmembrane proteins, designed as they are for water-soluble proteins (Elofsson and von Heijne, 2007). The physical differences between water-soluble and membrane proteins may mean that many of the steps in structure prediction should be approached differ-

ently.

There is currently no co-ordinate generation method specifically developed for membrane proteins. However, the successful ab initio fragment-assembly method ROSETTA has been adapted to predict membrane protein structure (ROSETTA Membrane (Yarov-Yarovoy et al., 2006)). The adapted method included an energy function that modelled a multi-layer artificial membrane environment. The performance of ROSETTA Membrane was tested on 12 membrane proteins of known structure, producing complete models with backbone RMSDs between 6 and 10Å. Sub-sets of each model (between 51 and 145 residues long) achieved RMSDs of 3 to 4Å. More recently, the method was modified to include experimentally derived constraints, such as known helix-helix contacts (Barth et al., 2009). The best models selected by the method had RMSDs of around 4Å, with all test proteins being single chains of at most 230 amino acids. The authors ventured that for larger proteins, multiple constraints were likely to be required in order to obtain accurate results. It should also be noted that, being a combinatorial ab initio method, ROSETTA requires large amounts of computing time and is typically run across large (possibly distributed) computing clusters. Homology modelling methods on the other hand, while relying on the availability of a template, can be run on a single desktop computer.

Wallner and Elofsson demonstrated (Wallner and Elofsson, 2005) that existing co-ordinate generators, on soluble proteins, showed little difference in terms of overall accuracy and none of the methods consistently produced models that were much closer to the native target structure than the template. In general, the models were also worse than simply copying the backbone co-ordinates of the template.

They identified Modeller (Sali and Blundell, 1993) as one of the best methods, due to its reliability and consistent model quality. Modeller attempts to satisfy spatial restraints in order to build a target protein structure. Its probability density function uses data obtained from the input target-template alignment, as well as prior knowledge

obtained from a database of structural alignments (Sali and Overington, 1994). This original database contained 105 families of soluble proteins and no membrane proteins. In this chapter, we use the most up-to-date version of Modeller (9v7, at the time of the experiment) as a representative of existing modelling methods and compare it to our own method.

Modeller's accuracy for modelling membrane proteins has been tested previously (Forrest et al., 2006). It was noted that it was possible to build models whose TM region had less than 2Å C-α RMSD to the native structure, given a template with >30% sequence identity. The accuracy for the whole protein was much lower than that of the TM region. This reflected important local differences in the regions connecting TM segments in membrane proteins with similar topology. This result indicated that template-based approaches can be successfully applied to membrane proteins.

However, this result also showed that Modeller (as a representative of the current standard in homology modelling software), in its current state, is not ideal for creating complete, accurate models of membrane proteins. A 2Å RMSD in the TM region is surprisingly high, given that the problem of TM protein prediction should theoretically be simplified by the additional physical constraints imposed by the presence of the lipid bilayer. Modelling errors in the TM region can then propagate to the loops that connect TM segments, thus resulting in even lower accuracy outside the TM region. Nevertheless, scientists commonly use Modeller, and comparable methods, to predict membrane protein structure (software reviews: (Reddy et al., 2006; Saxena et al., 2008); example case studies: (Fenosa et al., 2009; Yang et al., 2008)).

We present MEDELLER, a new method for co-ordinate generation specialised for membrane proteins. The input is a template protein structure and a sequence alignment between the target and template proteins. This alignment is not altered by MEDELLER. The most important part of the method is the identification of the reliable "core" structure shared by the template and target proteins. First, the template

protein's membrane insertion is calculated using iMembrane (Kelm et al., 2009). The "core" is initially restricted to the template residues buried in the middle layer of the membrane. It is then gradually extended using a specialised membrane-specific substitution score. The model is then completed, as far as possible, using the loop modelling protocols FREAD (Choi and Deane, 2010) and Modeller.

We test the modelling accuracy of our method on four large test sets, containing a total of 616 target-template pairs of transmembrane proteins.

Our method, MEDELLER, builds highly reliable core models (which usually correspond to a protein's TM region). Averaged over all test sets, MEDELLER produces more accurate core models and achieves a core model accuracy of 1.97Å RMSD versus 2.57Å for Modeller.

With added high-accuracy loops, MEDELLER remains the more accurate method in 65% of test cases and at least as good as Modeller in 77% of test cases, with an average accuracy of 2.62Å RMSD versus 3.16Å for Modeller.

4.3 Methods

4.3.1 Overview of the MEDELLER procedure

We have created a homology-based protocol for co-ordinate generation. The method is specific to membrane proteins. The algorithm is outlined below, with detailed explanations of the more complex steps following in separate sub-sections. Figure 4.1 gives a flowchart of the algorithm. All algorithm steps are identical, irrespective of the input protein's structure type (α-helical or β-barrel), although the substitution score itself is secondary structure-dependent and will thus differ for residues in helices or β sheets.

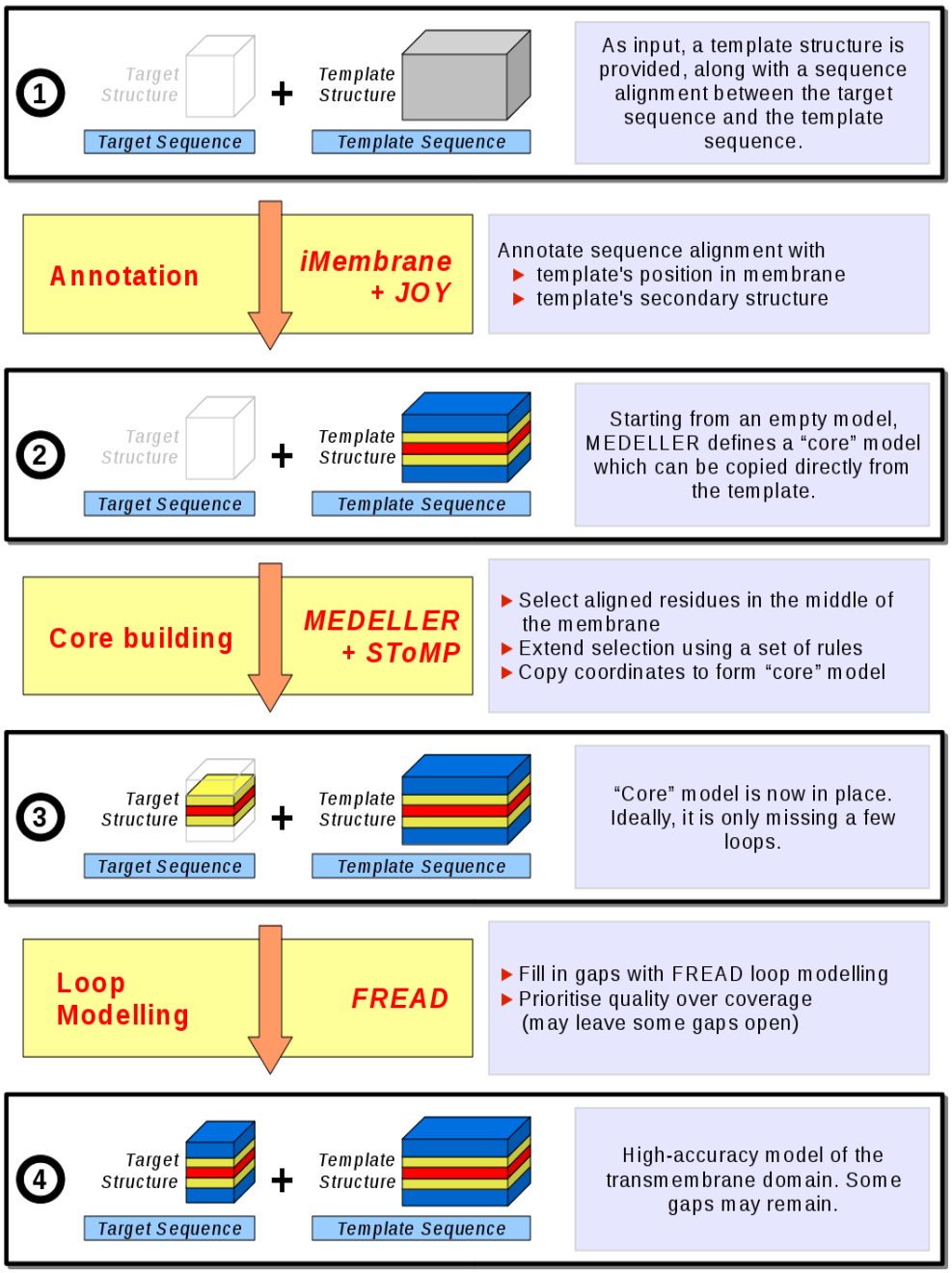


Figure 4.1: Algorithm for modelling from a single template - Progression from the initial user input to the high-accuracy model. When modelling from multiple templates, steps up to stage 3 are performed for each template separately. The data is then combined into a single core model (stage 3) and the algorithm proceeds as for a single template.

4.3.2 Algorithm Overview

1. **User Input.** The input to our co-ordinate generation method is the target protein's sequence, aligned to one or more homologous template protein sequences, as well as the 3D co-ordinates of those template structures.

2. **Annotation of the Sequence Alignment.** iMembrane (Section 4.3.3; (Kelm et al., 2009)) and JOY (Mizuguchi et al., 1998a) are run on the template structure in order to annotate its membrane insertion and secondary structure.

3. **Building the "core" model.** The core is built over four phases (Section 4.3.4). In each of these, a specific set of masking rules is in place (Section 4.3.5), alongside our smoothed fragment-based environment-specific substitution score (Section 4.3.6). Once the four core modelling stages are considered "complete" the remaining gaps, where the target and template are not aligned or where the substitution score is low, are modelled using FREAD (Choi and Deane, 2010). FREAD is a database search loop prediction method, which selects fragments based on an environment-specific substitution score and anchor RMSD. FREAD prioritises accuracy over coverage and thus may not make predictions for all the missing segments.

4. **Prioritising Accuracy or Coverage.** Our core building algorithm and our conservative use of FREAD are designed to give high accuracy co-ordinates. However, the model produced may still contain gaps. For the case where high coverage is required, MEDELLER also produces a "high-coverage" model, in addition to the default "high-accuracy" model. In this case, FREAD is used on a less conservative setting, resulting in a larger number of possibly less accurate loop predictions. Loops longer than 26 residues or terminal gaps cannot be modelled using FREAD. MEDELLER thus builds a set of backbone co-ordinates for the majority of the structure. For convenience, we include an option to fill any remaining gaps using

Modeller, allowing the user to always output a complete all-atom model, including side chains.

4.3.3 Insertion of Proteins into the Membrane

iMembrane (Chapter 2) is a method to ascertain a protein's position within the lipid bilayer. It relies on a database of known transmembrane protein structures, which have been simulated in an artificial lipid bilayer using molecular dynamics (Scott et al., 2008). iMembrane is used here to annotate the template protein structure with regards to its membrane insertion.

The version of iMembrane used in this chapter is the one described in Chapter 2 and not the one originally published (Kelm et al., 2009). The new version of iMembrane is included in the MEDELLER distribution and is available as a web server at http://imembrane.info.

4.3.4 Core Building Algorithm

A typical template-based co-ordinate generator starts from a given alignment between the target sequence and a single template structure. 3D co-ordinates are then copied from template residues to their corresponding, aligned, target residues. The underlying assumption is that aligned residues are evolutionarily related and share a similar structure. A model based solely on copying the backbone co-ordinates of all aligned residues is called a "naïve" model. Especially when the target and template proteins are distantly related, alignment quality will drop, introducing errors in regions that are hard to align (Rost, 1999). For membrane proteins, this will primarily be in the loop regions connecting transmembrane segments. These were demonstrated by Forrest et al. (2006) to be hard to model using existing modelling software. The centre of the transmembrane domains was, conversely, usually quite well conserved.

Our core modelling algorithm makes use of these observations, by first identifying

the central part of the transmembrane domain and then "growing" the model outwards, towards the loop regions, as long as our substitution score remains above a cut-off value. An example of this procedure is shown in Figure 4.2.

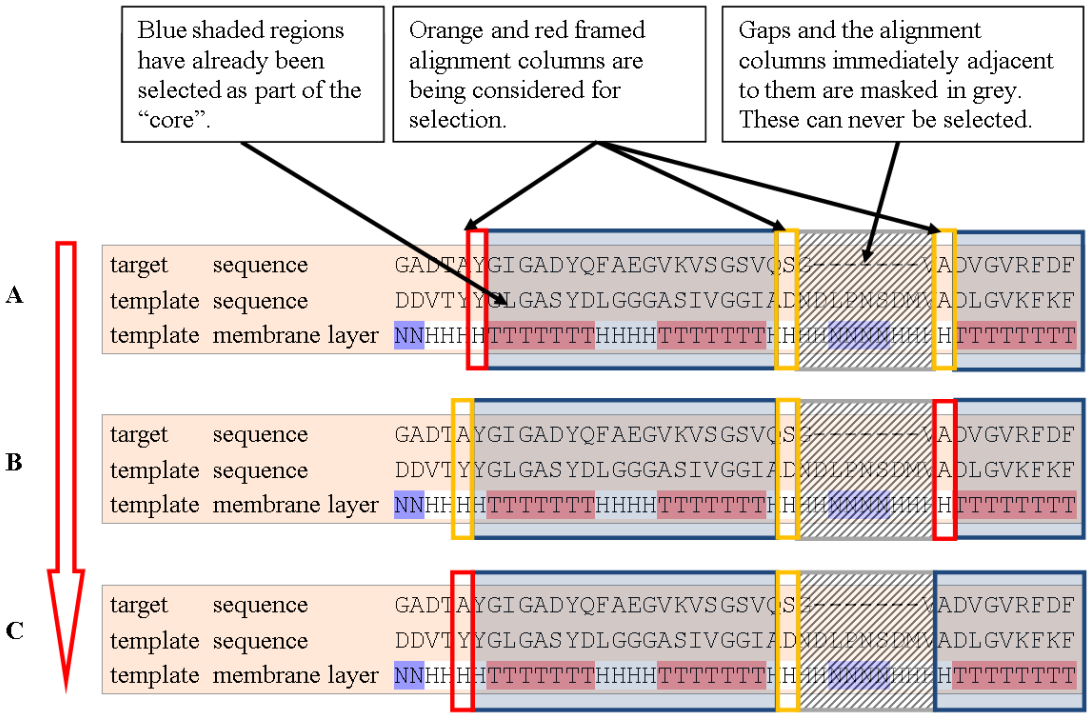


Figure 4.2: Core extension algorithm - In the "membrane layer" annotation, T, H and N respectively represent the middle hydrophobic (Tail group) membrane layer, the two peripheral polar (Head group) membrane layers and the two outer-most aqueous (Non-membrane) layers. **A:** Two "fragments" (stretches of alignment columns) have already been selected as part of the model core (blue boxes). All columns adjacent to an already selected (blue) column are now considered for selection (those highlighted in orange and red). The column marked in red has the highest substitution score and is thus selected. **B:** Again, the selection procedure is as in A. One new column is being considered for selection now: the one next to the column selected in A. **C:** The right-most fragment is now adjacent to a masked alignment region and can no longer be extended. Only columns next to the left-most fragment are considered, as the algorithm carries on. The subsequent steps are not shown.

1. **Selecting the "minimal core".** Using our membrane insertion and secondary structure annotation, we select the alignment columns containing those template protein residues, which are: (a) located in the middle layer of the membrane

(where the lipid tails reside) and (b) are not masked (Section 4.3.5).

2. **Core extension – phase 1: The "solid core".** Each selected fragment is extended, one column at a time, until all adjacent secondary structure elements or short, gap-less loops ($\leq$ 3 residues) have been selected. Extension stops when an alignment gap or a masked alignment column is reached (Section 4.3.5). If a short loop contains any alignment gaps, extension will stop before entering the loop. If a secondary structure element contains an alignment gap, it is selected right up to the gap.

3. **Core extension – phase 2: Adding long, gap-less loops.** Core extension continues in a similar way as described above. However, the alignment column masking rules have now changed to also allow longer ($>$ 3 residues) gap-less loops to be selected. At each step a single column is added. The column to be added is selected based on the smoothed fragment-based environment-specific substitution score (Section 4.3.6). If the value of the score falls below a predefined cut-off, the core extension procedure stops.

4. **Core extension – phase 3: Adding gappy and terminal loops.** Unless core extension was stopped during the previous phase, the procedure continues as above. The masking rules are changed (Section 4.3.5) to allow any remaining residues to be selected, that is gappy loops (loops that contain an alignment gap) and N- or C-terminal loops. As before, the core extension procedure will be stopped, if it reaches an alignment column whose addition would make the fragment score drop below a certain cut-off (Section 4.3.6).

5. **Filling any remaining gaps in the model.** All selected columns constitute the common core between template and target. We have now defined a core structure for our model of the target, which we refer to as the "core" model. This model usually still contains gaps, where the target and template are not aligned,

or where the substitution score is low. We use the loop modelling algorithm FREAD (Choi and Deane, 2009) to fill any remaining gaps with homologous fragments from a large fragment database, resulting in the "high accuracy" and "high coverage" models.

4.3.5 Alignment Column Masking

During each phase of the core building procedure, masks are used to prevent certain alignment columns from being selected. Masked columns may be those containing a gap in target or template, those annotated as "loop" (where masking may be dependent on loop length) or those outside the middle layer of the membrane (where the lipid tails reside). All rules are active at the beginning of the core building procedure and are then consecutively deactivated during the following phases of the algorithm. An overview of when each rule is active is given in Figure 4.3.

Rule	Minimal core (0)	Solid core (1)	Long loops (2)	Gappy loops (3)
Alignment gaps	X	X	X	X
Gappy & terminal loops	X	X	X	
Long gap-less loops	X	X		
Outside membrane "tail" layer	X			

Figure 4.3: Masking rules active during each core building phase - Each row corresponds to a single masking rule. Each column corresponds to a single phase during the core building procedure. X denotes that a rule is in effect. Note that the sets of active masking rules go from the most stringent combination (in phase 0, "minimal core") to the least stringent one (in phase 3, "adding gappy & terminal loops").

1. **Alignment gaps.** Alignment columns containing a gap in either the target or the template sequence are always masked, preventing any selected fragment from ever extending past an alignment gap. Residues adjacent to an alignment gap are also masked.

2. **Gappy & terminal loops.** Windows of columns, annotated as "loop" by JOY,

which contain a gap are masked. This rule represents the assumption that loops are the least structurally conserved regions of the protein. A gap in a loop region would indicate that the structures of the template and target loops do indeed differ. In addition, loops at the N- or C-terminal ends of the sequence alignment are also masked, as these are usually less structurally conserved. This rule is deactivated during the final core building phase.

3. **Long gap-less loops.** Loops longer than 3 residues, which do not contain an alignment gap, are masked. Long loops tend to be more structurally variable than shorter ones. We use this masking rule to treat long loops in a later stage of the core building procedure than their short ($\leq$3 residues) counterparts. This rule is deactivated during the "long loops" core building phase.

4. **Outside membrane "tail" region.** Any residue outside the middle layer of the membrane (where the lipid tails reside), is masked. This masking rule is applied only during the initial core building phase, in order to select the minimal core of the template structure, which is thought to be the structurally most conserved part of a transmembrane protein. The rule is deactivated during the "solid core" phase, where the extension of the "minimal core" begins.

4.3.6 Substitution Score

During core building, a smoothed fragment-based environment specific substitution score S_{cand} is used to determine the order, by which alignment columns are added to the model's core. During the later phases of core extension a score cut-off is used at each alignment column selection step, in order to decide whether the core building process should be halted.

4.3.6.1 Environment-specific substitution tables

In this chapter we use a set of 12 ESSTs dependent on *membrane layer* and *secondary structure* annotation (e.g. "helix residues in the membrane tail region"). The environment factors used are as follows:

- Membrane layer
  - membrane tail layer (T)
  - membrane head-group layers (H)
  - non-membrane layers (N)
- Secondary structure and phi angle
  - α-helix (X)
  - extended β-sheet (E)
  - coil/loop (C)
  - positive phi angle (P)

For each of the 12 possible combinations of the above environment classes, MEDELLER uses one substitution table: TX, TE, TC, TP, HX, HE, HC, HP, NX, NE, NC, NP.

The procedure of creating these ESSTs was described in Chapter 3.

Here, we use a far smaller number of ESSTs than considered in Chapter 3. This is due to MEDELLER being built before the examination described there.

4.3.6.2 Calculating the smoothed fragment-based environment-specific substitution score S_{cand}

Our ESSTs are used to assign a raw score S_{raw} (see Section 3.5.2) to every column in the sequence alignment. S_{raw} is smoothed over a window (of 3 residues, by default) to form $S_{smoothed}$ (Equation 4.1). S_{cand} is the score given to a candidate alignment column. It

is the sum of the smoothed scores $S_{smoothed}$ of all alignment columns already in the selected fragment, plus the candidate's own smoothed score.

$$S_{smoothed,i} = \frac{\sum_{j=-w}^{w} S_{raw,i+j}}{L} \tag{4.1}$$

$$S_{cand,i} = S_{smoothed,i} + \sum_{j=F_{first}}^{F_{last}} S_{smoothed,j} \tag{4.2}$$

where i, j, F_{first} and F_{last} are alignment column indices; i is the index of the alignment column, whose score is to be determined; w is a constant dependant on the window size (V), which is 3, by default; $w = (V-1)/2$; L is the actual number of scores inside the window ($L = w$ in the normal case, but can be smaller if the window contains alignment gaps or extends past an end of the sequence); F_{first} and F_{last} are the indices of the first and last columns of the fragment to be extended, respectively.

4.3.6.3 Calculating the substitution score cut-off

The substitution score cut-off S_{cutoff} for each core extension phase is dependent on the previous phase and is calculated as follows:

$$S_{cutoff} = \frac{2}{3} S_{mean,prev} + \frac{1}{3} S_{last,prev} \tag{4.3}$$

where $S_{mean,prev}$ is the mean score of the previous phase and $S_{last,prev}$ is the score of the last-added residue in the previous phase.

4.3.6.4 Using S_{cand} to determine selection order

Assume an alignment with several fragments (consecutive stretches of alignment columns) already selected. During a core extension iteration, one of the many fragments is extended by a single residue. In order to decide which fragment is extended by which

residue, the following steps are taken: (1) Identify all possible candidates (unselected alignment columns adjacent to an already selected column); (2) Discard any masked candidates; (3) Calculate the substitution score of each remaining candidate (equation 2); (4) select the best-scoring candidate. In the later phases of the core extension procedure, if the candidate score S_{cand} is below a defined score cut-off (equation 3), the candidate is rejected and core extension halted (Section 4.3.6.5).

4.3.6.5 Halting core extension

When core extension is halted, it regresses (i.e. columns are removed in the reverse order they were added) until a local S_{cand} score maximum is reached.

4.3.7 Multiple Templates

MEDELLER allows the user to provide more than one template protein. In this case, the same algorithm (Section 4.3.2, step 3) is run for each template separately, in order to identify the fragments that constitute the common core between the target protein and each template. Then, the fragments from all templates are pooled and the top-scoring subset of fragments is chosen to build the core model. Two fragments are allowed to overlap only if (a) the overlapping region is shorter than half the length of the smaller fragment, (b) there are no alignment gaps in the overlapping region, and (c) the backbone root mean square deviation (RMSD) of the overlap is lower than 1Å. Overlapping co-ordinates are melded (Choi and Deane, 2010). After assembling the single best-scoring core model, the algorithm proceeds as for a single template (Section 4.3.2, step 4).

4.3.8 Testing the Modelling Accuracy

Modelling accuracy was tested using the all-backbone-atom (C-α, N, C, O) RMSD between a model and the "native" target x-ray structure, as found in the PDB. In

addition, we report GDT_TS (Zemla et al., 1999) scores. We also analysed the models' transmembrane region in terms of tilt angle and rotation angle and shift (as number of residues) relative to the native structure. The accuracy of the models generated by our method was compared to the "top" model out of 10 equivalent models generated using Modeller with default settings. In every case, both methods started from an identical "ideal" sequence alignment, generated from a structure alignment between the template and the native target structure. The "top" Modeller model was selected using Modeller's own DOPE energy score (Eswar et al., 2007). To make the comparison fair, we calculated Modeller's RMSD using only that sub-set of residues present in the MEDELLER model. Two models with RMSDs to the native structure that differ by no more than 0.05Å were deemed to be equally accurate. We also conducted identical tests using the "best" Modeller model, which was selected from the set of 10 as the one with the lowest RMSD to the native target x-ray structure.

4.3.9 Sequence Identity and Coverage Measures

Sequence identity (ID) is calculated as the number of identical residues divided by the total number of alignment columns. On this scale, a value of approximately 20% identity corresponds to the "twilight zone" of sequence identity (Rost, 1999). Target coverage (Cov) is the number of residues, for which the model provides 3d co-ordinates, divided by the total length of the target sequence. Target "core coverage" (CoreCov) is calculated as above, except that the sequence is shortened to exclude any N- or C-terminal stretches of unmodelled residues.

4.3.10 Creation of the Test Sets

A list of membrane protein structures was compiled by combining data from several publicly available databases: all PDB entries annotated with the SCOP class "membrane proteins", the OPM database (Lomize et al., 2006), the PDB_TM database

(Tusnády et al., 2005) and the CGDB data-base (Scott et al., 2008). A list of unique PDB entries was compiled and filtered to include only x-ray structures with a resolution ≤ 3Å. These structures were split into single protein chains. The protein sequences were extracted and made non-redundant at a level of 80% sequence identity using CD-HIT (Li and Godzik, 2006). Thus, none of the target proteins in any of our test sets share more than 80% sequence identity. The remaining protein chains were run through iMembrane, in order to identify possible template structures in the CGDB database. The "structure search" option was used; this method searches iMembrane's database for homologous structures using pairwise structure alignment. All iMembrane search hits with a TM-score (TM-align (Zhang and Skolnick, 2005a)) >0.50 were kept. Some target proteins did not receive any iMembrane hits and were thus removed from the dataset. Pairs where target and template were the same protein chain were also removed. The remaining protein pairs were then classified into four test sets of varying sequence identity ranges. The test set for modelling multiple templates is a subset of the above set and contains targets associated with at least two templates.

4.4 Results

4.4.1 Test Sets

Our complete test set contains 616 pairs of protein chains (target-template pairs). Proteins of both the α-helical (413) and the β-barrel (203) type are included. The target-template pairs were classified, by their sequence identity, into 4 test sets representing 4 different levels of modelling difficulty:

- "easy" set: 128 protein pairs, 40-100% sequence identity
- "medium" set: 115 protein pairs, 20-40% sequence identity
- "hard" set: 102 protein pairs, 10-20% sequence identity
- "hardest" set: 271 protein pairs, 0-10% sequence identity

A target may be paired with more than one template. Conversely, a template may also be paired with more than one target. The full list of target-template pairs is given in Appendix 7.4.1. Our test set is approximately 17 times larger than the HOMEP set by Forrest et al. (2006) (36 pairs of membrane proteins) and roughly two thirds of the size of that used by Wallner and Elofsson (Wallner and Elofsson, 2005) (1036 representative protein pairs from different protein families) for testing co-ordinate generation in soluble proteins.

4.4.2 Modelling from a single template

Test Set	More acc	Less acc	MED rmsd	MOD rmsd	Diff rmsd	MED gdt_ts	MOD gdt_ts	Diff gdt_ts	CoreCov	Cov
all	65%	23%	2.62Å	3.16Å	-0.54Å	0.71	0.64	0.07	92%	71%
easy	68%	13%	0.93Å	1.56Å	-0.63Å	0.94	0.85	0.09	99%	88%
medium	59%	23%	1.92Å	2.40Å	-0.48Å	0.78	0.73	0.05	92%	80%
hard	49%	39%	2.82Å	2.97Å	0.15Å	0.65	0.62	0.03	90%	70%
hardest	72%	21%	3.64Å	4.32Å	-0.68Å	0.59	0.51	0.08	88%	58%

Figure 4.4: Accuracy of MEDELLER's high-accuracy model vs Modeller - Rows correspond to test sets. The "all" test set is the union of all the test sets. "More acc" and "Less acc" are the percentages of test cases where MEDELLER (MED) is more or less accurate, respectively, than Modeller (MOD). If the two methods' accuracies are within 0.05Å RMSD of each other, they are deemed equal. "MED rmsd" and "MOD rmsd" are accuracy values measured using the RMSD between the native structure and the MEDELLER or Modeller model, respectively. "Diff rmsd" is the difference in accuracy between the two methods. Analogously, "MED gdt_ts" and "MOD gdt_ts" are accuracy values measured using the GDT_TS score. GDT_TS scores are normalised by the number of target residues not in a terminal gap. "Diff gdt_ts" is the difference in GDT_TS between the two methods. Cov is the percentage coverage of the target sequence by the MEDELLER model. CoreCov is equivalent to Cov, but disregards uncovered terminal regions. All values are averaged over an entire test set.

We ran our new co-ordinate generator, MEDELLER, as well as Modeller on all target-template pairs in our test set. Modelling accuracy and target coverage was compared. The average accuracy of both methods as well as MEDELLER's coverage are summarised in Figure 4.4. The distribution of both methods' accuracy over all target-template pairs in the "easy" test set is shown in Figure 4.5. On average, our

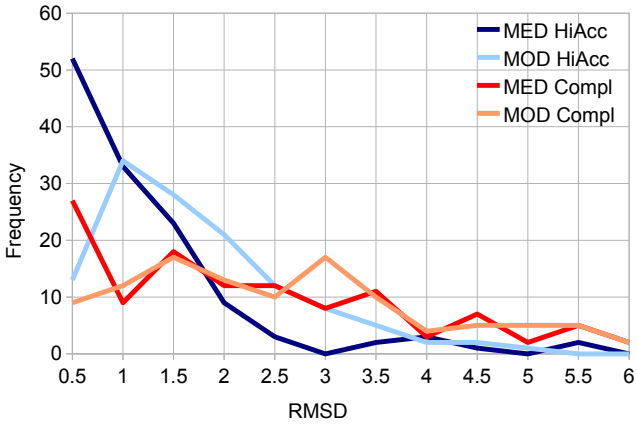


Figure 4.5: Distribution of modelling accuracy for the "easy" test set, when modelling from structure-based sequence alignments - Backbone RMSD (in bins of 0.5Å) to the native structure achieved by MEDELLER's high-accuracy (MED HiAcc) and complete (MED Compl) models and the corresponding co-ordinates in the Modeller model (MOD HiAcc, MOD Compl). MEDELLER's accuracy distribution peaks at the 0-0.5Å bin for both high-accuracy and complete models. Modeller peaks at 0.5-1Å and the 1-1.5Å bin, with its equivalent co-ordinates to the two model types.

method outperforms Modeller on the entire test set, in terms of accuracy, by 0.60Å RMSD for the core model and by 0.54Å RMSD for the high-accuracy model. On our "easy" data-set, the difference in average modelling accuracy between the methods is the most visible, with 0.71Å versus 1.46Å RMSD for the core model and 0.93Å versus 1.56Å RMSD for the high-accuracy model. This can also be seen in terms of GDT_TS with MEDELLER achieving a GDT_TS of 0.94 for high-accuracy models in the easy test set, whereas Modeller achieves 0.85.

The major advantage of MEDELLER lies in its highly reliable core models. Compared to Modeller, our core models are at least 0.05Å more accurate in 66% of test cases and at least as good (0.05Å) in 88% of test cases. A scatter plot of "core" model accuracies is shown in Figure 4.6, for both MEDELLER (A) and Modeller (B). With added high-accuracy loops, MEDELLER remains the more accurate method in 65% of test cases and at least as good as Modeller in 77% of test cases.

The worst core models made by the two methods have respective RMSDs of 5.40Å

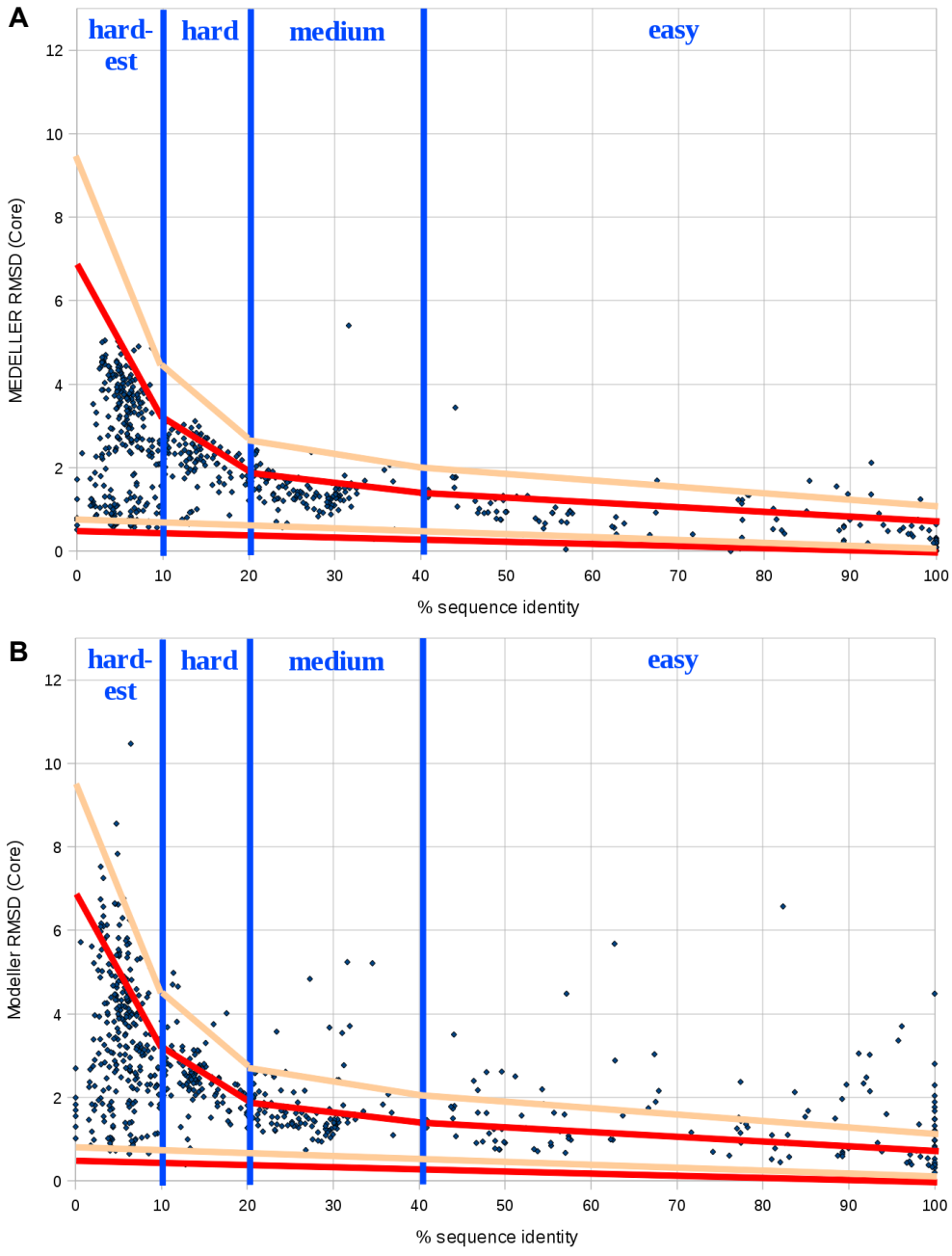


Figure 4.6: Modelling accuracy of the "core" model, over the entire test set - Results shown here are when modelling from structure-based sequence alignments made using TM-align. Accuracy is measured using backbone RMSD of the model to the native structure. **A:** Accuracy of MEDELLER's "core" model. **B:** Accuracy of Modeller's corresponding co-ordinates. Blue vertical lines show the division of the dataset into the "easy", "medium", "hard" and "hardest" subsets. Red and orange lines serve purely as visual aids for comparison between A and B and delimit the area containing the mass of data points for the MEDELLER (red) and Modeller (orange) models.

(MEDELLER) and 10.47Å (Modeller). Mostly, these inaccurate models are where target and template have different β-barrel diameters, or considerable local changes in helix bundle geometry. With added "high accuracy" loops, the worst accuracy values are 13.72Å and 13.33Å, respectively. Any errors in the core models are propagated and result in very poor loop prediction. In terms of loop accuracy, both globally and locally, MEDELLER's high-accuracy models outperform Modeller, with average RMSDs of 5.9Å (global) and 1.32Å (local). This relatively high global RMSD is mainly due to a number of loops whose general shape is correct, but which are at a wrong angle to the rest of the model, resulting in a very high RMSD. Both MEDELLER and Modeller show relatively poor performance if all loops are built (complete model) with global RMSDs of over 11Å. An overview of loop modelling accuracies for both methods is given in Figure 4.7.

In terms of coverage Modeller always provides a complete model. However, its loop regions tend to be unreliable. MEDELLER, on the other hand, provides four models: a core model with only highly conserved regions, a high-accuracy model containing high-confidence loops, a high-coverage model that includes low-confidence regions and a complete model. MEDELLER's modelling confidence is shown in the output using an atom's B-factor (Figure 4.10B). All our tests were repeated when selecting the "best" out of the 10 Modeller decoys using the RMSD to the native structure, instead of Modeller's DOPE energy score. This did not change the overall trend of the results.

4.4.3 Modelling from pure sequence alignments

All accuracy values reported in Section 3.2 are achieved when modelling from an "ideal" alignment, based on a structure superposition between target and template. Whether such values can be achieved in a real modelling case always depends on the quality of the input alignment. As a practical "worst case" scenario, we report accuracy values achieved when modelling from a simple pairwise sequence alignment, generated using

A: ALL

Model type	Loop count	Loop length	MED global	MED local	Mod global	Mod local
all loops	4145	7.42	8.34	2.2	**7.96**	2.2
hiacc	1682	5.76	**5.9**	**1.32**	6.36	1.73
hicov	1378	8.23	9.12	2.65	**7.24**	**2.35**
complete	1085	8.98	**11.12**	2.98	11.34	**2.72**

B: EASY

Model type	Loop count	Loop length	MED global	MED local	Mod global	Mod local
all loops	276	6.73	6.64	1.98	**6.53**	**1.89**
hiacc	88	5.5	4.51	**1.13**	**4.12**	1.51
hicov	72	7.08	7.91	2.32	**5.73**	**1.92**
complete	116	7.44	**7.46**	2.41	8.85	**2.17**

C: MEDIUM

Model type	Loop count	Loop length	MED global	MED local	Mod global	Mod local
all loops	648	6.99	7.45	2.1	**6.5**	2.08
hiacc	180	5.89	**4.61**	**1.23**	5.36	1.77
hicov	250	7.46	7.56	2.38	**5.21**	**2.2**
complete	218	7.37	9.66	2.48	**8.92**	**2.21**

D: HARD

Model type	Loop count	Loop length	MED global	MED local	Mod global	Mod local
all loops	1000	7.2	7.96	2.14	**6.81**	**2.07**
hiacc	420	5	5.58	**1.33**	**5.25**	1.57
hicov	391	7.81	8.78	2.56	**6.33**	**2.16**
complete	189	10.84	11.51	3.09	**11.24**	**2.97**

E: HARDEST

Model type	Loop count	Loop length	MED global	MED local	Mod global	Mod local
all loops	2221	7.73	**8.98**	**2.28**	9.08	2.33
hiacc	994	6.09	**6.38**	**1.35**	7.21	1.82
hicov	665	8.88	10.04	2.85	**8.71**	**2.56**
complete	562	9.29	**12.32**	3.26	12.82	**2.95**

Figure 4.7: Global and local loop accuracy in MEDELLER and Modeller models, across the test sets. - "Loop count" is the number of example loops in a particular dataset and model type. "Loop length" is the average length of these same loops. All other numeric values are average loop RMSDs. The global loop RMSD was calculated, for each loop separately, by first aligning the model to the native structure using only the model core and then calculating the RMSD between the loop coordinates. The local loop RMSD was calculated by directly aligning the loop coordinates of the model and the native structure and then calculating the RMSD between the loop coordinates. med = MEDELLER; mod = Modeller.

Test Set	MED rmsd	MOD rmsd	Diff rmsd	MED gdt_ts	MOD gdt_ts	Diff gdt_ts	CoreCov	Cov
all	8.08Å	8.23Å	-0.16Å	0.48	0.44	0.04	89%	71%
easy	0.96Å	1.78Å	-0.82Å	0.94	0.84	0.10	99%	95%
medium	3.49Å	3.76Å	-0.27Å	0.67	0.62	0.04	92%	80%
hard	8.87Å	8.87Å	0.00Å	0.37	0.34	0.03	87%	66%
hardest	13.08Å	12.94Å	0.15Å	0.23	0.21	0.02	83%	57%

Figure 4.8: Accuracy of MEDELLER's high-accuracy model vs Modeller when modelling from MUSCLE sequence alignments - Rows correspond to test sets. The "all" test set is the union of all the test sets. "More acc" and "Less acc" are the percentages of test cases where MEDELLER (MED) is more or less accurate, respectively, than Modeller (MOD). If the two methods' accuracies are within 0.05Å RMSD of each other, they are deemed equal. "MED rmsd" and "MOD rmsd" are accuracy values measured using the RMSD between the native structure and the MEDELLER or Modeller model, respectively. "Diff rmsd" is the difference in accuracy between the two methods. Analogously, "MED gdt_ts" and "MOD gdt_ts" are accuracy values measured using the GDT_TS score. GDT_TS scores are normalised by the number of target residues not in a terminal gap. "Diff gdt_ts" is the difference in GDT_TS between the two methods. Cov is the percentage coverage of the target sequence by the MEDELLER model. CoreCov is equivalent to Cov, but disregards uncovered terminal regions. All values are averaged over an entire test set.

MUSCLE (Edgar, 2004a). As expected, the resulting model accuracy was reduced for both MEDELLER and Modeller. On average MEDELLER's high-accuracy model achieved a backbone accuracy of 0.96Å compared to 1.78Å for the equivalent Modeller co-ordinates, on the "easy" test set. The average accuracy of both methods as well as MEDELLER's coverage are summarised in Figure 4.8. In practice, users will most likely use a more sophisticated alignment method, such as our membrane-specific version of FUGUE (Chapter 3), yielding somewhat better accuracies.

It should, of course, be understood that, even when using MUSCLE alignments as input, we still base our results on the prior knowledge that the chosen template is adequate for homology modelling (target-template TM-score > 0.50; see Section 4.3.10). In a standard modelling case, this knowledge is not available and one has to rely on the accuracy of homology detection methods.

A

	NAIVE				MEDELLER Core				MEDELLER HiAcc			
	Cov	CC	bckb	TSc	Cov	CC	bckb	TSc	Cov	CC	bckb	TSc
	%	%	rmsd	gdt	%	%	rmsd	gdt	%	%	rmsd	gdt
all	**82.04**	**90.26**	**2.77**	**0.65**	**65.92**	**85.84**	**1.97**	**0.69**	**70.56**	**91.81**	**2.62**	**0.71**
easy	96.72	98.03	1.06	0.92	86.37	97.15	0.71	0.94	87.69	98.64	0.93	0.94
medium	91.00	93.59	2.13	0.75	76.95	88.33	1.47	0.76	80.21	92	1.92	0.78
hard	83.74	91.19	3.15	0.60	65.79	85.2	2.2	0.63	70.44	90.61	2.82	0.65
hardest	70.66	84.83	3.71	0.50	51.63	79.67	2.68	0.56	58.42	88.96	3.64	0.59

B

	NAIVE				MEDELLER Core				MEDELLER HiAcc			
	Cov	CC	bckb	TSc	Cov	CC	bckb	TSc	Cov	CC	bckb	TSc
	%	%	rmsd	gdt	%	%	rmsd	gdt	%	%	rmsd	gdt
all	**87.11**	**90.97**	**9.58**	**0.43**	**68.81**	**86.21**	**7.96**	**0.48**	**70.85**	**88.79**	**8.08**	**0.48**
easy	97.81	99.07	1.15	0.93	94.3	98.52	0.93	0.94	94.83	99.05	0.96	0.94
medium	92.58	94.12	4.87	0.61	76.98	88.65	3.3	0.65	79.67	92.04	3.49	0.67
hard	85.97	89.85	11.55	0.29	63.23	82.75	8.74	0.36	66.33	86.54	8.87	0.37
hardest	80.16	86.22	14.83	0.16	55.41	80.67	12.97	0.23	57.49	83.41	13.08	0.23

Figure 4.9: Accuracy of MEDELLER's core and high-accuracy models compared to the naïve model - A: Modelling from TM-align structure-based alignments; **B:** Modelling from MUSCLE sequence alignments. Rows correspond to test sets. The "all" test set is the union of all the test sets. Abbreviations: Cov, fraction of the target covered by a particular model; CC, "core coverage", i.e. the fraction of the target covered by the model when ignoring terminal gaps and gaps longer than 26 residues (which cannot be closed with FREAD loop modelling); bckb, backbone RMSD; TSc, GDT_TS normalised by CC.

4.4.4 Comparison to the naïve model

When modelling from an "ideal" alignment, the core selected by MEDELLER is on average smaller than the naïve model but has a far lower RMSD (1.97Å for MEDELLER vs 2.77Å for the naïve model). In other words, MEDELLER's core model successfully excludes residues away from the conserved transmembrane portion of the protein, thus achieving a better accuracy. MEDELLER's high-accuracy model is in between the core and naïve models, in terms of coverage and RMSD (2.62Å), but has a very similar "core coverage" (CC) to the naïve model (see Figure 4.9A). In other words, the loop modelling step closes most internal gaps (resulting in a high CC), while residues away from the conserved regions remain excluded from the model (resulting in a lower overall coverage but also lower RMSD than the naïve model).

The same trend is observed when modelling from a pure sequence alignment (0.93Å and 0.96Å RMSD for MEDELLER's core and high-accuracy models vs 1.15Å for the naïve model, on the "easy" test set). For further details, see Figure 4.9B.

4.4.5 Main chain bumps and transmembrane geometry

The model quality assessment software WHATCHECK (Hooft et al., 1996) was run on the complete MEDELLER and Modeller models. On average, both methods had a significantly higher amount of bumps between main chain atoms than the template and target structures. Modeller produces slightly fewer bumps, thanks to its model refinement function. For details, see Appendix 7.4.2.

Using iMembrane, we also compared the accuracy of transmembrane helix/β-sheet geometry, relative to the native x-ray structure. While both methods behaved very similarly, on average MEDELLER produced slightly better geometries. MEDELLER and Modeller, respectively, had average TM shifts of 1.9 vs 2.3 residues, tilt angle deviations of 2.4° vs 3.4° and rotation angle deviations of 31.2° vs 36.0°. For further details, see Appendix 7.4.3.

4.4.6 Modelling from multiple templates

We tested both MEDELLER and Modeller on a set of 35 target proteins with at least 2 templates per target. On average, MEDELLER's core model achieves an accuracy of 3.24Å RMSD, compared to 3.32Å for the corresponding Modeller co-ordinates. This average is heavily biased by two outliers, where both methods' accuracy was worse than 9Å. In one case, Modeller is the least bad of the pair, in the other case MEDELLER is. After discarding these two test cases, the two methods' average accuracies are 2.20Å for MEDELLER and 2.44Å for Modeller. In almost every test case, both MEDELLER and Modeller could create a better core model when given only a single template structure (e.g. the one with the highest sequence identity to the target). For further details, see Appendix 7.4.4.

4.4.7 Example model: Human adenosine A2A receptor

The human adenosine A2A receptor (PDB 3EML, chain A) is a G-protein coupled receptor (GPCR). The structure was resolved at a resolution of 2.6Å and contains only a single gap in a loop connecting two TM helices. In 2008, this protein was the subject of a CASP-style blind prediction competition (Michino et al., 2009) in order to assess the current state in GPCR membrane protein structure prediction. The most direct comparison possible is between values for TM helix accuracy from Michino et al. and our MEDELLER "core" models, which are roughly equivalent.

Our test set contains three models of 3EML (all from the "hard" set), with core accuracies (Cα RMSD) of 2.1Å, 2.3Å and 2.5Å. The model in Michino et al. to best predict the TM helices had a Cα RMSD of 2.1Å (the model submitted by Davis, Barth and Baker). The average TM helix (Cα) RMSD for all submitted models was 2.8±0.5Å. This places MEDELLER's models at the top end of the scale. Of course this comparison is not entirely fair, as our models are based on a structure alignment to the native structure. However, Michino et al. reported that alignment did not seem

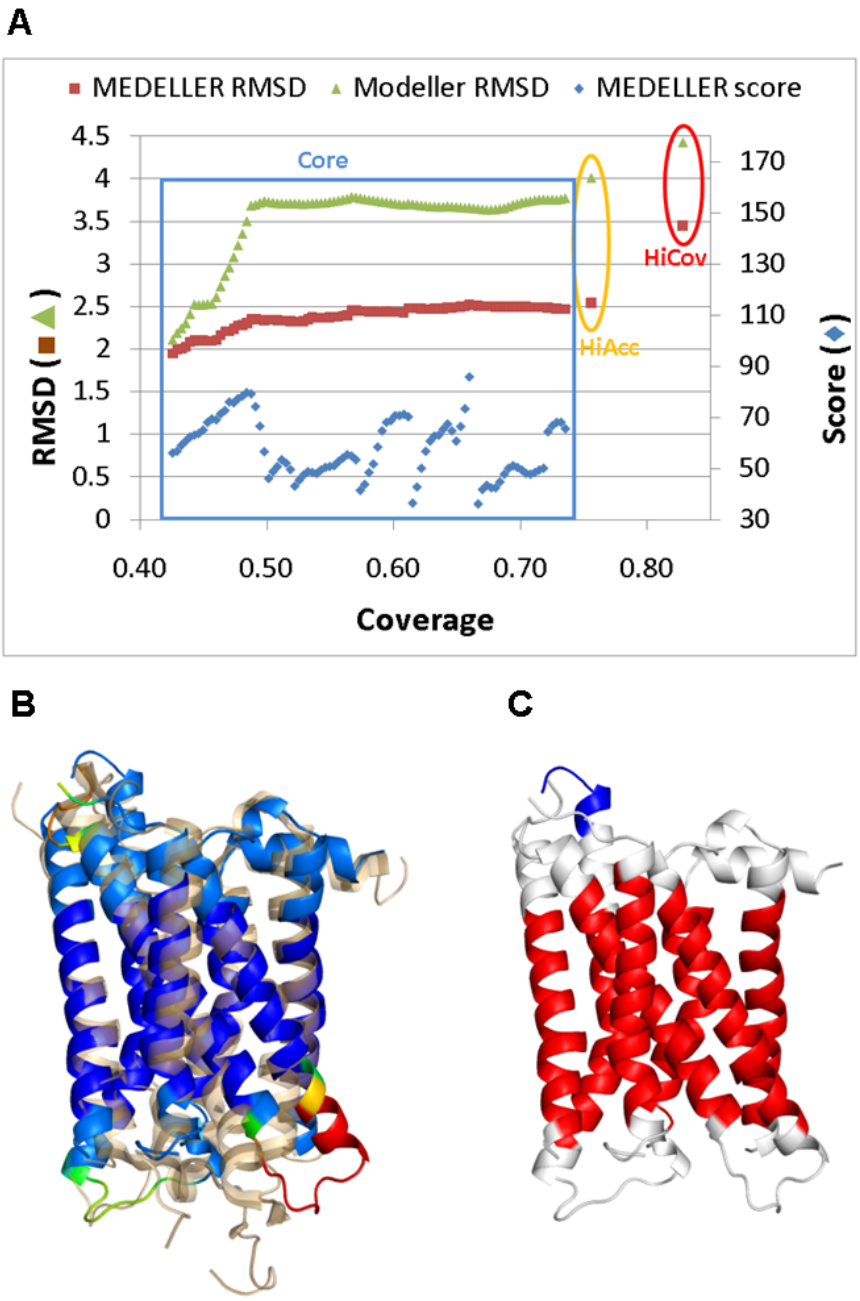


Figure 4.10: Example model of human adenosine A2A receptor (PDB 3EML, chain A) - (A) Progression of model accuracy through the modelling process. The modelling phases corresponding to the different MEDELLER models [Core, High Accuracy (HiAcc) and High Coverage (HiCov)] are labelled. (B) MEDELLER's high-coverage model, coloured by modelling confidence in a blue-to-red spectrum, aligned to the native x-ray structure (transparent orange). (C) MEDELLER's high-coverage model, coloured by membrane insertion using iMembrane [red, middle hydrophobic (tail group) layer; white, peripheral polar (head group) layers; blue, aqueous (non-membrane) layers].

to be a problem due to the conserved TM sequence patterns.

The high-coverage MEDELLER model with a core (Cα) RMSD of 2.5Å is shown in figure 4.10. The template structure is Opsin (PDB 3CAP, chain A), another GPCR. The MEDELLER model is more accurate than the corresponding co-ordinates of the top Modeller model. MEDELLER's backbone RMSD was 2.64Å, 2.65Å and 3.57Å for its core, high-accuracy and high-coverage models, respectively. Modeller achieved an RMSD of 4.70Å, 4.83Å and 5.11Å for the corresponding co-ordinates.

Loops modelled using FREAD had the right general shape, even in the high-coverage model, whereas Modeller's loops tended to be at the wrong angle relative to the core of the protein. Only one very long loop was not modelled by FREAD. Modeller produced co-ordinates for this loop but these were far from the loop's position in the x-ray structure.

One particular mistake was made by both methods: failure to predict a helix kink that was present in the native structure but not in the template. For further details, see Appendix 7.4.5.

4.5 Discussion and Conclusion

We present MEDELLER, a new template-based co-ordinate generation protocol for membrane proteins. First, a common core between target and template proteins is defined, using membrane-insertion and secondary structure information. Then, any gaps in this core model are completed, as far as possible, using FREAD, a database search loop modelling algorithm. Finally, any remaining gaps are filled using Modeller. This results in four models with increasing target coverage: the "core", "high-accuracy", "high-coverage" and "complete" models. In the output, co-ordinate reliability is indicated using B-factors.

MEDELLER's algorithm speed is comparable to that of Modeller. Modeller, with-

out molecular dynamics optimisation, takes between 5 and 30 seconds to generate one model on a single Intel Xeon 2.33GHz processor core. This is multiplied by the number of models the user chooses to generate (i.e. about 50-300 seconds for 10 models). In addition, the user may choose to run a separate model quality assessment software, to decide which of the generated models to keep. MEDELLER takes 1 to 4 seconds to generate the "core" model of a membrane protein, which is likely to be of better accuracy than the Modeller counterpart. For the FREAD loop modelling step, the additional runtime depends on the number of gaps in the model. Per gap (of any length from 1 to 26 residues), one should expect about 45-190 seconds of runtime (on the same processor as above). Completing the model with Modeller adds another 5-30 seconds to the runtime. Future versions of MEDELLER should include a re-implemented version of FREAD that will take less time to run (see Chapter 5).

MEDELLER's core and high-accuracy models are potentially incomplete. However, the level of certainty in these co-ordinates is high. We have shown that MEDELLER's core models (1.97Å average RMSD) are consistently more accurate than their corresponding co-ordinates in the Modeller models (2.57Å average RMSD). Even adding high-accuracy loops MEDELLER still outperforms Modeller in most cases. MEDELLER's high-coverage model represents a trade-off between accuracy and coverage. At this coverage level, the two methods are, on average, approximately equal, with MEDELLER better in the "easy" set and Modeller better in the "hard" set. Where complete coverage is required, we also produce a complete model by filling any remaining gaps using Modeller. However, the accuracy of such co-ordinates is unreliable. Here both methods have average RMSDs of over 10Å, mainly due to large regions of the targets not aligned to a template in the "hard" and "hardest" test sets. Only on the easy set, where unaligned regions are short, MEDELLER's complete models have a clear advantage with 2.80Å RMSD versus 3.39Å for Modeller. In the "medium" set, Modeller's complete models lead with 5.33Å versus 5.84Å for MEDELLER.

The fact that MEDELLER's core models are consistently better than Modeller's, even though MEDELLER employs no structure optimisation methods, suggests that Modeller's probability density function, which was created to model soluble proteins, may distort the template structure of a membrane protein.

The high quality of MEDELLER's core models is achieved by reliably selecting parts of the template structure that are similar to the correct target co-ordinates. This is made possible by using membrane insertion annotation and an environment-specific substitution score along with the rule-based MEDELLER algorithm (Sections 4.3.2 - 4.3.6).

In terms of loop modelling, MEDELLER is the first software to use the recently revised FREAD algorithm (Choi and Deane, 2010). For soluble proteins, FREAD guarantees consistent loop quality independent of loop length as long as the anchor structures are correctly modelled. The high accuracy of our core models allows FREAD, in many cases, to produce accurate loop structures ("high-accuracy" models). In our test set, FREAD sometimes does not correctly filter out loops which are similar in shape to the native structure but are attached at a wrong angle to the model core. We have not observed this phenomenon when modelling loops in soluble proteins. In future versions of MEDELLER, we hope to introduce membrane protein-specific loop selection procedures to deal with such errors (see Chapter 5).

MEDELLER's loss in accuracy in the higher coverage models, especially at low target-template sequence identity, is due to small local structure differences in the membrane protein "core" region. These errors are amplified when adding loops to the model, based on such erroneous anchor structures. Modeller lessens this dependency using its model refinement method. This has two opposing effects, however: smaller errors at low target-template identity (which is desirable) but also worse core models overall (which is not).

The obvious way to improve model accuracy would be to either greatly reduce the

size of the model core or to create a refinement method that corrects local structural differences such as helix kinks. Helix kink prediction will allow better core accuracy, even with templates that are locally different from the correct target structure. Future versions of MEDELLER should include such refinement methods, as well as a more accurate ab initio loop modelling protocol that should allow for better "high-coverage" and "complete" models. This is discussed further in Chapter 5.

4.6 What's next

MEDELLER is a working tool to model membrane proteins. However, membrane-specific information has only gone into the modelling of the core model. Loops are, so far, being modelled using FREAD, a method developed for soluble proteins. Its results already outperform Modeller when conservative substitution score cut-off values are used. Nevertheless, results from Chapter 3 have shown that membrane protein loops have significantly different substitution patterns from soluble protein loops, when considering residues in direct contact with the membrane. It should be possible to improve loop modelling results for membrane proteins, either by using a database containing only membrane proteins, or by replacing the FREAD substitution score with one specific to membrane protein loops. This will be explored in the next chapter, in order to further improve the overall quality of the MEDELLER models.

Another problem that has become apparent is the presence of helix kinks and twists within the "core" models. The more distant the target and template proteins, the higher the chance of finding such local structural variation, resulting in errors in the final model. Prediction of the location of such errors and model refinement to correct them would increase the model quality for all but the easiest modelling cases. This problem is not addressed in this thesis, but should be the subject of future work, discussed briefly in the final chapter.

Chapter 5

PyFREAD: High-Speed Loop Modelling for Membrane Proteins

5.1 Context

The previous chapters have described a homology-based modelling approach specifically developed for membrane proteins. The input provided by the user is the sequence of a target protein, which is to be modelled, and the structure of a suitable template. This template is annotated in terms of its membrane insertion using iMembrane (see Chapter 2) and, possibly, re-aligned to the target sequence. Chapter 3 has demonstrated that superior alignment quality can be achieved in target-template alignment by using environment-specific substitution tables of membrane proteins. From this alignment, a highly accurate core model of the target protein can be generated using the MEDELLER algorithm (Chapter 4). The loops in Chapter 4 were modelled using the FREAD database search algorithm. This final step is the only remaining one not using membrane protein-specific information.

5.2 Motivation

In membrane proteins, transmembrane (TM) segments are usually one of two structure types: alpha helices or beta strands. These TM segments are connected to each other by stretches of amino acids with irregular structure, termed loops. In helical TM proteins, TM helix geometry is often well-conserved, whereas the structure of loop regions can vary greatly between homologues (Forrest et al., 2006). Therefore, loops tend to be the parts of transmembrane proteins that are the hardest to model.

The loop modelling algorithm FREAD has been shown to produce consistently accurate loop models in soluble proteins (Choi and Deane, 2010). Using the same default parameters, FREAD also achieves a good average accuracy in membrane proteins, in cases where the core model is sufficiently accurate, and performs markedly better than the popular programme Modeller (see Chapter 4). However, compared to soluble proteins FREAD's accuracy is reduced and unexpected outliers appear where the local loop structure is often correct, but globally the loop is in the wrong orientation.

TM segments tend to be roughly parallel to each other and loops connecting them can thus be expected to have characteristic shapes specific to membrane proteins. Operating under the hypothesis that membrane protein loops are potentially shaped differently from soluble protein loops, we created a fragment database of only membrane proteins. In this chapter we show that by using such a specialised database it is possible to greatly improve modelling accuracy. In addition we explore the use of membrane protein-specific substitution tables.

We performed these tests using a re-written version of FREAD, PyFREAD. The new program runs an order of magnitude faster than the original implementation, due to improvements to its database architecture and search algorithm. In addition, a simplified parameter scheme results in slightly higher coverage.

PyFREAD is intended to replace FREAD as the final modelling step in the MEDELLER

suite, described in Chapter 4, resulting in higher accuracy models of membrane proteins and reduced runtimes.

5.3 Materials and Methods

5.3.1 Definition of a loop

In structural terms, a "loop" is typically a stretch of amino acids that has no obvious secondary structure (i.e. is neither helix nor strand). In a practical loop modelling situation, the definition is slightly different. Here, "loop modelling" typically fills the gaps in an existing template-based model. In other words, regions conserved between the target and template proteins have been modelled, but unconserved regions have not yet been assigned co-ordinates. Most often, these regions do correspond to a structural loop, but this need not necessarily be the case.

Thus, a "loop", for the purposes of this chapter, is simply a stretch of amino acids of unknown 3D structure, bounded by two "anchor" regions of known structure. Only the amino acid sequence of the "loop" is known. PyFREAD employs both the anchor structure and loop sequence to assign 3D co-ordinates to the loop.

5.3.2 PyFREAD

5.3.2.1 Overview of the FREAD algorithm

The FREAD algorithm, which is at the heart of the PyFREAD program, relies upon a database of known protein structures. The input to the algorithm is the incomplete model of a target protein, lacking the co-ordinates of a particular query loop. The loop's amino acid sequence is known and is provided as a second input. From the model, the main chain co-ordinates of the N-terminal and C-terminal loop "anchors" (two residues on each side of the loop) are extracted. The database of known protein structures is then searched for fragments of the required length with a similar sequence

and anchor geometry.

The anchor geometry match is performed by superimposing the query and database anchor co-ordinates and calculating the RMSD for all main chain atoms (N, Cα, C, O). Here, only loops with an anchor RMSD match below 1Å are considered (the least stringent cut-off employed in the original FREAD program).

The sequence match is made using environment-specific substitution tables. These tables are slightly different from those discussed in Chapter 3 in that they are based on different structural environments. The environments used here are six classes of dihedral (ϕ and ψ) angles, which are partitions in the Ramachandran plot (Figure 5.1). Typically, a score cut-off of 25 is used to identify "good" matches (Choi and Deane, 2010).

Once suitable database loops have been identified, they are inserted into the model and a clash check is performed. Loops clashing with the model framework are discarded. This step replaces the Samudrala-Moult pseudo-energy calculation used in the original FREAD algorithm (Choi and Deane, 2010). It has the advantage of being applicable to any type of protein, without needing to be trained on a set of known protein structures.

Finally, database loops matching the search criteria (also called "decoys") are ranked by their anchor RMSD and returned to the user, along with their 3d co-ordinates.

5.3.2.2 Why PyFREAD is fast: Database structure and search procedure

The original FREAD program used a database of Cα distances (between anchor residues) to preselect potentially interesting loops. These loops where then extracted from their PDB-style structure files and anchor RMSD and sequence score computed. The cut-off values for Cα distances were determined empirically, with the most recent version using a fixed cut-off of 0.7Å independent of loop length. In addition anchor RMSD cut-offs were scaled by loop length, with a maximum cut-off of 1Å for loops longer than 10

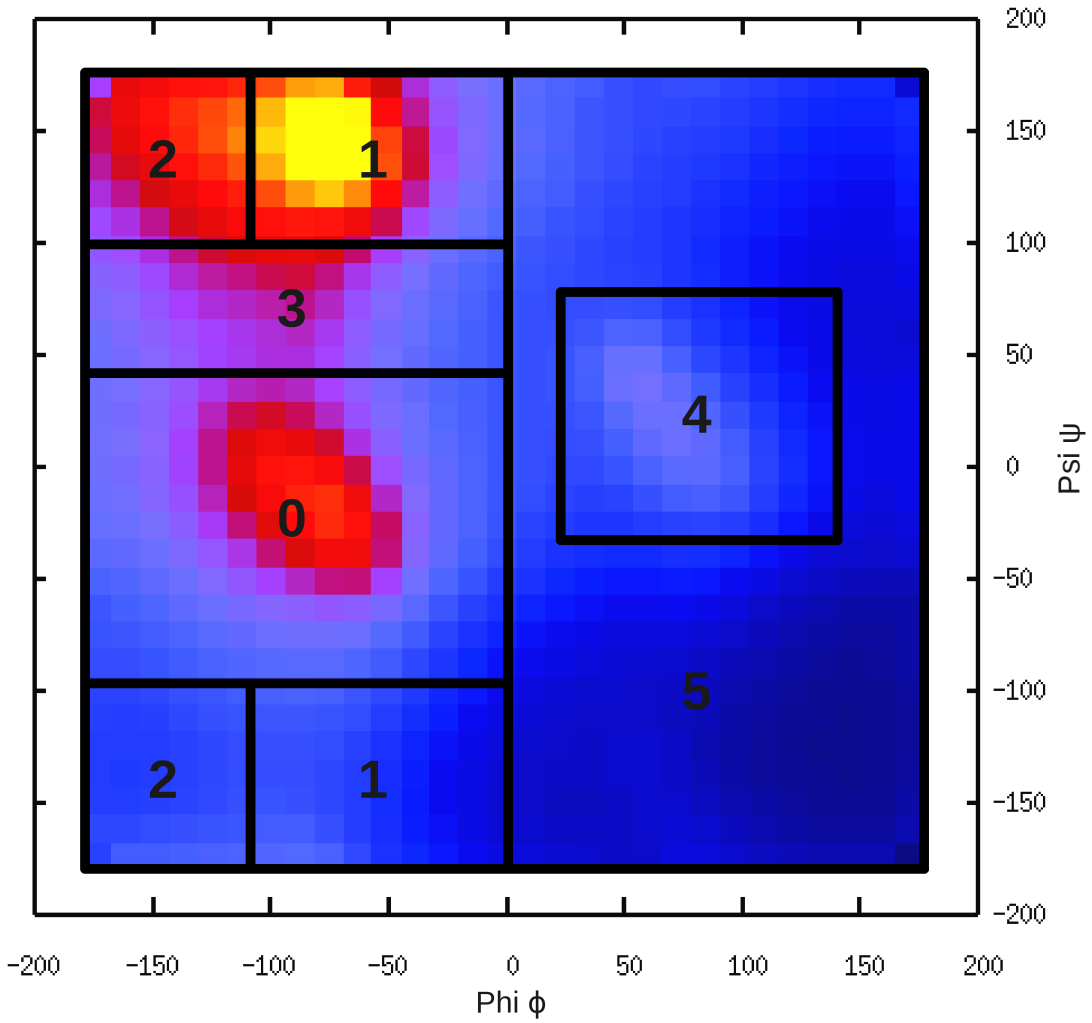


Figure 5.1: Ramachandran plot showing PyFREAD's dihedral classes - The X and Y axes show an amino acid's dihedral angles (ϕ and ψ). The plot is partitioned into six classes of dihedral angle combinations numbered from 0 to 5. These "dihedral classes" form the structural environments used in PyFREAD's environment-specific substitution score. The heat map in the background represents the observed frequencies of dihedral angles in membrane protein loops (as defined by JOY) with warmer colours indicating higher frequencies. The equivalent plot for soluble protein loops has a virtually identical shape.

residues.

The main bottleneck of this implementation was the need to parse a large number of structure files, in order to compute the RMSD and sequence score of each candidate loop. The purpose of the Cα distance cut-off was to reduce the number of loops for which this was necessary. However, a large number of loops (hundreds of thousands) did meet this distance cut-off criterium while failing to satisfy the RMSD and sequence score cut-offs.

The PyFREAD database is designed in such a way as to avoid parsing structure files for all but the most promising database matches, but without sacrificing coverage. This is achieved by extracting most of the required information from the structure files and archiving it in a SQL database. Using a single SQL query, database loops can now be effectively filtered down to that small subset of loops that potentially match the query criteria.

The information contained in the SQL database includes the loop sequence and dihedral angles, allowing PyFREAD to compute the substitution score for each database loop without touching a single structure file. In addition, the Cα distances between the anchor residues are extracted.

PyFREAD allows the user to set two cut-off values: the maximum RMSD R_{max} (1Å by default), and the minimum substitution score S_{min} (25 by default). Internally, additional cut-offs are imposed upon the Cα distances between anchor residues, but the values of these parameters are automatically determined from the anchor RMSD cut-off and are invisible to the user.

PyFREAD's search strategy to identify matching loops is as follows:

1. For each loop length a separate SQL database exists. Only the subset of loops with the required loop length is considered.

2. For any pair of anchor Cα atoms i and k, where i is part of the N-terminal anchor

and k is part of the C-terminal anchor, restrict the Euclidean distance d_{ik} to be within the margin Δd of the values in the query (Figure 5.2). The size of the margin is scaled as: $\Delta d = 3R_{max}$. This scaling was determined from randomly generated sets of 3d co-ordinates (Figure 5.3) and set to have no impact on the search results. All decoys rejected in this step would have failed to meet the RMSD cut-off R_{max} (in Cα space).

3. Calculate the "internal anchor RMSD", which is a combined measure calculated from all the Cα distances d_{ik} defined in the previous step. It is half the root mean square difference of the d_{ik} values within the query and decoy anchors (red and yellow dotted lines in Figure 5.2):

$$RMSD_{internal} = \frac{1}{2}\sqrt{\text{Average}_{i,k}\left(\left[d_{ik}(query) - d_{ik}(decoy)\right]^2\right)} \qquad (5.1)$$

where $\text{Average}_{i,k}$ is the average function over all values of i and k. This "internal" anchor RMSD has the desirable property of being independent of the orientation of the two shapes being compared. It thus requires no superposition and is fast to compute. In addition, it is always less or equal to the optimal anchor RMSD (of the Cα co-ordinates), see Figure 5.4. Therefore, applying the same RMSD cut-off R_{max} to this parameter results in virtually no loss of coverage, while eliminating a large number of decoys that would have failed to meet the R_{max} cut-off.

4. Calculate the substitution score and apply the cut-off S_{min}. Storing the loop sequence and dihedral angle classes within the SQL database avoids the time-consuming task of parsing a great number of structure files.

5. Load the 3D co-ordinates of each decoy loop from its corresponding PDB-style structure file.

6. Insert each loop decoy into the query model by optimally superimposing the

anchor backbone co-ordinates (N, Cα, C, O). Discard any loops not meeting the anchor RMSD cut-off R_{max}.

7. Optionally, close the loop to perfectly fit the query anchors, using the Cyclic Co-ordinate Descent algorithm (Boomsma and Hamelryck, 2005). This step is currently disabled by default.

8. Perform a soft-sphere clash check and discard clashing loops. By default, an atom's soft-sphere radius is set to 70% of its Van-der-Waal's radius. All heavy atoms in the input model are taken into account while, for the loop, only backbone and Cβ atoms are used. Anchor residues are excluded from the clash check. PyFREAD's clash checking algorithm follows the concept of Bugalho and Oliveira (2009) and employs a geometric hash, allowing it to run in essentially constant time. Clash checking can be disabled by the user, if desired.

9. All remaining database loops are sorted by their anchor backbone RMSD and returned to the user. Loop co-ordinates are written to individual PDB files.

10. If no database loops were found, the query loop is extended by one residue at each end and the entire search procedure repeated. This is done up to 3 times, thus extending the input loop by a maximum of 6 residues. This extension procedure accounts for the possibility that the input model may have a non-native conformation. Performing the search with "wrong" anchor structures is likely to yield no database hits. Extending the loop region past these "wrong" anchors and into more conserved regions is more likely to yield results (provided that loops of the required length are present in the database). This length extension procedure does not reduce prediction accuracy, as FREAD's predictions are independent of loop length (Choi and Deane, 2010).

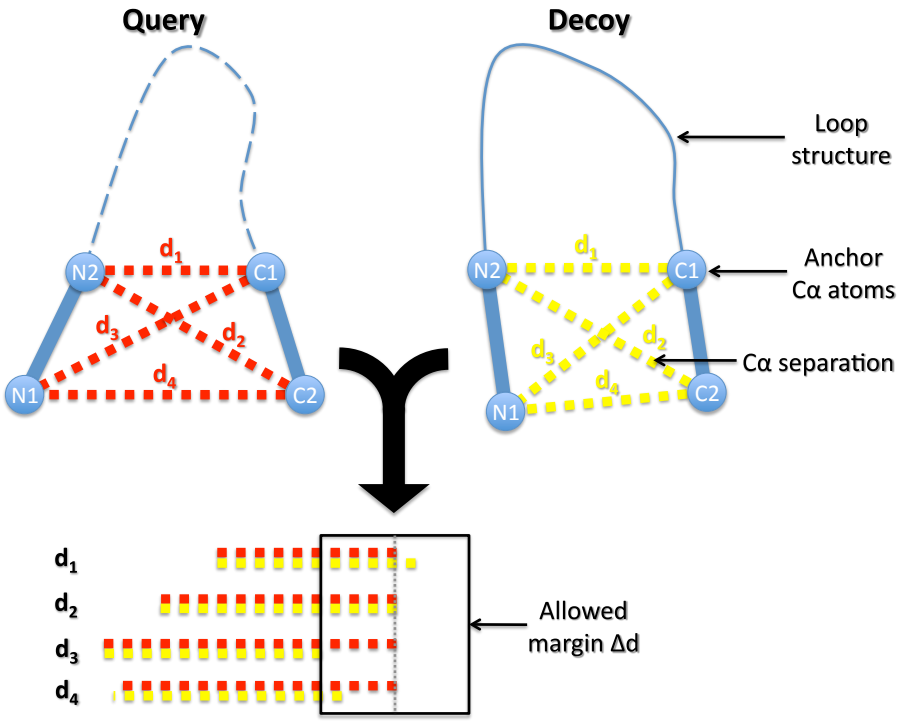


Figure 5.2: Restricting anchor geometry using Cα separations - N1 and N2 are the Cα atoms of the N-terminal anchor, C1 and C2 are those of the C-terminal anchor of a loop. Cα separations are shown as red and yellow dotted lines. The margin Δd by which Cα separations in the decoy (yellow) are allowed to vary from their values in the query (red) are shown by a black rectangle.

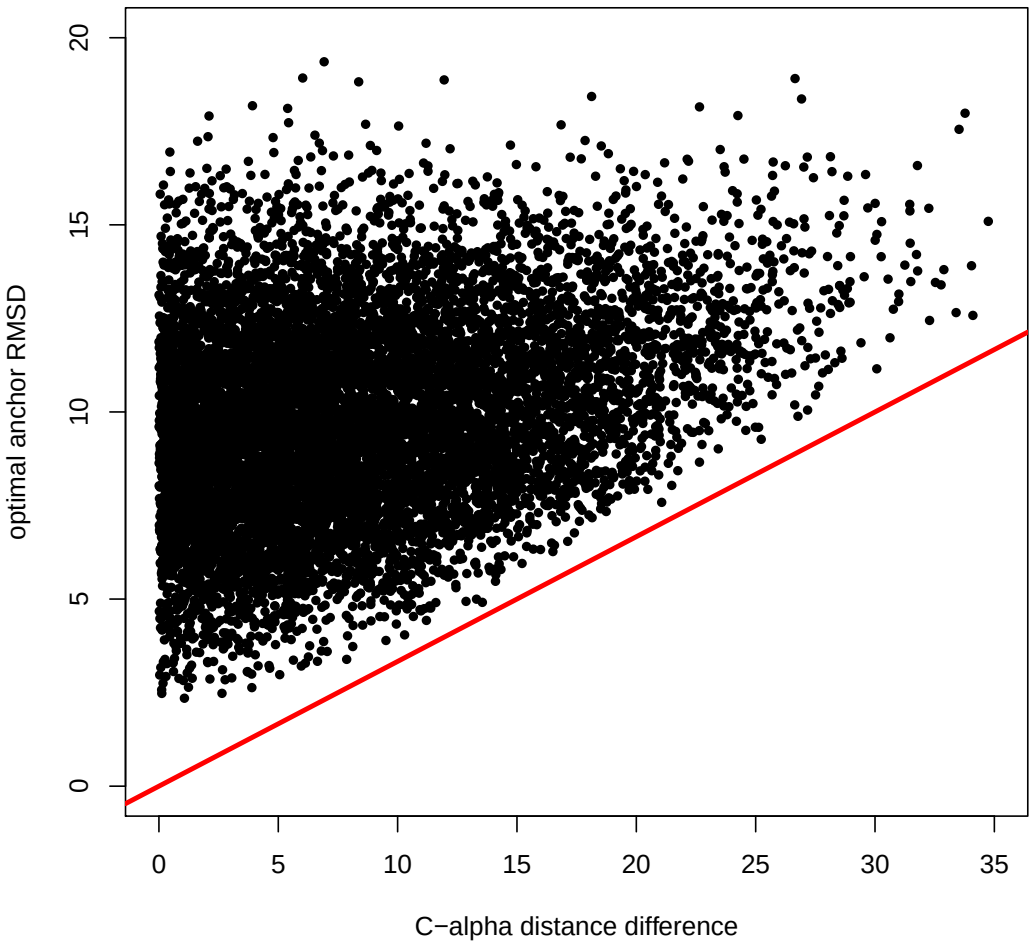


Figure 5.3: Relationship between Cα distance difference and optimal anchor RMSD - 10000 sets of eight random points (p_1, p_2, ..., p_8) were generated in a 3D space of the dimensions ([0,30], [0,30], [0,30]). In each case, the quantity $abs(d(p_1, p_2) - d(p_5, p_6))$ was calculated (X axis), as well as the RMSD between the first four and last four points, after their optimal superposition (Y axis). Conceptually, the first value is equivalent to the difference in Cα distance between anchor co-ordinates of two loops, while the latter value is the optimal Cα RMSD of all anchor residues. The red line has the slope 1/3, showing that the difference in Cα separation is always $< 3 \times optimal\_anchor\_RMSD$. Thus, setting the allowed margin $\Delta d = 3 \times R_{max}$ will result in no loss of coverage.

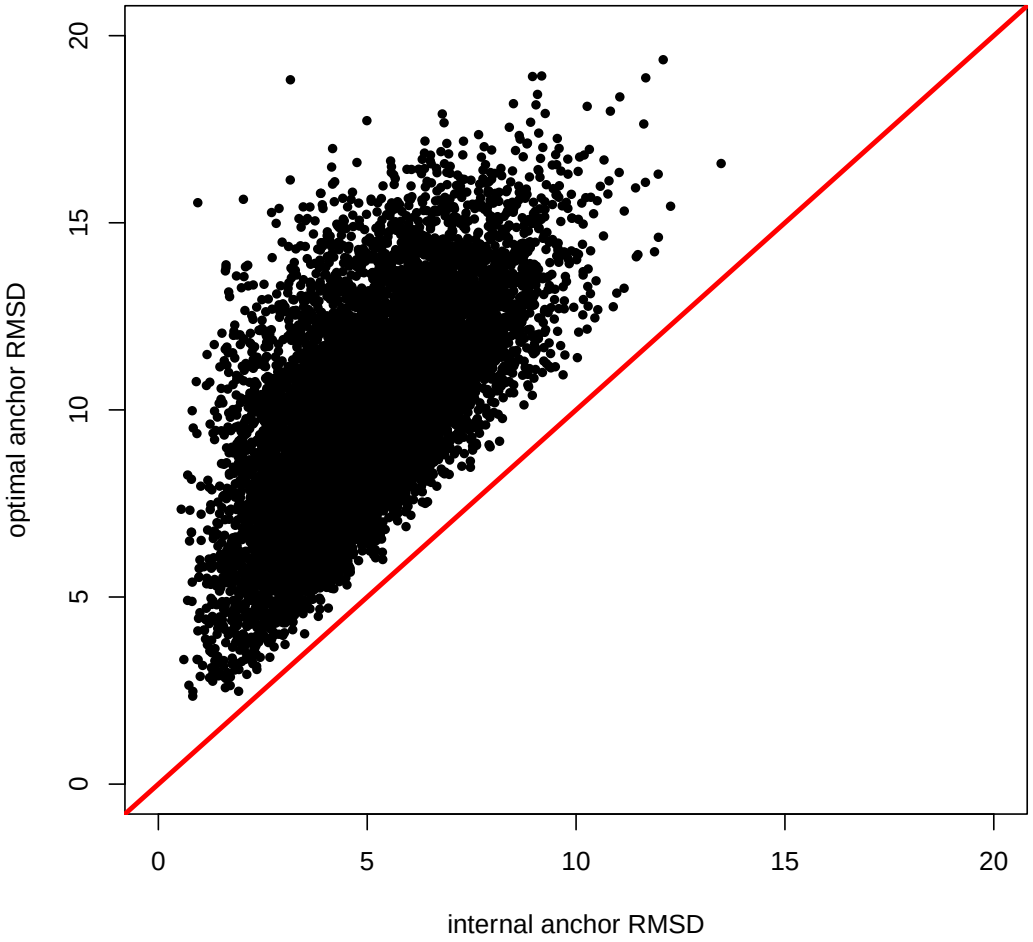


Figure 5.4: Relationship between internal anchor RMSD and optimal anchor RMSD - 10000 sets of eight random points (p_1, p_2, ..., p_8) were generated in a 3D space of the dimensions ([0,30], [0,30], [0,30]), as in Figure 5.3. In each case, the "internal anchor RMSD" was calculated, as well as the standard "optimal anchor RMSD" between the first four and last four points, after their optimal superposition (Y axis). The red line has the slope 1, showing that the internal anchor RMDS is always less than the optimal anchor RMSD. Thus, using the cut-off R_{max} on the internal anchor RMSD will result in no loss of coverage.

5.3.2.3 Adjusting the substitution score cut-off for short loops

For short loops, it is occasionally the case that even a decoy with identical sequence and structure does not satisfy the substitution score cut-off. For example, a four-residue loop consisting only of Alanines (sequence "AAAA") can achieve a maximal score of $S_{max,query} = 4 \times 6 = 24$. With the default cut-off of 25 it is thus impossible to find any matching decoys. Loops shorter than 4 residues would almost always suffer from this problem. To account for such cases, we introduce a simple rule:

$$S_{min} = min(S_{user}, S_{max,query} - L) \tag{5.2}$$

where S_{min} is the cut-off substitution score actually used during the search, S_{user} is the cut-off defined by the user (25 by default), $S_{max,query}$ is the maximum achievable substitution score when the decoy and query have identical sequences, and L is the loop length (number of residues). For simplicity, $S_{max,query}$ is calculated using the TOTAL (environment-independent) substitution table (see Section 3.5.2). L is subtracted from $S_{max,query}$ in order to account for small variations in substitution scores between structural environments.

This rule makes PyFREAD applicable even at small loop lengths, where the original FREAD program, with default parameters, would always have produced zero coverage.

5.3.3 Benchmark

5.3.3.1 Test sets

This study uses two sets of X-ray structures: one containing only soluble proteins, another containing only membrane proteins.

An initial list of potential membrane proteins was created by listing the union of PDB codes contained within the PDB_TM, OPM and CGDB databases and those in the SCOP category "membrane and cell surface proteins and peptides". This list was

then run through the PISCES server (Wang and Dunbrack, 2003), to keep only X-ray structures with resolution $\leq$ 3Å, R factor $\leq$ 0.3 and length $\geq$ 40. The remaining structures were split into component chains and duplicate chains (with 100% sequence identity) were removed. Each structure was then run though iMembrane and only those chains with an iMembrane hit were retained.

Residues annotated by JOY as being anything but helices and sheets were treated as loop residues. Only loops of length >2 and within the membrane, or close to it, were considered. Loops close to the membrane were defined as those which start/end within 4 residues of the nearest TM residue. For each loop length, loops were clustered by sequence identity using UPGMA and made non-redundant at the 40% identity level. From each cluster, the representative loop with the lowest average B factor was chosen. For each loop length from 4 to 17, twenty representative loops were randomly chosen as the test set. Longer loops could not be tested due to the limited number of examples in the database.

A test set of soluble protein loops was obtained from the FREAD website [1]. Loops were clustered at the 40% identity level, as above, and 20 representatives chosen for each loop length from 4 to 17.

Both of the above test sets contain known x-ray structures of proteins, taken from the PDB. In addition, we created a test set of homology models of membrane proteins, which is a subset of the MEDELLER test set explained in Chapter 4. We grouped the 616 pairs of membrane proteins by their target protein. For each target protein we chose the most sequence-similar template, excluding any template above 90% sequence identity (to ensure that there were indeed gaps in most of the models). This resulted in a set of 156 loops from 59 models of varying accuracy.

[1] http://www.stats.ox.ac.uk/~choi/FREAD

5.3.3.2 Databases

The "soluble" database was created by filtering the entire PDB using PISCES, to keep only X-ray structures with resolution ≤ 3Å, R factor ≤ 0.3 and length ≥ 40, and making the resulting list non-redundant at a sequence identity level of 99%. Those chains originating from a protein complex included in the initial list of potential membrane proteins (see previous section) were removed from the "soluble" database.

The "membrane" database was created from the same set of iMembrane-annotated structures as explained in the above section.

The "all" database is the union of the "soluble" and "membrane" databases.

All three databases include entire proteins, not just the loop regions. The first and last 5 residues in each protein were discarded.

5.3.3.3 Substitution tables

Three substitution table sets were tested: the original substitution tables used by FREAD, new tables for soluble protein loops and membrane protein loop-specific tables.

The new sets of ESSTs were created from the membrane and soluble datasets described in Chapter 3 (Section 3.5.1). One major difference from the tables discussed in Chapter 3 is that, for these tables, only substitutions between loop residues are counted. The other main difference lies in the structural environments used to build the tables. Each environment is defined as one particular combination of dihedral angles (ϕ and ψ), as shown in Figure 5.1.

Before generating the substitution tables, homologous sequences with identity <20% to their corresponding protein of known structure were discarded, to avoid errors introduced via low quality alignments. In all remaining sequences, only loops with at most 3 gaps were considered when counting substitutions, to further ensure only well-aligned loops are considered. These parameters were chosen from among a set of similar parameters based on the following reasoning: The original FREAD tables are built by

counting substitutions between proteins of known structure. Because of the lack of known membrane protein structures we adopt a different approach, where we count substitutions between a single structure and multiple homologous sequences (Section 3.5.2). We thus first determine the parameters needed to minimise the difference between our new tables and the original FREAD tables by building several versions of the soluble tables. These are then compared to the original FREAD tables and the one with the smallest difference retained. The same set of parameters is then used to build equivalent tables for membrane proteins.

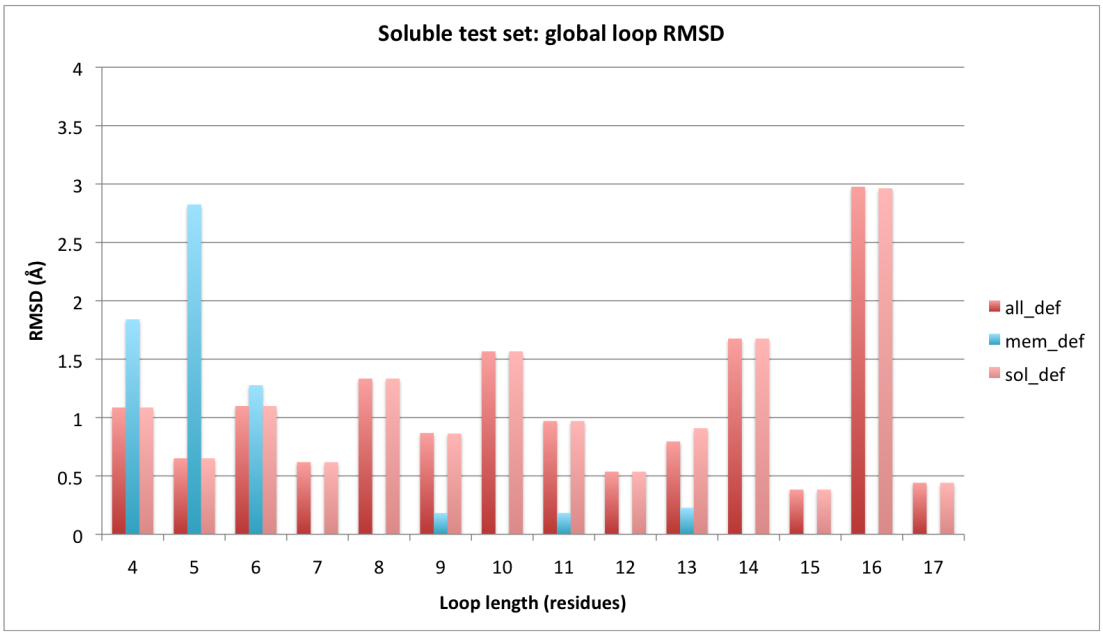
5.4 Results

5.4.1 Database choice matters

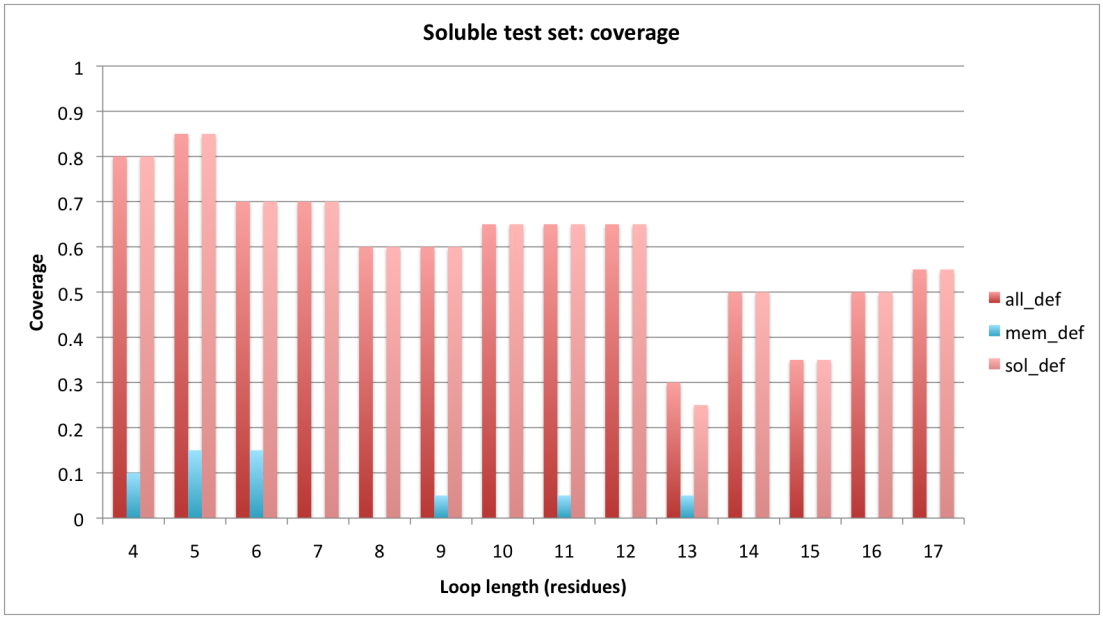
PyFREAD was run on the two test sets (soluble proteins and membrane proteins), using the three types of databases (soluble only, membrane only, or the union of the two). The original FREAD substitution tables were used in this test. The results are shown in Figures 5.5 and 5.6.

In general, we found that using the 'correct' database, corresponding to the test set, resulted in the highest accuracy and coverage, around 1Å RMSD at about 60-80% coverage. Attempting to predict membrane protein loops using a soluble protein database resulted in a rapidly declining coverage, at higher loop lengths, and overall poor accuracy. The reverse scenario, predicting soluble loops with a membrane database, yielded almost zero coverage. Using a union of the soluble and membrane databases (the "all" database) generally resulted in slightly higher coverage and lower accuracy compared to the 'correct' database for each test set.

There thus appears to be a clear difference between the structures of loops in membrane and soluble proteins and one should therefore use membrane protein fragments to predict membrane protein loops.

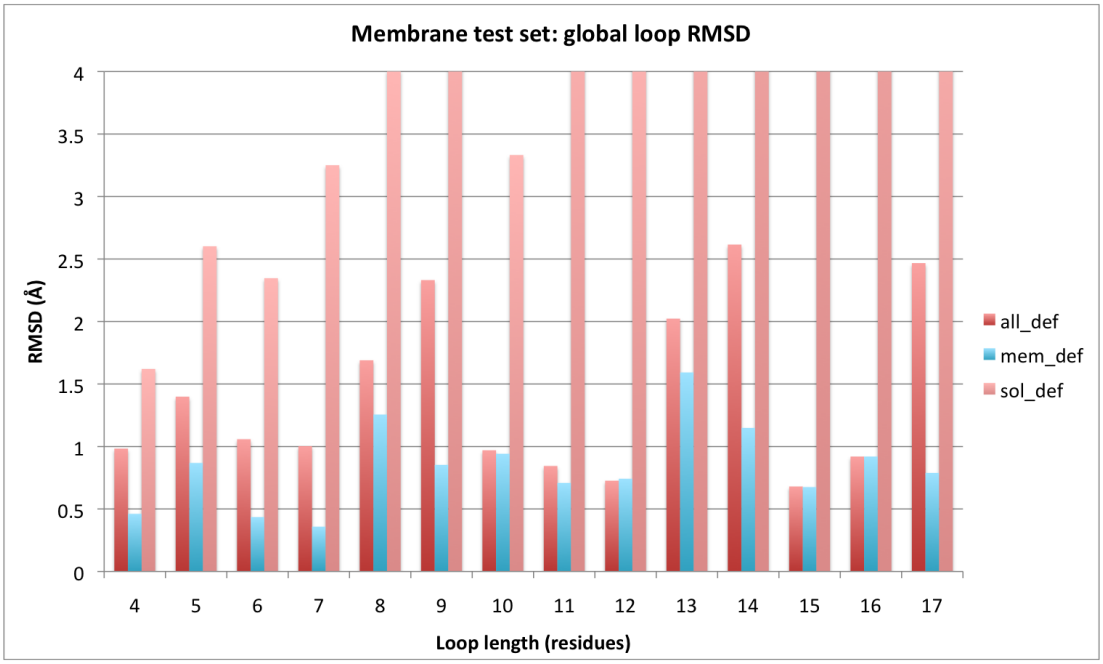


(a)

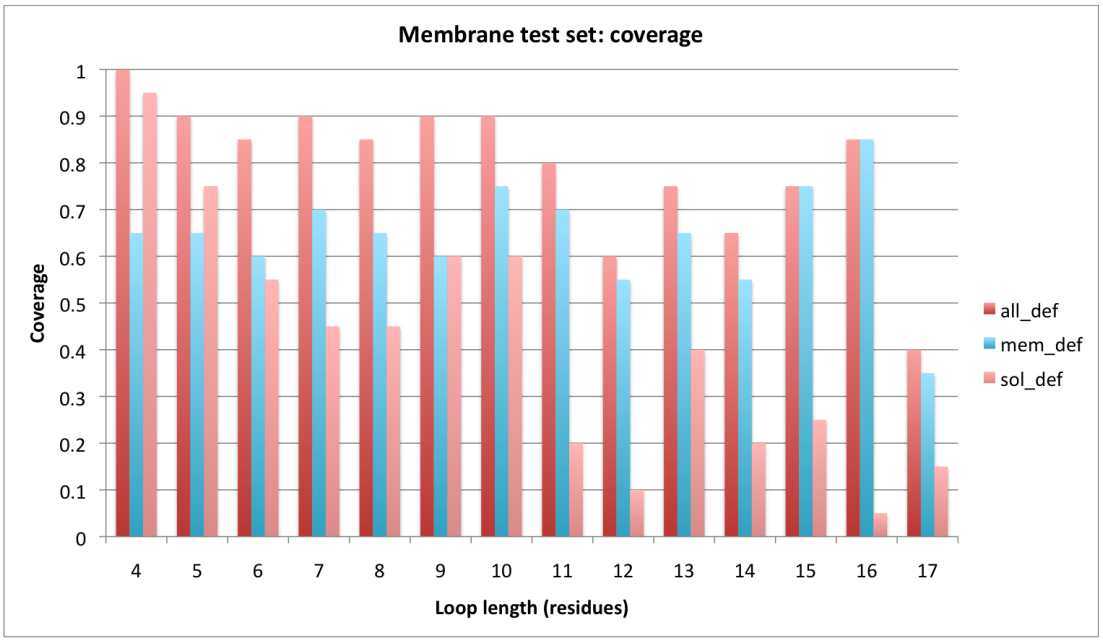


(b)

Figure 5.5: Influence of database choice on (a) accuracy and (b) coverage in soluble proteins - A membrane protein database is not useful in predicting soluble protein loops as it results in almost zero coverage.



(a)



(b)

Figure 5.6: Influence of database choice on (a) accuracy and (b) coverage in membrane proteins - At high loop lengths, use of a soluble protein database results in low coverage. Use of a membrane protein database results in much higher coverage and better accuracy. A union of the two databases achieves slightly higher coverage at a loss of accuracy.

5.4.2 Choice of substitution table type makes little difference

Next, we assess the importance of substitution tables by testing the predictive power using the default ESSTs used in the original FREAD program, a new set of ESSTs created from soluble protein loops and a new set of ESSTs built from membrane protein loops. The structural environments are identical for each set of tables. In this test, only the 'correct' database was used for each test set (i.e. membrane database for membrane test set). The results of the test are shown in Figures 5.7 and 5.8.

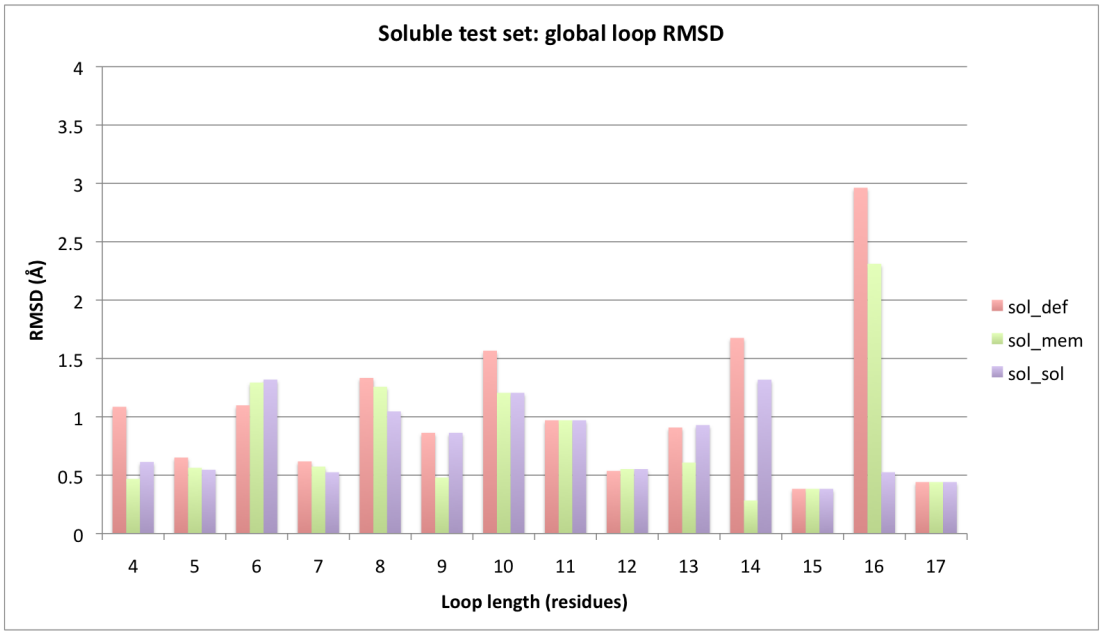
In general, the choice of substitution tables had little influence on the results. Minor variations could be observed, usually due to one or two decoys being included or excluded by one of the ESSTs. The main overall trend was that the newly created tables resulted in slightly lower overall RMSDs, compared to the default FREAD tables. This may be due to table smoothing procedure that forcibly introduces symmetry to each substitution table.

Only very small differences between the results given by the membrane and soluble tables are observed. Overall, loop residues in membrane and soluble proteins seem to have similar substitution patterns. This is not entirely unexpected, considering the current choice of structural environments. In Chapter 3, we observed no obvious difference between the "coil" environments in membrane and soluble proteins. Differences only started to become visible when considering each residue's proximity to the polar head groups of the membrane lipids.

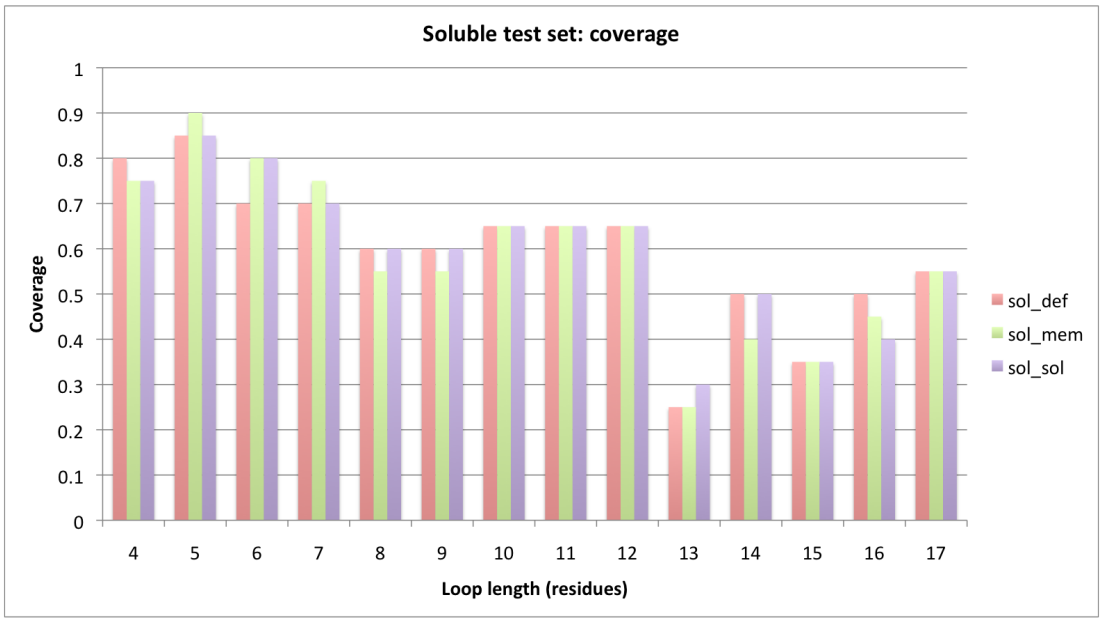
With the current scoring scheme, it makes little difference which of the new sets of ESSTs is used. We therefore chose the soluble ESSTs as the new default for PyFREAD. In subsequent tests, only this one set of tables was used.

5.4.3 Prediction on models of membrane proteins

The previous tests were performed on X-ray structures. While the loop structures were unknown, the anchor structures were known to be correct. In a real modelling

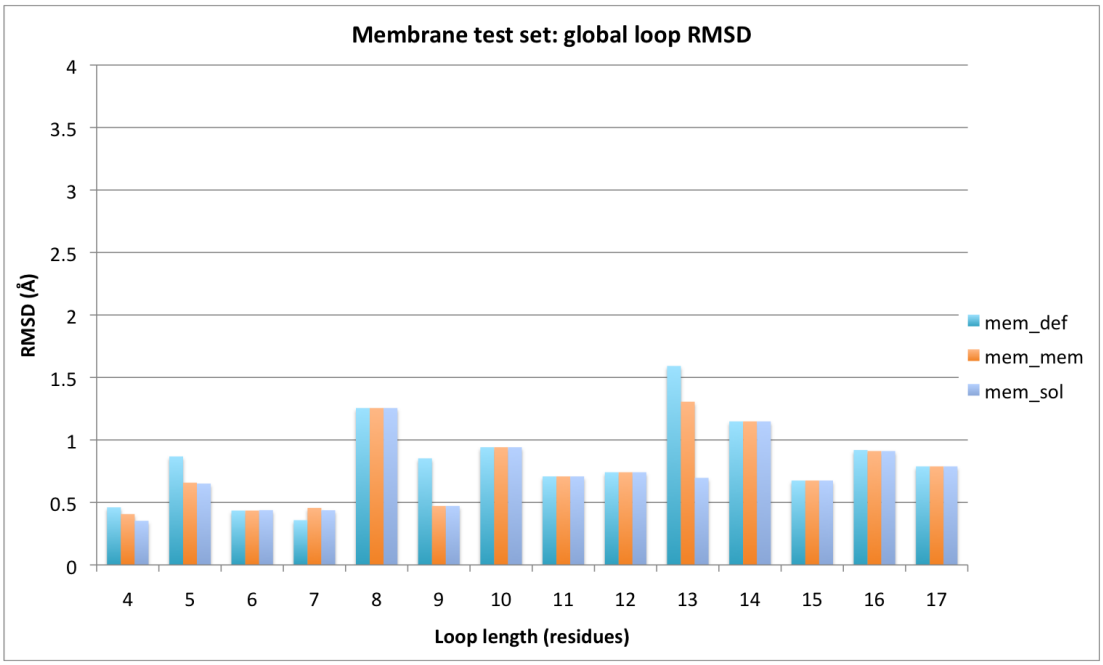


(a)

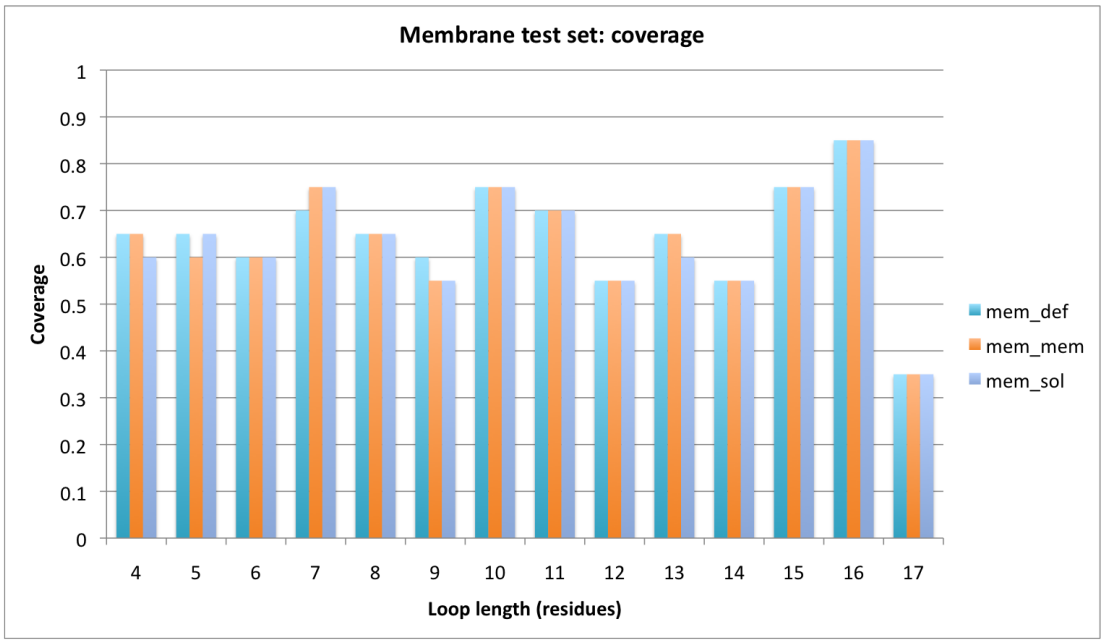


(b)

Figure 5.7: Influence of ESST choice on (a) accuracy and (b) coverage in soluble proteins - Overall, few differences are observed when changing the ESSTs used for decoy scoring. Most differences result from addition or removal of a single loop from the result set. The main trend is that the old FREAD tables ("def") tend to produce slightly worse accuracy, even in cases where coverage is equal or higher than for the other ESSTs. For length 16, only the new soluble tables ("sol") successfully exclude one very low-accuracy decoy.



(a)



(b)

Figure 5.8: Influence of ESST choice on (a) accuracy and (b) coverage in membrane proteins - On the membrane protein test set, the choice of ESSTs makes little difference. The largest change in accuracy is observed at length 13, resulting from a single decoy being excluded by the new soluble ESSTs ("sol").

situation the anchor structures are non-native, making it harder to correctly predict the loop's structure. The loop structure's global RMSD directly depends on the anchor structure's global RMSD (Figure 5.9).

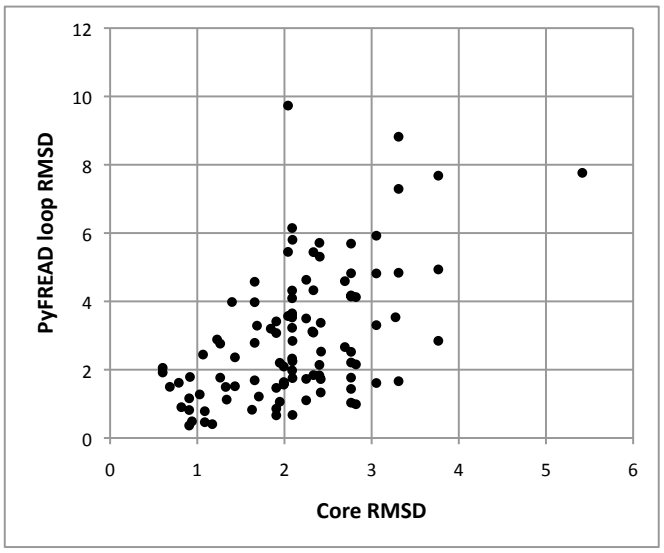


Figure 5.9: Relationship between core model RMSD and loop RMSD - Loop prediction accuracy depends on the accuracy of the loop anchors.

The model test set consists of 156 loops taken from 59 homology models of varying accuracy.

In Chapter 4 FREAD was used to build the loops on MEDELLER's "core" models to produce the "high accuracy" models. Here the same step is performed, while replacing FREAD with PyFREAD. We can thus directly compare the accuracy achieved by FREAD in the earlier study to that achieved by PyFREAD here. In addition, we built Modeller models and compared the accuracy of the corresponding loop co-ordinates. All models were superimposed onto the target's native x-ray structure using only those atoms present in MEDELLER's "core" model. Then the global RMSD was calculated for the loop co-ordinates without re-superimposing them.

Figure 5.10 shows a summary of the test results. All methods created with soluble proteins in mind (Modeller, FREAD and soluble PyFREAD) achieved average global

RMSDs above 6Å. FREAD and Soluble PyFREAD have a coverage of 74/156 and 79/156 – Modeller always gives complete coverage due to being an ab initio method.

Membrane PyFREAD achieves an average accuracy of about 3Å at a coverage of 102/156. With the "all" database, PyFREAD's accuracy is reduced to 4.4Å but coverage is raised to 123/156. See Figure **??** for a plot of Membrane PyFREAD's accuracy against that of Modeller.

Conceptually FREAD's database is more similar to PyFREAD's "all" database, which includes both soluble and membrane proteins. However, in this comparison, PyFREAD clearly outperforms FREAD, both in terms of accuracy and coverage. This is probably due to the higher number of membrane proteins in the PyFREAD database. In addition, the fixed anchor RMSD cut-off (1Å, independent of loop length) may also increase coverage relative to FREAD, where anchor RMSD cut-offs become stricter at lower loop lengths.

	Mod	**FREAD**	**PyF sol**	**PyF all**	**PyF mem**
Avgerage RMSD	6.65Å	6.34Å	6.55Å	4.43Å	**2.93Å**
Loops predicted	**156**	74	79	123	102
Wins/Losses vs Modeller	-	40/34	41/38	87/36	**89/13**
Wins/Losses vs FREAD	34/40	-	27/21	47/22	**44/12**

Figure 5.10: Accuracy and coverage on homology models - Several loop modelling tools were run on 156 loops from 59 homology models of varying accuracy built with MEDELLER. The average backbone RMSD for the core models used as input to the various loop modelling methods was 2.2Å. Abbreviations: Mod=Modeller, PyF=PyFREAD, sol=soluble protein database, mem=membrane protein database, all=union of soluble and membrane databases.

5.4.4 Applicability of the FREAD approach to membrane proteins

In order to verify the validity of PyFREAD's modelling approach, we considered the relationship between accuracy and the similarity of the target protein and the protein providing the loop decoy (Figure 5.12). As previously shown by Choi and Deane (2010) for soluble proteins, there is no obvious correlation between the local similarity of the

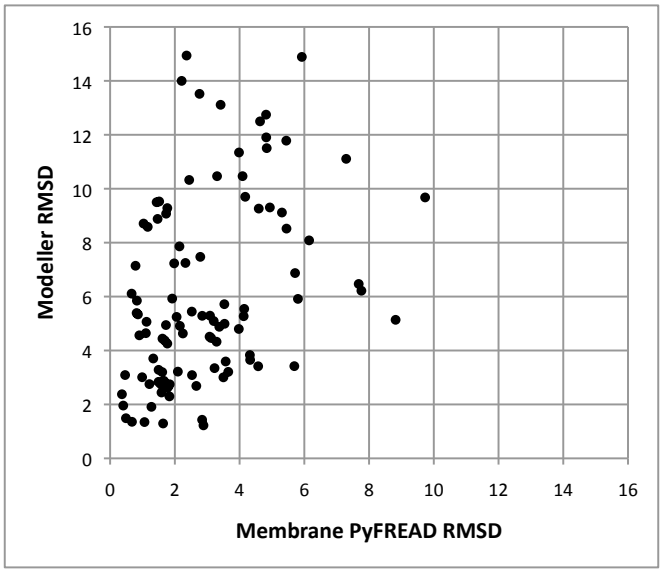


Figure 5.11: Membrane PyFREAD's accuracy versus Modeller's accuracy - Membrane PyFREAD consistently achieves lower RMSDs than Modeller.

loop structure and the global similarity between the two proteins.

However, in the case of PyFREAD on membrane proteins, Figure 5.12 does show that a large amount of loop decoys originate from highly sequence-similar proteins. We thus re-computed the benchmark results while excluding all hits to proteins with over 90% sequence similarity. The average accuracy remained similar (3.56Å average RMSD), but the coverage was reduced to 42/156. This reflects the fact that the currently known membrane protein structures are largely clustered into dense groups that are highly similar within-cluster but very different across-cluster. In other words, the effective number of currently known membrane protein folds is low and only covers small discrete parts of the available fold space. Until the number of known membrane protein folds increases, it may thus be hard to predict loops of newly discovered families of membrane proteins with the methodology presented here.

When the FREAD algorithm was first invented (Deane and Blundell, 2001), the situation was similar with regards to soluble proteins. Coverage was low, making it necessary to combine FREAD with ab initio loop modelling methods. Ten years later,

FREAD was shown to be the most accurate loop modelling method available (Choi and Deane, 2010). The results presented here suggest that the same may happen with regards to the prediction of membrane protein loops.

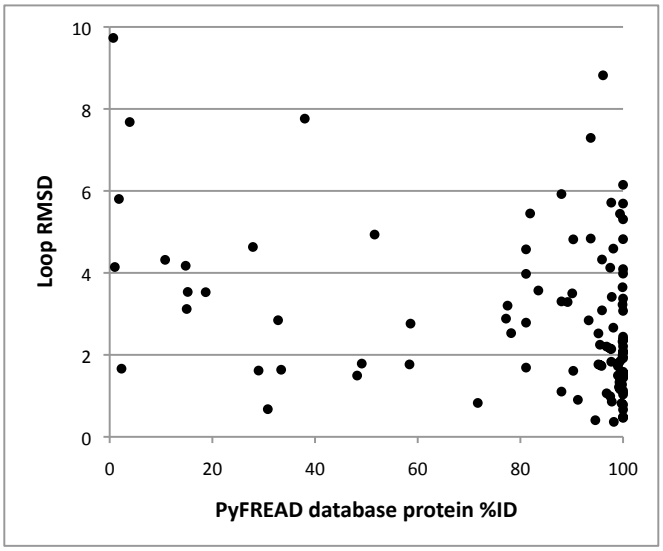


Figure 5.12: Relationship between local and global structural similarity - PyFREAD loop RMSD (membrane database) is plotted against the percentage sequence identity between the target protein and the protein from which the decoy loop was taken. There is no obvious correlation between the two metrics. However, it is obvious that many of the loops predicted by PyFREAD originate from highly similar proteins to the target.

5.5 Speed

PyFREAD's current implementation takes about 10 seconds to model a single soluble loop (by searching the soluble database) and 1 second for a membrane loop (by searching the membrane database). If no database hit is found, the loop region is extended and the search repeated up to 3 times, resulting in a maximal runtime of 40 and 4 seconds, respectively. This is an order of magnitude faster than the original implementation of FREAD and many orders of magnitude faster than the most accurate ab initio methods.

5.6 Conclusions

We have developed PyFREAD, a fast and accurate method for loop prediction, founded upon the principles of the FREAD fragment database search algorithm. We used the PyFREAD program to predict loops in soluble proteins and membrane proteins. Our results showed that, to achieve good accuracies, the type of proteins included in the fragment database should match the type of protein being modelled. The choice of substitution tables (built either from soluble or membrane proteins), on the other hand, had little effect, given the current structural environment definitions.

PyFREAD achieves better accuracies on membrane protein loops than any of the other methods tested.

Modelling accuracy is especially important in the loop regions of membrane proteins, which are hardest to model, even when the target and template are closely related. As a part of the MEDELLER modelling pipeline, PyFREAD will permit more accurate models of membrane proteins to be built.

As only a small number of membrane protein structures are known the applicability of the approach may still be limited. This is likely to change over the next few years, as the number of membrane protein structures increases.

5.7 What's next

The results presented in this chapter suggest that membrane protein loops differ in structure from soluble protein loops. Being able to characterise this difference would enable us to predict loops from a soluble protein database and filter the results in order to exclude loops that do not resemble a typical membrane protein loop. If successful, this would increase our method's applicability to membrane proteins and circumvent the coverage problem.

Like all loop modelling methods, PyFREAD's accuracy is dependent on the anchor

structures of the framework model. In membrane proteins, even the most conserved part of the target protein often contains small structural variations, compared to the template. In helical membrane proteins these variations appear in the form of helix kinks and twists. In the MEDELLER approach, this causes errors in the core model resulting in poor anchor orientation and wrongly oriented loops. Detecting such errors and correcting them in a membrane protein-specific model refinement step should further improve the overall quality of MEDELLER models.

Chapter 6

Conclusions and Future Directions

We have created a toolkit that accurately models the structure of the transmembrane domains of proteins.

6.1 iMembrane: Inserting proteins into a lipid bilayer

iMembrane is a fast, accurate way to obtain a membrane protein's orientation within the lipid bilayer. It provides simple "membrane layer" annotation as well as more detailed "membrane contact" annotation on a per-residue basis. By combining these two types of information with classical surface accessibility annotation one can build a complete picture of a membrane protein's orientation in the membrane. iMembrane has an intuitive user interface and is freely available online[1] to academic users. This work has been published in the journal *Bioinformatics* (Kelm et al., 2009).

If a structure is known, the iMembrane database (iMemDB) can be searched for a similarly shaped membrane protein using structure alignment tools. The query protein

[1] iMembrane web server: http://imembrane.info

is then annotated in terms of its membrane insertion based on the database protein's known annotation.

If no structure is known, iMemDB can instead be searched using sequence in order to find a suitable homologous protein for use as a modelling template. Currently this step is performed using a PSI-BLAST search and/or pairwise sequence alignment with MUSCLE, a method which uses no structural information. In Hill et al. (2011) we have demonstrated that markedly higher alignment accuracies can be achieved when using a sequence-to-structure alignment method utilising membrane protein-specific information. Integrating such a method into iMembrane would make it a suitable tool for template identification and target-template alignment.

Currently, iMembrane operates on a single-chain basis. Each database entry consists of a single protein chain and the database search is performed with a single input chain. In order to annotate whole complexes of membrane proteins it may be worth exploring the use of complex-to-complex alignment methods (such as, for example, MM-align (Mukherjee and Zhang, 2009)). Annotation of the whole complex may also involve a consensus method that uses the annotation of each individual chain to infer a more general annotation of the entire complex.

In addition to improving the search algorithms used by iMembrane, the database itself could be made browsable via a user-friendly web interface or as a downloadable archive to increase further its usefulness to the scientific community.

6.2 Environment-specific substitution tables of membrane proteins

Based on the structural environments defined by iMembrane, we have built environment-specific substitution tables (ESSTs) for membrane proteins. This was done using our

software JSUBST, which is freely available online[1]. We have shown that substitution patterns differ between membrane and non-membrane environments. In addition, we have confirmed that substitutions in soluble portions of membrane proteins resemble those in soluble proteins.

We have developed a method for comparing structural environments to one another through the comparison of substitution tables. This procedure is entirely automatable and could be used to make decisions about which combination of environments to use when creating substitution tables for a particular set of proteins.

In collaboration with Jamie R. Hill we have applied our new substitution tables to sequence-to-structure alignment of membrane proteins and have demonstrated increased accuracy compared to the traditional soluble protein tables. These better alignments directly translate into 3D models of higher accuracy. This work has been published in the ISMB 2011 conference proceedings (Hill et al., 2011).

The studies presented in this thesis have yielded a very clear separation of membrane and non-membrane environments when clustering structural environments by their substitution patterns, something not always observed by *Hill et al.*, due to the simpler membrane model used there. It may thus be possible to obtain a further increase in alignment accuracy by taking into account this information. In addition, further improvements may be achieved through an iterative approach to the creation of substitution tables, as shown in Figure 6.1.

6.3 MEDELLER: co-ordinate generation for membrane proteins

Our template-based co-ordinate generation method MEDELLER builds highly reliable core models of transmembrane domains. This is achieved through the use of the

[1]JSUBST and substitution table download: http://www.stats.ox.ac.uk/proteins/resources

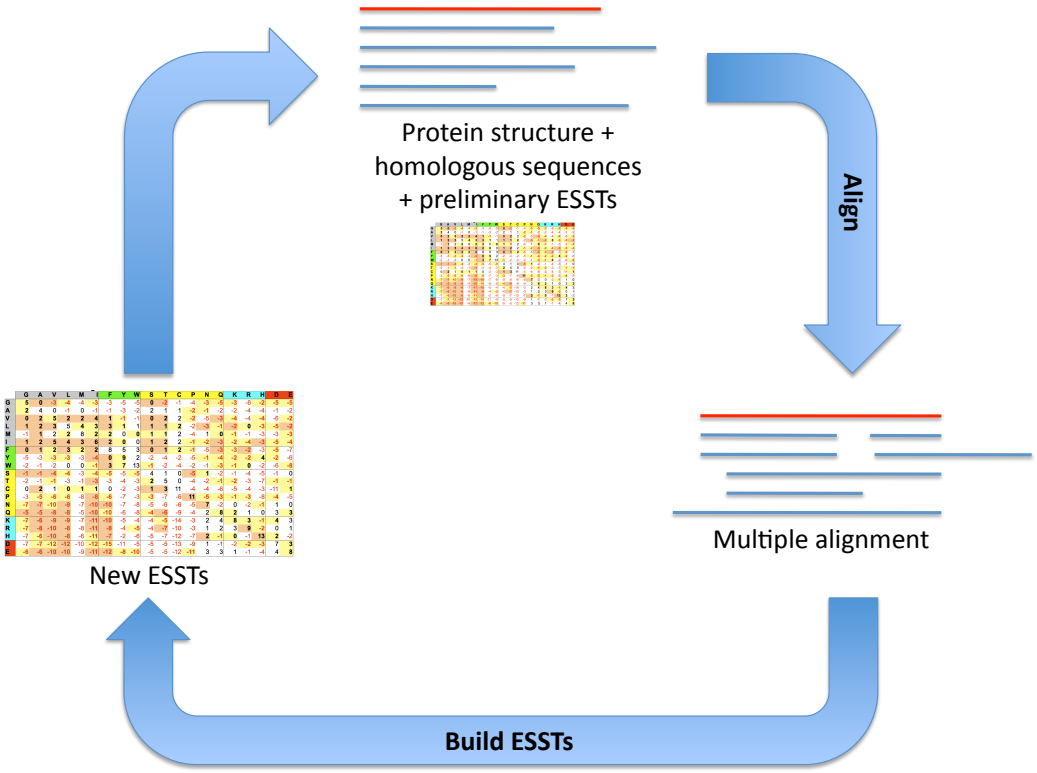


Figure 6.1: Iterative table building procedure - The input to the procedure is a dataset containing protein structures, each associated with a number of homologous sequences. We also have a pre-existing set of ESSTs. These ESSTs are used to align each set of homologous sequences to their corresponding structure. From the resulting set of multiple alignments, new ESSTs are built. These can now be used to create new multiple alignments, etc. The procedure would end when the ESSTs no longer change from one iteration to the next.

membrane protein-specific information from our iMembrane annotation, in order to choose the correct ESST to score each residue in an input alignment. This score informs a rule-based algorithm that grows the model's "core" from the centre of the membrane outwards, towards the less reliable loop regions of the template structure. Residues with a low score are deemed unreliable and are discarded. Missing segments are then modelled using the FREAD database search algorithm. MEDELLER shows a clear improvement in accuracy over the popular program Modeller. The user-friendly MEDELLER web server[1] is freely accessible to academic users. This work has been published in the journal *Bioinformatics* (Kelm et al., 2010).

As MEDELLER was built before our detailed study of ESSTs, the first step in improving MEDELLER should be to replace the original substitution tables with equivalent newer ones. This should result in slightly more consistent scoring of rare substitutions.

The other improvement to MEDELLER is the replacement of the FREAD loop modelling program by our new implementation, PyFREAD. As described in Chapter 5, loop models should initially be selected from membrane structures but, due to limited coverage, it may be useful to also add the option of searching a database of soluble protein loops.

Currently, MEDELLER does not employ any model refinement. Developing a function that could deal with the local differences between template and target, such as helix kinks and twists, would result in better core models. Since loop modelling accuracy directly depends on the accuracy of the anchor structures, this would translate into better loop models.

Another possible extension to MEDELLER would be the support of multi-chain modelling. Protein function happens at the complex level – individual chains might not even be stable in the membrane. Obtaining whole complex models would enable

[1]MEDELLER web server: http://medeller.info

functional and interaction studies that would not be feasible with single chain models. A straight forward way to implement this would be to use a whole protein complex as a modelling template. This is already possible with MEDELLER's current implementation. However, this approach is not applicable when the complex structure of a homologous protein is not available. In this case, the integration of various types of experimental information as well as ab initio prediction methods may be required to infer the structure of the macromolecular assembly (Alber et al., 2007). This is a complex and compute-intensive process.

Currently, MEDELLER only models transmembrane domains. However, its approach is readily applicable to any type of protein or domain. Making the program applicable to other types of proteins would increase its usefulness to the scientific community. More specifically, it would allow the modelling of soluble domains of membrane proteins, or separate protein chains associated with membrane proteins, an essential requirement when aiming to model entire membrane protein complexes.

6.4 PyFREAD: high-speed loop modelling for membrane and soluble proteins

PyFREAD is our new and improved implementation of the FREAD loop modelling algorithm. Using PyFREAD, we have shown that membrane protein loops are not easy to predict when using a database of only soluble proteins and attempts to do so result in low accuracies and declining coverage at increasing loop lengths. Conversely, predicting soluble protein loops with a membrane protein database results in near-zero coverage. However, when using the "correct" database (i.e. a database of membrane proteins when modelling membrane protein loops), accuracies of about 1Å global RMSD can be achieved, on native anchor structures.

We further tested our approach by running it on a set of homology models of

varying accuracy and comparing it to prediction using a soluble database, to the old implementation of FREAD and to Modeller. Membrane PyFREAD was the most accurate method.

The major drawback of our approach is its reliance on a database of membrane proteins. Due to the low number of currently known membrane protein structures, a relatively low coverage (about 45%) is achieved as soon as highly homologous proteins (to the query) are excluded from the database. Accuracy stays high, indicating that our approach is valid. However, the reduction in coverage means that modelling loops on a protein of previously unknown fold is likely to fail, simply because no usable loops are yet present in the database. The same problem was encountered when FREAD was first written for soluble proteins, in 2001. It is likely that this issue will resolve itself within the next decade as the number of known membrane protein structures increases.

The obvious inability of one type of protein loop to replace the other indicates a difference in the typical structures of soluble and membrane protein loops. Characterising this difference may enable us to mitigate the coverage problem by filtering the soluble database to leave only those loops that "look like" a typical membrane protein loop.

6.5 Final words

We have developed a suite of tools to build accurate 3D models of transmembrane domains. All our software was developed to be fast, easy to use and more accurate than previously available methods. All the above tools are, or will shortly be, available online and some already enjoy an active and growing user base (iMembrane has had users from 39 countries world wide and over 60 visits within the single month of May 2011).

Chapter 7

Appendix

7.1 MUSCLE

Overview of the MUSCLE algorithm:

1. Draft progressive alignment

   (a) Calculate the distance between all input sequences using a fast heuristic distance measure, D_{kmer}, based on k-mers (a.k.a. words) appearing in both sequences. This produces distance matrix $D1$.

$$D_{kmer} = 1 - F \tag{7.1}$$

   where

$$F = \sum_{T} d_{XY}(T)/[min(L_x, L_y) - k + 1] \tag{7.2}$$

$$d_{XY}(T) = min[n_x(T), n_y(T)] \tag{7.3}$$

   T is a k-mer (a sub-sequence of length k); X and Y are the two sequences being compared; $n_x(T)$ is the number of times that the k-mer T was found in sequence X; L_x is the length of sequence X. d_{XY} is the maximum number of times that the k-mer T could potentially be aligned in the two sequences. A decrease in time and space complexity can be obtained by using a binary measure for of whether a k-mer appears in both sequences, instead of using

an integer count. In this case,

$$d_{XY}(T) = 1 \tag{7.4}$$

if T appears in both sequences X and Y; otherwise

$$d_{XY}(T) = 0 \tag{7.5}$$

(b) Cluster D1 using UPGMA (Sneath and Sokal, 1973) to produce binary tree TREE1.

(c) Perform a progressive alignment using TREE1 as the guide tree, producing multiple alignment MSA1.

2. Improved progressive alignment

(a) Calculate the distance between all sequences using their Kimura distances as calculated from the current MSA (in the first iteration, this is MSA1). The Kimura distance D_{Kimura} is based on percentage sequence identity, but includes a correction for the possibility of multiple substitutions at the same site. This produces distance matrix D2.

$$D_{Kimura} = -log_e(1DD^2/5) \tag{7.6}$$

(b) Cluster D2 using UPGMA to produce binary tree TREE2.

(c) Perform a new progressive alignment using TREE2 as the guide tree, producing multiple alignment MSA2.

(d) Iterate until TREE2 no longer changes.

3. Refinement

(a) Choose a single edge from TREE2. Edges are chosen starting from the leaves and moving towards the root.

(b) Choose a single edge from TREE2. Edges are chosen starting from the leaves and moving towards the root.

(c) Delete the edge, thus dividing the tree into two sub-trees. Calculate the sequence profile representing each sub-tree.

(d) Re-align the two profiles, producing a new complete MSA.

(e) Calculate the sum of pairs (SP) score for the new MSA. If the score is higher than the one for the previous MSA, accept the change, otherwise reject it.

(f) Iterate until convergence or until a user-defined number of iterations. This method is a variant of tree-dependent restricted partitioning (Hirosawa et al., 1995).

7.2 JOY

Full list of annotation types created by JOY:

- secondary structure and phi angle
- solvent accessibility
- hydrogen bond to mainchain CO
- hydrogen bond to mainchain NH
- hydrogen bond to other sidechain/heterogen
- cis-peptide bond
- hydrogen bond to heterogen
- covalent bond to heterogen
- disulphide
- mainchain to mainchain hydrogen bonds (amide)
- Mainchain to mainchain hydrogen bonds (carbonyl)
- DSSP
- positive phi angle
- percentage accessibility
- Ooi number

7.3 iMembrane

See Figure 7.1 for iMembrane's Q2 accuracy plots.

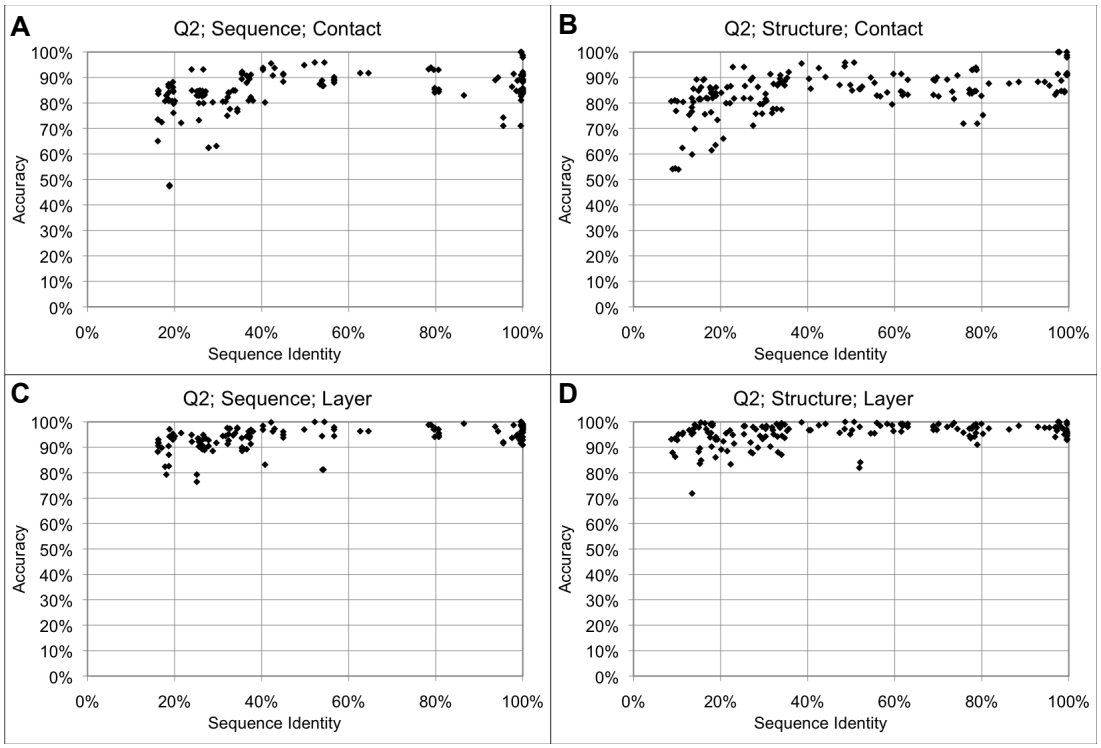


Figure 7.1: iMembrane Q2 accuracy - The X-axis shows the percentage sequence identity between the input and database proteins, as calculated from their structure alignment. The Y-axis shows the Q2 accuracy, i.e. the percentage of residues annotated correctly by iMembrane as either "N" or anything else ("H" and "T" are treated as being equivalent for the purpose of this test). **A+B:** *Membrane contact* annotation; **C+D:** *Membrane layer* annotation; **A+C:** Sequence input; **B+D:** Structure input. This figure is the Q2 equivalent to Figure 2.10.

7.4 MEDELLER

7.4.1 Target-template pairs

The format is TARGET_TEMPLATE, where TARGET is the target protein's PDB code followed by the chain code (twice), and TEMPLATE is the template protein's PDB code, followed by its CGDB simulation number, the chain identifier and a number distinguishing between duplicate chains in the same CGDB simulation (i.e. if the

protein is part of a homo-oligomer).

7.4.1.1 Easy test set

1A0SPP_1A0T0P1 1J4NAA_3D9S0A1 1QLECC_1OCC0C1 2BHWAA_1RWT0B1
1A0SPP_1A0T0Q1 1J4NAA_3D9S0B1 1QLECC_1OCC0P1 2BHWAA_1RWT0C1
1A0SPP_1A0T0R1 1J4NAA_3D9S0C1 1U19AA_3CAP0A1 2E74AA_1VF50A1
1AF6AA_2MPR0A1 1J4NAA_3D9S0D1 1VF5DD_1VF50Q1 2E74AA_1VF50N1
1AF6AA_2MPR0B1 1KPLBB_1KPL0A1 1WP1BB_1WP10A1 2E74BB_1VF50B1
1AF6AA_2MPR0C1 1KPLBB_1OTS0A1 1XL6AA_1XL40B1 2E74BB_1VF50O1
1AIGLL_1EYS0L1 1KPLBB_1OTS0B1 1XMEAA_1EHK0A1 2E74CC_1VF50C1
1AIGLL_1OGV6L1 1LGHBB_1LGH0E2 1YCEaa_1YCE0A1 2E74CC_1VF50P1
1AR1BB_1QLE0B1 1M56AA_1OCC0A1 1YCEaa_1YCE0B1 2EI4AA_1C3W0A1
1AR1BB_2GSM0B1 1M56AA_1OCC0N1 1YCEaa_1YCE0C1 2EI4AA_1VGO0A1
1BE3CC_1BGY0C1 1M56AA_1QLE0A1 1YCEaa_1YCE0D1 2EVUAA_2F2B0A1
1BE3CC_1BGY0O1 1M56AA_2GSM0A1 1YCEaa_1YCE0E1 2FBWCC_1ZOY0C1
1BE3CC_1EZV0C1 1M56BB_1QLE0B1 1YCEaa_1YCE0F1 2GSKAA_1NQE0A1
1BE3GG_1BGY0G1 1M56BB_2GSM0B1 1YCEaa_1YCE0G1 2GUFAA_1NQE0A1
1BE3GG_1BGY0S1 1M56CC_1OCC0C1 1YCEaa_1YCE0H1 2HYDAA_2HYD0B1
1BE3JJ_1BGY0J1 1M56CC_1OCC0P1 1YCEaa_1YCE0I1 2IXXAA_2J1N0A1
1BE3JJ_1BGY0V1 1M56CC_1QLE0C1 1YCEaa_1YCE0J1 2IXXAA_2J1N0B1
1DXRLL_1EYS0L1 1NEKCC_1NEN0C2 1YCEaa_1YCE0K1 2IXXAA_2J1N0C1
1DXRLL_1OGV6L1 1NKZAA_1NKZ0C3 1YEWAA_1YEW0E1 2IXXAA_2OMF0A2
1DXRMM_1EYS0M1 1NKZAA_1NKZ0E1 1YEWAA_1YEW0I1 2JAFAA_1E126A1
1EYSLL_1OGV6L1 1NKZBB_1NKZ0D3 1YEWBB_1YEW0F1 2NQ2AA_2NQ20B1
1EZVCC_1BGY0C1 1OCCAA_1OCC0N1 1YEWBB_1YEW0J1 2VT4BB_2VT40A1
1EZVCC_1BGY0O1 1OCCAA_1QLE0A1 1YEWCC_1YEW0G1 3CX5DD_1BGY0D1
1H6S11_1PRN0A1 1OCCAA_2GSM0A1 1YEWCC_1YEW0K1 3CX5DD_1BGY0P1
1HXXAA_2J1N0A1 1OCCBB_1OCC0O1 1YQ3DD_1ZOY0D1 3CX5DD_1EZV0D1
1HXXAA_2J1N0B1 1OCCCC_1OCC0P1 2B6PAA_1J4N0A1 3CX5II_1EZV0I1
1HXXAA_2J1N0C1 1OCCCC_1QLE0C1 2B6PAA_1YMG0A1 3D31CC_3D310D1
1HXXAA_2OMF0A3 1OGVMM_1EYS0M1 2B6PAA_3D9S0A1 3D9SAA_1J4N0A3
1IJDAA_1NKZ0C3 1PY6AA_1C3W0A1 2B6PAA_3D9S0B1 3D9SAA_1YMG0A1
1IJDBB_1NKZ0B3 1PY6AA_1VGO0A1 2B6PAA_3D9S0C1 3D9SAA_3D9S0B1
1IJDBB_1NKZ0D2 1PY6AA_2EI40A1 2B6PAA_3D9S0D1 3D9SAA_3D9S0C1
1J4NAA_1YMG0A1 1QFGAA_2FCP0A1 2BHWAA_1RWT0A1 3D9SAA_3D9S0D1

7.4.1.2 Medium test set

1A0SPP_2MPR0A1 1AF6AA_1A0T0R1 1AR1BB_1OCC0O1 1DXRMM_1OGV0L1
1A0SPP_2MPR0C1 1AIGLL_1EYS0M1 1BE3JJ_1EZV0I2 1EYSLL_1EYS0M1
1AF6AA_1A0T0P1 1AR1BB_1OCC0B1 1DXRLL_1EYS0M1 1EYSMM_1EYS0L1

1EYSMM_1OGV1L1	1M56BB_1OCC0B1	1Z98AA_3D9S0C1	2O9DBB_2F2B0A4
1FX8AA_1J4N0A4	1M56BB_1OCC0O1	1Z98AA_3D9S0D1	2O9DBB_3C020A2
1FX8AA_1YMG0A2	1M56DD_1QLE0D1	2B6PAA_1FX80A3	2O9DBB_3D9S0A1
1FX8AA_1Z980A3	1NKZAA_1LGH0D4	2B6PAA_1Z980A4	2O9DBB_3D9S0B1
1FX8AA_2F2B0A3	1NKZBB_1LGH0B1	2B6PAA_2F2B0A1	2O9DBB_3D9S0C1
1FX8AA_3C020A4	1NKZBB_1NKZ0F3	2EI4AA_1E121A1	2O9DBB_3D9S0D1
1FX8AA_3D9S0A1	1OCCBB_1QLE0B1	2EI4AA_1H2S0A1	2ODJBB_2QTK0A1
1FX8AA_3D9S0B1	1OCCBB_2GSM0B1	2EVUAA_1FX80A4	2PORAA_1PRN0A1
1FX8AA_3D9S0C1	1OGVMM_1EYS0L1	2EVUAA_1J4N0A3	2QKSAA_1XL40B1
1FX8AA_3D9S0D1	1OGVMM_1OGV1L1	2EVUAA_1YMG0A2	3C02AA_1FX80A4
1H2SAA_1C3W0A1	1PY6AA_1E121A1	2EVUAA_1Z982A4	3C02AA_1J4N0A3
1H2SAA_1E121A1	1PY6AA_1H2S0A1	2EVUAA_3C020A2	3C02AA_1YMG0A2
1H2SAA_1VGO0A1	1SU4AA_1XP50A1	2EVUAA_3D9S0A1	3C02AA_2F2B0A3
1H2SAA_2EI40A1	1U77AA_2B2F0A2	2EVUAA_3D9S0B1	3C02AA_3D9S0A1
1H6S11_2POR0A3	1XIOAA_1C3W0A1	2EVUAA_3D9S0C1	3C02AA_3D9S0B1
1IJDAA_1LGH0A1	1XIOAA_1E122A1	2EVUAA_3D9S0D1	3C02AA_3D9S0C1
1IJDAA_1LGH0D4	1XIOAA_1H2S0A1	2HDIAA_1NQE0A1	3C02AA_3D9S0D1
1IJDAA_1NKZ0A1	1XIOAA_1VGO0A1	2JAFAA_1C3W0A1	3CX5II_1BGY0V1
1IJDAA_1NKZ0E1	1XIOAA_2EI40A1	2JAFAA_1H2S0A1	3D9SAA_1FX80A3
1IJDBB_1LGH0B2	1YC9AA_1WP10A2	2JAFAA_1VGO0A1	3D9SAA_1Z983A1
1IJDBB_1LGH0E2	1Z98AA_1FX80A4	2JAFAA_2EI40A1	3D9SAA_2F2B0A1
1J4NAA_1FX80A4	1Z98AA_1J4N0A3	2NR9AA_2IC80A1	3D9SAA_3C020A2
1J4NAA_1Z987A1	1Z98AA_1YMG0A2	2O9DBB_1FX80A3	3DDLAA_1VGO0A1
1J4NAA_2F2B0A1	1Z98AA_2F2B0A1	2O9DBB_1J4N0A3	3DDLAA_2EI40A1
1LGHBB_1NKZ0B3	1Z98AA_3D9S0A1	2O9DBB_1YMG0A2	3DWOXX_1T160A1
1LGHBB_1NKZ0D3	1Z98AA_3D9S0B1	2O9DBB_1Z982A4	

7.4.1.3 Hard test set

1A0SPP_2MPR0B1	1HXXAA_2POR0A3	1NKZBB_1VF50S1	1XKWAA_2FCP0A1
1AF6AA_1A0T0Q1	1IJDAA_1EHK0C1	1OCCAA_1EHK0A1	1XMEAA_1OCC0A1
1BE3GG_1EZV0G1	1IJDBB_1EHK0C1	1QFGAA_1KMO0A1	1XMEAA_1OCC0N1
1DXRMM_1EYS0L1	1J4NAA_3C020A2	1QFGAA_1NQE0A1	1XMEAA_1QLE0A1
1EZVEE_1VF50D1	1KMOAA_1NQE0A1	1QFGAA_1XKW0A1	1XMEAA_2GSM0A1
1EZVEE_1VF50Q1	1KMOAA_1XKW0A1	1QLEDD_1NKZ0D3	1YQ3DD_1NEN0D1
1FEPAA_1KMO0A1	1KMOAA_2FCP0A1	1T16AA_3BS00A1	1Z98AA_3C020A2
1FEPAA_1NQE0A1	1M56AA_1EHK0A1	1TQQAA_1WP10A2	1ZOYCC_1NEK0C1
1FEPAA_1XKW0A1	1M56DD_1NKZ0F2	1U19AA_2VT40A1	1ZOYCC_1NEN0C3
1FEPAA_2FCP0A1	1NEKCC_1NEK0D1	1U19AA_2Z730A1	1ZOYDD_1NEN0D1
1H6S11_2FGQ0X3	1NEKCC_1NEN0D1	1U77AA_3B9W0A1	2AHYAA_1K4C0C1
1H6S11_2OMF0A2	1NEKCC_1ZOY0C1	1VF5DD_1EZV0E2	2AXTaa_1EYS0L1
1HXXAA_1PRN0A1	1NKZAA_1LGH0A2	1XKWAA_1KMO0A1	2AXTaa_1EYS0M1
1HXXAA_2FGQ0X1	1NKZBB_1VF50F1	1XKWAA_1NQE0A1	2AXTaa_1OGV6L1

2AXTdd_1EYS0L1	2GUFAA_1XKW0A1	2R6GGG_3D310D1	3CSLAA_1XKW0A1
2AXTdd_1EYS0M1	2GUFAA_2FCP0A1	2RH1AA_1U190A1	3CSLAA_2FCP0A1
2AXTdd_1OGV6L1	2HDIAA_1KMO0A1	2RH1AA_2Z730A1	3CX5II_1BGY0J1
2B2FAA_3B9W0A1	2HDIAA_1XKW0A1	2RH1AA_3CAP0A1	3DDLAA_1C3W0A1
2B6PAA_3C020A2	2HDIAA_2FCP0A1	2VT4BB_1U190A1	3DDLAA_1E127A1
2E74GG_1VF50R1	2IAHAA_1KMO0A1	2VT4BB_2Z730A1	3DDLAA_1H2S0A1
2FBWCC_1NEK0C1	2IAHAA_1NQE0A1	2VT4BB_3CAP0A1	3DWOXX_3BS00A1
2FBWCC_1NEN0C3	2IAHAA_2FCP0A1	3B9WAA_2B2F0A2	3EMLAA_1U190A1
2GSKAA_1KMO0A1	2IXXAA_2FGQ0X1	3BS0AA_1T160A1	3EMLAA_2Z730A1
2GSKAA_1XKW0A1	2PORAA_2FGQ0X2	3C02AA_1Z980A3	3EMLAA_3CAP0A1
2GSKAA_2FCP0A1	2PORAA_2OMF0A1	3CSLAA_1KMO0A1	
2GUFAA_1KMO0A1	2R6GGG_3D310C1	3CSLAA_1NQE0A1	

7.4.1.4 Hardest test set

1A0SPP_1PRN0A1	1BE3JJ_1NKZ0E1	1IJDAA_1VF50F1	1LGHBB_1VF50E1
1A0SPP_2FGQ0X3	1BE3JJ_1QLE0D1	1IJDAA_1VF50R1	1LGHBB_1VF50F1
1A0SPP_2IWV0A1	1H2SAA_2VT40A1	1IJDAA_1VF50S1	1LGHBB_1VF50R1
1A0SPP_2J1N0A1	1H6S11_1QD60C1	1IJDBB_1NKZ0F1	1LGHBB_1VF50S1
1A0SPP_2J1N0B1	1H6S11_1QD60D1	1IJDBB_1QLE0D1	1M56CC_1YEW0C1
1A0SPP_2J1N0C1	1H6S11_2IWV0A1	1IJDBB_1VF50E1	1M56DD_1EHK0C1
1A0SPP_2O4V0A1	1H6S11_2J1N0A1	1IJDBB_1VF50F1	1M56DD_1LGH0B1
1A0SPP_2O4V0B1	1H6S11_2J1N0B1	1IJDBB_1VF50R1	1M56DD_1LGH0E4
1A0SPP_2O4V0C1	1H6S11_2J1N0C1	1IJDBB_1VF50S1	1M56DD_1NKZ0B2
1A0SPP_2OMF0A1	1HXXAA_1A0T0P1	1IJDBB_1VF50T1	1M56DD_1NKZ0D3
1A0SPP_2POR0A3	1HXXAA_1A0T0Q1	1IJDBB_1VF50U1	1M56DD_1VF50E1
1A0SPP_2QTK0A1	1HXXAA_1A0T0R1	1JB0JJ_1EHK0C1	1M56DD_1VF50F1
1AF6AA_1PRN0A1	1HXXAA_1QD50A1	1JB0JJ_1LGH0B1	1M56DD_1VF50R1
1AF6AA_2FGQ0X3	1HXXAA_1QD60C1	1JB0JJ_1OCC0M1	1M56DD_1VF50S1
1AF6AA_2J1N0A1	1HXXAA_1QD60D1	1JB0JJ_1OCC0Z1	1NEKCC_1ZOY0D1
1AF6AA_2J1N0B1	1HXXAA_2IWV0A1	1JB0JJ_1VF50E1	1NKZAA_1LGH0B1
1AF6AA_2J1N0C1	1HXXAA_2MPR0A1	1JB0JJ_1VF50F1	1NKZAA_1LGH0E2
1AF6AA_2O4V0A1	1HXXAA_2MPR0B1	1JB0JJ_1VF50R1	1NKZAA_1NKZ0B1
1AF6AA_2O4V0B1	1HXXAA_2MPR0C1	1JB0JJ_1VF50S1	1NKZAA_1NKZ0D3
1AF6AA_2O4V0C1	1HXXAA_2O4V0A1	1L0LKK_1LGH0B4	1NKZAA_1NKZ0F2
1AF6AA_2OMF0A1	1HXXAA_2O4V0B1	1L0LKK_1LGH0E4	1NKZAA_1OCC0X1
1AF6AA_2POR0A3	1HXXAA_2QTK0A1	1L0LKK_1NKZ0F2	1NKZAA_1VF50S1
1AF6AA_2QTK0A1	1IJDAA_1LGH0B1	1L0LKK_1OCC0X1	1NKZBB_1EHK0C1
1AQBAA_1P4T0A1	1IJDAA_1LGH0E4	1LGHBB_1EHK0C1	1NKZBB_1LGH0E4
1AQBAA_1THQ8A1	1IJDAA_1NKZ0B1	1LGHBB_1NKZ0F1	1NKZBB_1NKZ0A2
1BE3JJ_1LGH0B2	1IJDAA_1NKZ0D3	1LGHBB_1OCC0J1	1NKZBB_1QLE0D1
1BE3JJ_1LGH0E4	1IJDAA_1NKZ0F2	1LGHBB_1OCC0W1	1NKZBB_1VF50E1
1BE3JJ_1NKZ0C1	1IJDAA_1VF50E1	1LGHBB_1QLE0D1	1NKZBB_1VF50R1

1OCCBB_1EHK0B1
1OCCCC_1YEW0C1
1PY6AA_2VT40A1
1QLEDD_1EHK0C1
1QLEDD_1LGH0B1
1QLEDD_1LGH0E2
1QLEDD_1NKZ0B2
1QLEDD_1NKZ0F2
1QLEDD_1VF50E1
1QLEDD_1VF50F1
1QLEDD_1VF50R1
1QLEDD_1VF50S1
1T16AA_2FGQ0X1
1T16AA_2OMF0A1
1T16AA_2POR0A1
1U19AA_1E127A1
1U19AA_1VGO0A1
1U19AA_2EI40A1
1U2MBB_1YCE0A1
1U2MBB_1YCE0B1
1U2MBB_1YCE0C1
1U2MBB_1YCE0D1
1U2MBB_1YCE0E1
1U2MBB_1YCE0F1
1U2MBB_1YCE0H1
1U2MBB_1YCE0I1
1U2MCC_1K4C0C2
1XIOAA_2VT40A1
1YCEaa_1H2S0B1
1YCEaa_1K4C0C1
1YQ3DD_1NEK0D1
1ZOYCC_1NEN0D1
1ZOYCC_1ZOY0D1
1ZOYDD_1NEK0D1
2E74GG_1EHK0C1
2E74GG_1LGH0B1
2E74GG_1LGH0E4
2E74GG_1NKZ0B1
2E74GG_1NKZ0D3
2E74GG_1NKZ0F1
2E74GG_1VF50F1
2E74GG_1VF50S1
2E74GG_1VF50U1
2EI4AA_2VT40A1
2EI4AA_3CAP0A1
2FBWCC_1ZOY0D1
2GSKAA_1A0T0P1
2GSKAA_1A0T0Q1
2GSKAA_1A0T0R1
2GSKAA_2MPR0A1
2GSKAA_2MPR0B1
2GSKAA_2MPR0C1
2GUFAA_1A0T0P1
2GUFAA_1A0T0Q1
2GUFAA_1A0T0R1
2GUFAA_2MPR0A1
2GUFAA_2MPR0B1
2GUFAA_2MPR0C1
2GUFAA_2QTK0A1
2HDIAA_1A0T0P1
2HDIAA_1A0T0Q1
2HDIAA_1A0T0R1
2HDIAA_2MPR0A1
2HDIAA_2MPR0B1
2HDIAA_2MPR0C1
2IWVAA_1PRN0A1
2IWVAA_1QD50A1
2IWVAA_1QD60C1
2IWVAA_1QD60D1
2IWVAA_1UYN0X1
2IWVAA_2OMF0A3
2IWVAA_2QOM0B1
2IXXAA_1A0T0P1
2IXXAA_1A0T0Q1
2IXXAA_1A0T0R1
2IXXAA_1PRN0A1
2IXXAA_1QD60C1
2IXXAA_1QD60D1
2IXXAA_2IWV0A1
2IXXAA_2MPR0A1
2IXXAA_2MPR0B1
2IXXAA_2MPR0C1
2IXXAA_2O4V0A1
2IXXAA_2O4V0B1
2IXXAA_2O4V0C1
2IXXAA_2POR0A1
2IXXAA_2QTK0A1
2JAFAA_2VT40A1
2ODJBB_1A0T0P1
2ODJBB_1A0T0R1
2ODJBB_1PRN0A1
2ODJBB_2FGQ0X2
2ODJBB_2IWV0A1
2ODJBB_2J1N0A1
2ODJBB_2J1N0B1
2ODJBB_2J1N0C1
2ODJBB_2MPR0B1
2ODJBB_2MPR0C1
2ODJBB_2O4V0A1
2ODJBB_2O4V0B1
2ODJBB_2O4V0C1
2ODJBB_2OMF0A3
2ODJBB_2POR0A1
2PORAA_2IWV0A1
2PORAA_2J1N0A1
2PORAA_2J1N0B1
2PORAA_2J1N0C1
2PORAA_2O4V0B1
2PORAA_2O4V0C1
2QTKAA_1A0T0P1
2QTKAA_1A0T0R1
2QTKAA_1PRN0A1
2QTKAA_2FGQ0X2
2QTKAA_2IWV0A1
2QTKAA_2J1N0A1
2QTKAA_2J1N0B1
2QTKAA_2J1N0C1
2QTKAA_2MPR0A1
2QTKAA_2MPR0B1
2QTKAA_2MPR0C1
2QTKAA_2O4V0A1
2QTKAA_2O4V0C1
2QTKAA_2OMF0A2
2QTKAA_2POR0A1
2UUHAA_1K4C0C2
2UUHAA_2BL20A1
2UUHAA_2BL20B1
2UUHAA_2BL20C1
2UUHAA_2BL20D1
2UUHAA_2BL20E1
2UUHAA_2BL20F1
2UUHAA_2BL20G1
2UUHAA_2BL20H1
2UUHAA_2BL20I1
2UUHAA_2BL20J1
2VT4BB_1C3W1A1
2VT4BB_1E124A1
2VT4BB_1H2S0A1
2VT4BB_1VF50N1
2VT4BB_1VGO0A1
2VT4BB_2EI40A1
3BS0AA_1I780A1
3BS0AA_1UYN0X1
3BS0AA_2IWV0A1
3CP1AA_1H2S0B1
3CX5II_1LGH0B1
3CX5II_1LGH0E4
3CX5II_1NKZ0B3
3CX5II_1NKZ0C2
3CX5II_1NKZ0E1
3CX5II_1NKZ0F2
3DDLAA_1U190A1
3DDLAA_2VT40A1
3DDLAA_2Z730A1
3DDLAA_3CAP0A1
3DWOXX_1I780A1
3DWOXX_2FGQ0X1
3DWOXX_2OMF0A1
3DWOXX_2POR0A1

7.4.2 Main chain bumps

The number of main chain bumps for each model was computed using the program WHATCHECK (Hooft et al., 1996). See Figures 7.2 and 7.3.

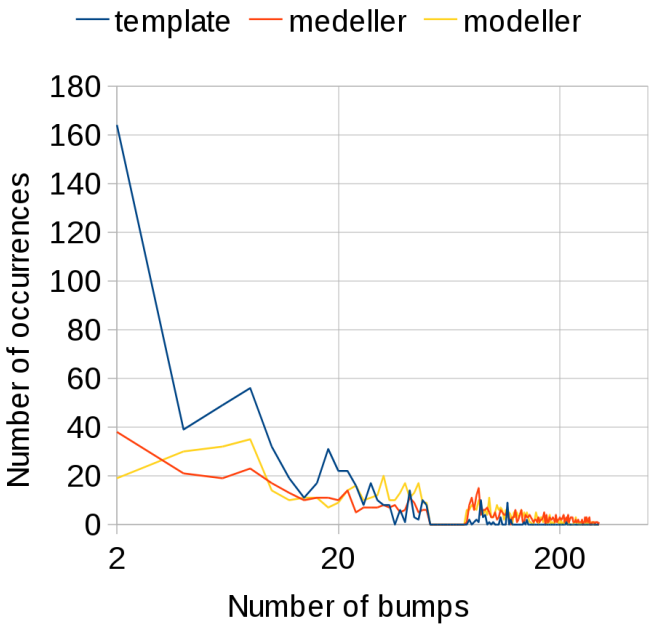


Figure 7.2: Distribution of main chain bumps for the template as well as the MEDELLER and Modeller models - The X axis is shown in logarithmic scale.

7.4.3 Transmembrane geometry

7.4.3.1 Transmembrane "shift"

Transmembrane shift was assessed by annotating the models with iMembrane, using the native target structure as the template. A window around the TM region was defined by selecting consecutive alignment columns marked as being in the membrane "tail" layer in the native structure. The window was further enlarged, until all adjacent residues were included, where the model's iMembrane annotation placed them in the membrane "tail" region. The "shift" is defined as the number of columns in the window where the annotation for native and model differs. See Figures 7.4 and 7.5.

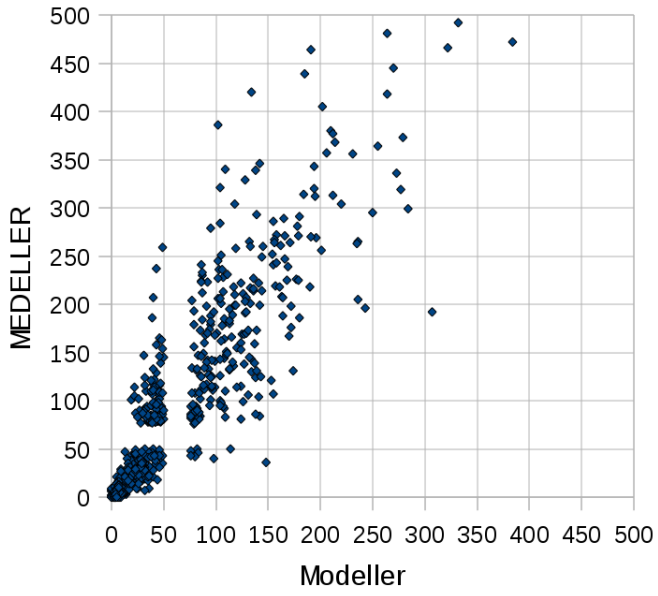


Figure 7.3: Number of main chain bumps for Modeller versus MEDELLER complete models. - One datapoint was cut off when scaling the figure, for increased clarity.

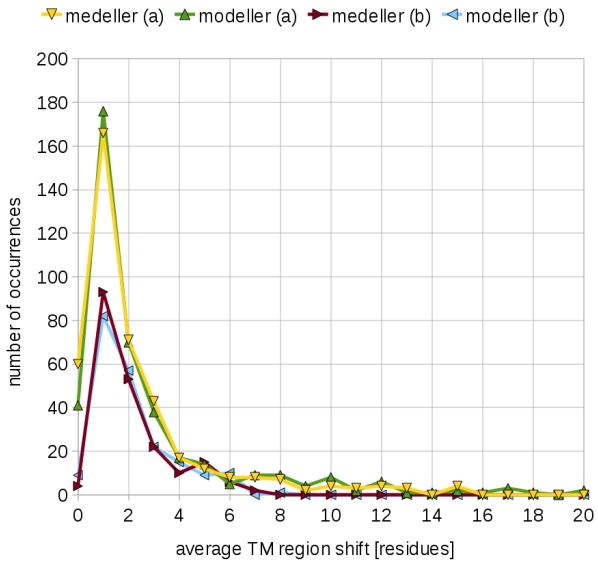


Figure 7.4: Distribution of transmembrane shift (as number of residues) for Modeller and MEDELLER. - The dataset has been divided into alpha-helical (a) and beta-barrel (b) proteins

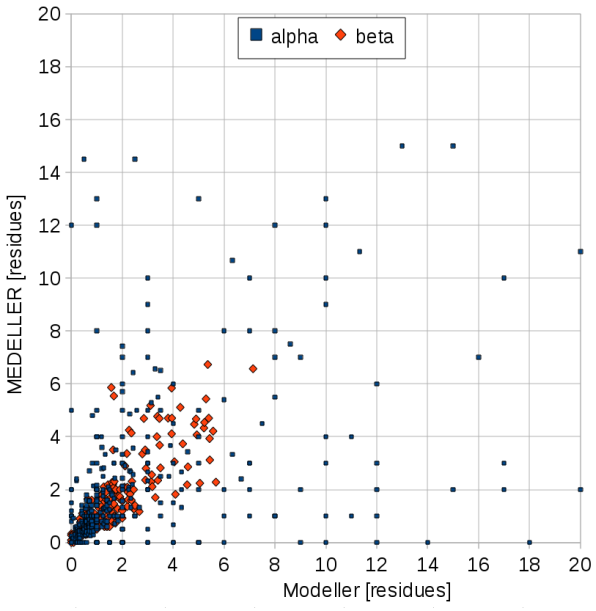


Figure 7.5: Comparison of transmembrane shift (as number of residues) between Modeller and MEDELLER. - The dataset has been divided into alpha-helical (a) and beta-barrel (b) proteins. Two datapoints (with Modeller shift ¿ 20 residues) were cut off during scaling of this figure, for better visualisation.

7.4.3.2 Transmembrane "relative tilt angle"

The (relative) transmembrane tilt angle was calculated by aligning the native and model structures and calculating the angle between the helix (or beta sheet) axis and the membrane normal, in both the model and the native structure. The difference of the two values is defined as the relative tilt angle. See Figures 7.6 and 7.7.

7.4.3.3 Transmembrane "relative rotation angle"

The (relative) transmembrane rotation angle was calculated by first aligning the native and model structures. The helix (or beta sheet) axis was calculated. Then the C-alpha atom of the middle residue of the TM region was defined as point P. Let vP be the vector that is (a) perpendicular to the helix axis and (b) goes through point P. We superimposed the helix axes, effectively setting the tilt angle to 0 degrees. The relative rotation angle is now defined as the angle between the vectors vP of the native and model structure. See Figures 7.8 and 7.9.

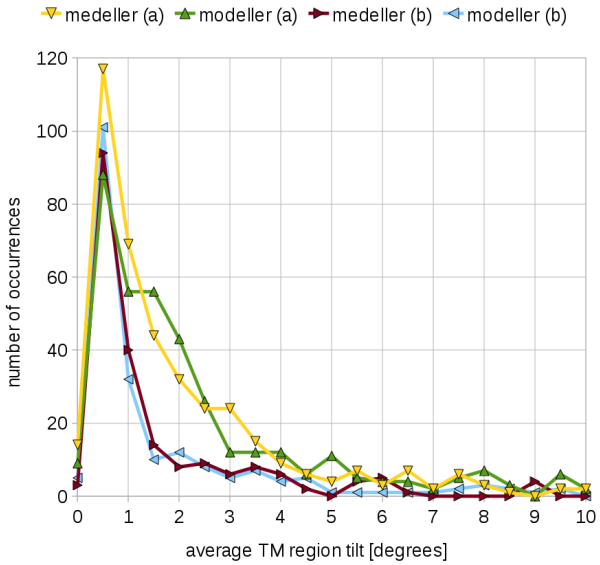


Figure 7.6: Distribution of transmembrane tilt angles for Modeller and MEDELLER. - The dataset has been divided into alpha-helical (a) and beta-barrel (b) proteins.

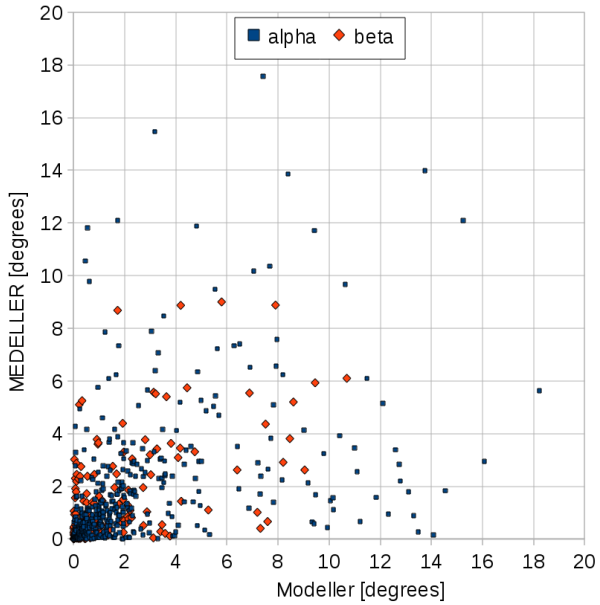


Figure 7.7: Comparison of transmembrane tilt angles between Modeller and MEDELLER. - The dataset has been divided into alpha-helical (a) and beta-barrel (b) proteins.

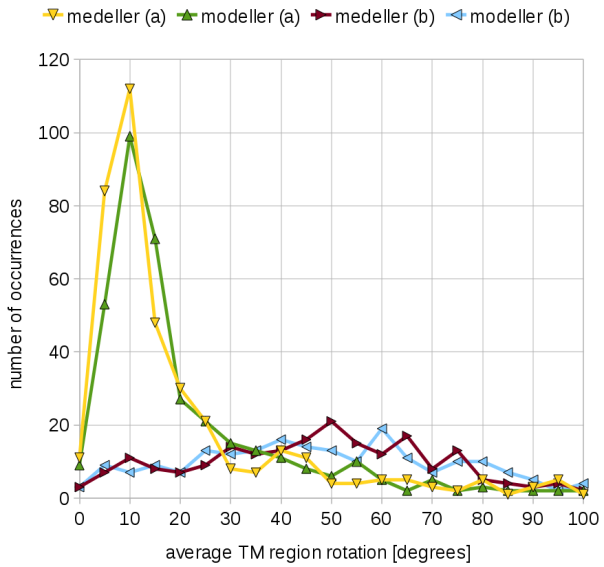


Figure 7.8: Distribution of transmembrane rotation angles for Modeller and MEDELLER. - The dataset has been divided into alpha-helical (a) and beta-barrel (b) proteins.

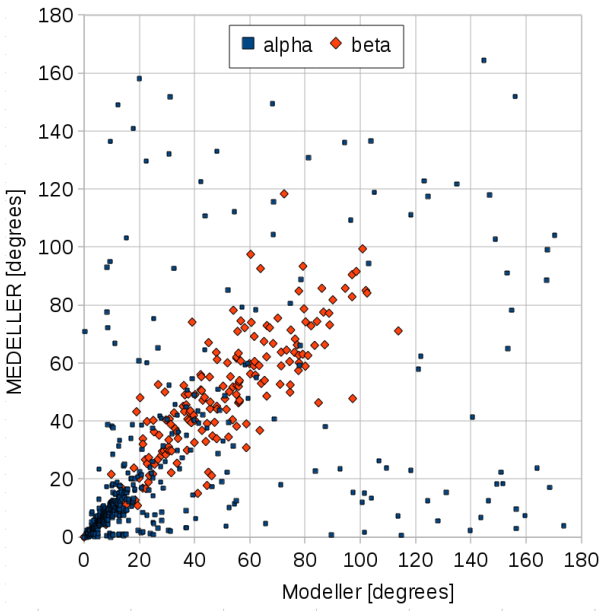


Figure 7.9: Comparison of transmembrane tilt angles between Modeller and MEDELLER. - The dataset has been divided into alpha-helical (a) and beta-barrel (b) proteins.

7.4.4 Comparison when modelling using multiple templates

See Figures 7.10, 7.11 and 7.12.

	Core								HiAcc							
			MEDELLER		Modeller		Difference				MEDELLER		Modeller		Difference	
	Cov	CC	bckb	TSc	bckb	TSc	bckb	TSc	Cov	CC	bckb	TSc	bckb	TSc	bckb	TSc
	%	%	rmsd	gdt	rmsd	gdt	rmsd	gdt	%	%	rmsd	gdt	rmsd	gdt	rmsd	gdt
all	92.52	96.77	3.24	0.75	3.32	0.72	-0.07	0.03	93.02	97.28	3.24	0.76	3.33	0.73	-0.09	0.03

	HiCov								Complete							
			MEDELLER		Modeller		Difference				MEDELLER		Modeller		Difference	
	Cov	CC	bckb	TSc	bckb	TSc	bckb	TSc	Cov	CC	bckb	TSc	bckb	TSc	bckb	TSc
	%	%	rmsd	gdt	rmsd	gdt	rmsd	gdt	%	%	rmsd	gdt	rmsd	gdt	rmsd	gdt
all	95.56	100.00	3.57	0.76	3.48	0.74	0.09	0.02	100.00	100.00	5.03	0.76	4.65	0.75	0.39	0.01

Figure 7.10: Modelling accuracy when modelling from more than one template - Comparison of modelling accuracy for MEDELLER and Modeller across test sets and model types. Modeller's top decoy was selected using Modeller's DOPE score. All models were based on a multiple structure alignment, between the target and all templates, using Multiprot. Abbreviations: Cov, fraction of the target covered by a particular model; CC, "core coverage", i.e. the fraction of the target covered by the model when ignoring terminal gaps and gaps longer than 26 residues (which cannot be closed with FREAD loop modelling); bckb, backbone RMSD; TSc, GDT_TS normalised by CC.

7.4.5 Example model

Figures 7.13 and 7.14 relate to the example model of human adenosine A2A receptor (PDB code "3EML", chain A) mentioned in Section 4.4.7.

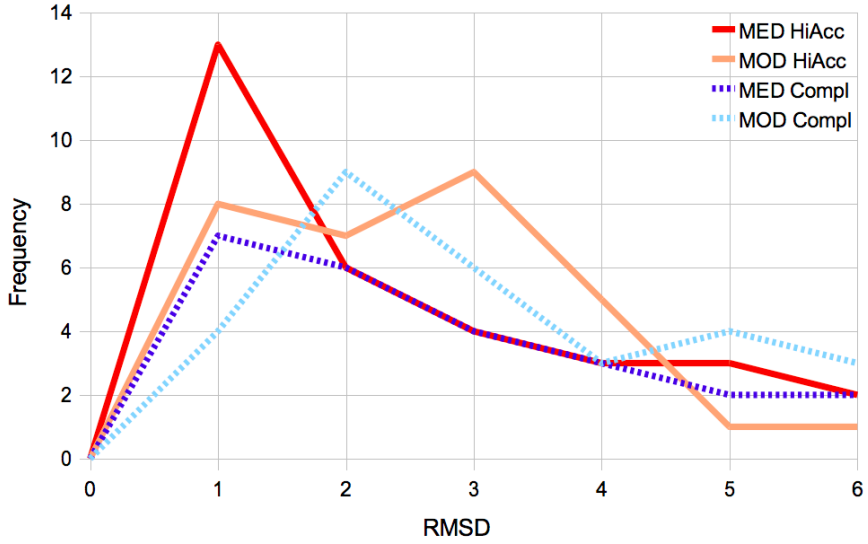


Figure 7.11: Distribution of modelling accuracy, when modelling from multiple templates - Backbone RMSD to the native structure achieved by MEDELLER's high-accuracy (MED HiAcc) and complete (MED Compl) models and the corresponding coordinates in the Modeller model (MOD HiAcc, MOD Compl). MEDELLER's accuracy distribution peaks in the 0-1Å range for both the high-accuracy models and the complete models. In comparison, Modeller's equivalent co-ordinates have a 0-1Å / 2-3Å dual peak (high-accuracy models) and a 1-2Å peak (complete models). Both methods' accuracy distributions have long tails, which are not shown in this figure.

Target	Multi-template						Single-template (structure alignment)							
	#tem	best %id	worst %id	avg %id			Best %id		Average		Best		Worst	
					MED	MOD	MED	MOD	MED	MOD	MED	MOD	MED	MOD
1YEWA	2	100.00	100.00	100.00	**0.11**	1.21	**0.11**	1.48	**0.11**	2.14	**0.11**	1.48	**0.11**	2.80
1YEWB	2	100.00	100.00	100.00	**0.09**	2.19	**0.08**	1.77	**0.08**	1.47	**0.08**	1.18	**0.09**	1.77
1BE3C	3	100.00	46.65	82.22	**0.32**	0.53	**0.32**	0.53	**0.51**	0.63	**0.30**	0.45	0.92	0.91
2EVUA	10	100.00	25.19	35.26	**0.27**	0.44	**0.27**	0.38	1.22	1.23	**0.27**	0.38	1.50	1.49
3D9SA	9	99.19	21.19	54.99	0.60	0.59	0.50	0.54	1.13	1.13	0.41	0.43	1.77	1.79
2JAFA	6	98.35	8.71	36.85	0.56	0.56	0.56	0.58	1.82	1.85	0.56	0.58	**4.86**	5.09
2VT4B	10	98.19	3.64	17.83	**1.93**	2.02	**1.24**	1.35	**3.06**	3.23	**1.24**	1.35	**4.01**	4.34
2EI4A	6	97.90	4.91	36.66	0.75	**0.68**	0.75	0.72	1.90	**1.82**	0.75	0.72	4.22	**3.93**
1YEWC	2	95.77	82.90	89.34	**0.07**	3.21	**0.49**	3.36	**0.70**	2.23	**0.49**	1.10	**0.91**	3.36
1PY6A	6	93.86	5.39	44.36	0.73	**0.62**	**0.63**	0.71	1.41	1.41	**0.63**	0.71	3.93	3.88
2E74C	2	92.44	89.73	91.09	**1.35**	2.39	**1.36**	3.02	**1.36**	2.66	**1.36**	2.31	**1.37**	3.02
2B6PA	10	89.27	19.93	41.19	**0.26**	1.13	**0.26**	0.59	1.19	1.22	**0.26**	0.59	1.61	1.59
1U19A	6	85.27	5.75	22.65	2.80	**2.72**	1.68	1.68	2.87	2.82	1.68	1.68	3.86	**3.77**
3CX5D	3	83.07	49.61	61.29	**0.27**	2.88	**0.17**	0.57	**0.93**	1.90	**0.17**	0.57	**1.32**	2.62
1BE3G	3	82.56	13.13	59.35	**0.41**	7.68	**0.14**	0.91	**0.50**	3.01	**0.14**	0.91	**0.93**	6.57
2E74A	2	82.03	81.02	81.53	**2.58**	2.85	0.50	0.45	0.51	0.48	0.50	0.45	0.53	0.52
2BHWA	3	81.06	67.52	73.42	**0.35**	0.64	**1.02**	1.10	**1.04**	1.38	**0.40**	1.10	**1.69**	1.89
1VF5D	2	77.40	15.14	46.27	**2.96**	3.71	**0.32**	2.37	**1.29**	2.84	**0.32**	2.37	**2.27**	3.30
2E74B	2	62.73	57.14	59.94	4.16	**3.31**	**0.82**	5.68	**0.86**	5.08	**0.82**	4.48	**0.89**	5.68
1OGVM	3	57.01	24.01	35.47	**1.04**	1.22	0.68	0.67	**1.06**	1.37	0.68	0.67	**1.29**	2.12
1EZVC	2	49.87	48.84	49.36	1.15	**0.90**	0.94	0.93	0.94	0.94	0.94	0.93	0.95	0.96
1J4NA	9	44.27	19.93	35.66	**1.57**	1.70	1.07	1.07	1.62	1.59	1.07	1.07	2.07	2.07
1Z98A	9	35.77	17.78	27.52	5.36	**5.00**	1.95	**1.83**	1.53	1.48	**1.18**	1.24	1.95	**1.83**
1U77A	2	33.50	16.34	24.92	3.16	**2.48**	1.69	1.67	1.89	1.88	1.69	1.67	2.10	2.09
1FX8A	9	32.56	21.91	24.79	5.64	**2.57**	**1.38**	1.47	1.61	1.66	1.38	1.42	1.84	1.87
2O9DB	10	31.91	22.00	28.87	4.82	**1.60**	1.63	**1.57**	1.61	1.59	1.26	**1.20**	1.81	1.81
3C02A	9	31.66	17.91	22.51	3.89	**1.82**	**1.27**	1.40	1.61	1.62	**1.27**	1.33	**2.09**	2.14
1XIOA	6	30.25	6.62	23.46	2.78	**1.87**	**1.53**	1.60	2.03	2.05	**1.27**	1.36	4.81	**4.40**
1H2SA	5	29.11	6.14	23.06	1.48	**1.33**	1.08	1.09	1.75	1.77	0.92	0.94	**4.05**	4.13
3DDLA	9	20.74	4.64	12.61	4.92	**3.95**	2.32	2.33	2.74	**2.67**	1.90	**1.81**	**4.11**	3.77
1EZVE	2	16.51	12.79	14.65	6.08	**5.98**	**1.41**	1.87	1.11	1.13	0.81	**0.39**	**1.41**	1.87
2R6GG	2	14.33	14.33	14.33	**6.78**	7.57	**2.65**	3.11	**2.59**	2.92	**2.54**	2.73	**2.65**	3.11
1L0LK	4	6.67	3.12	4.45	3.44	**3.30**	1.85	1.85	**1.62**	2.27	**1.10**	1.85	**1.97**	3.26
1U2MB	8	4.48	2.22	3.33	23.57	**16.35**	**2.42**	3.81	**2.43**	3.04	2.21	**1.98**	**3.04**	4.04
1AQBA	2	4.11	3.60	3.86	**17.23**	19.06	**2.93**	3.08	2.85	**2.77**	2.77	**2.46**	**2.93**	3.08
Average					**3.24**	3.32	**1.09**	1.63	**1.47**	1.98	**0.96**	1.31	**2.17**	2.88

Figure 7.12: Comparing multiple and single template modelling accuracy for MEDELLER and Modeller - All models are "core" models. All accuracy values are in Å backbone RMSD. The method with the better accuracy, i.e. lower RMSD (by a margin of more than 0.05Å), is formatted in bold green font for MEDELLER and bold red font for Modeller.

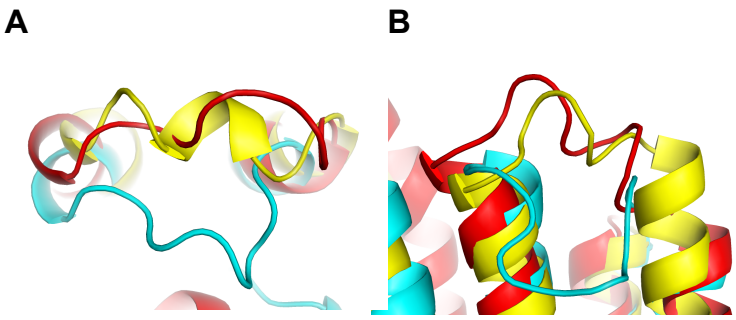


Figure 7.13: Predicted loop structures - MEDELLER (yellow) and Modeller (cyan) models aligned to the native x-ray structure (red).

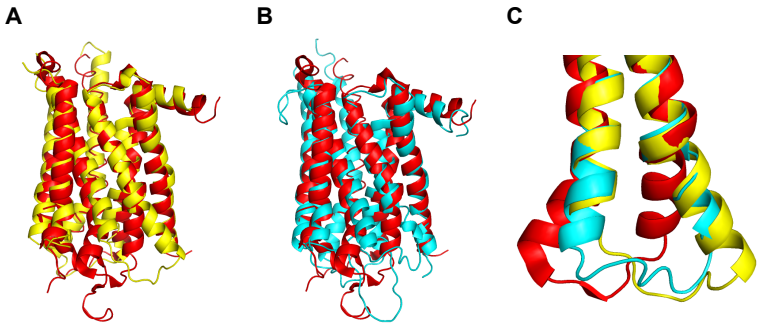


Figure 7.14: Failure to model a helix kink - MEDELLER (yellow) and Modeller (cyan) models aligned to the native x-ray structure (red).

References

Alber, F., Dokudovskaya, S., Veenhoff, L., Zhang, W., Kipper, J., Devos, D., Suprapto, A., Karni-Schmidt, O., Williams, R., Chait, B., Rout, M., and Sali, A. (2007). Determining the architectures of macromolecular assemblies. *Nature*, 450(7170):683–694. 170

Alberts, B., Johnson, A., Lewis, J., Raff, M., Robets, K., and Walter, P. (2002). *Molecular Biology of the Cell*. Garland Science, a member of the Taylor & Francis Group, 4th edition. 1, 2, 10

Altschul, S., Gish, W., Miller, W., Myers, E., and Lipman, D. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410. 16, 19

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402. 16, 19, 71

Amini, A., Shrimpton, P. J., Muggleton, S. H., and Sternberg, M. J. (2007). A general approach for developing system-specific functions to score protein-ligand docked complexes using support vector inductive logic programming. *Proteins*, 69(4):823–831. 12

Anfinsen, C. (1973). Principles that govern the folding of protein chains. *Science*, 181(96):223–230. 2, 15

Argos, P., Rao, M. J. K., and Hargrave, P. A. (1982). Structural Prediction of Membrane-Bound Proteins. *European Journal of Biochemistry*, 128(2-3):565–575. 35

Aszódi, A. and Taylor, W. R. (1994). Secondary structure formation in model polypeptide chains. *Protein engineering*, 7(5):633–644. 31

Azzazy, H. M. and Highsmith, W. E. (2002). Phage display technology: clinical applications and recent innovations. *Clinical biochemistry*, 35(6):425–445. 11

Barth, P., Wallner, B., and Baker, D. (2009). Prediction of membrane protein structures with complex topologies using limited constraints. *Proceedings of the National Academy of Sciences*, 106(5):1409–1414. 109

Bateman, A., Coin, L., Durbin, R., Finn, R., Hollich, V., Griffiths-Jones, S., Khanna, A., Marshall, M., Moxon, S., Sonnhammer, E., Studholme, D., Yeats, C., and Eddy, S. (2004). The pfam protein families database. *Nucleic acids research*, 32(Suppl 1):D138–D141. 11

Bates, P. A., Kelley, L. A., MacCallum, R. M., and Sternberg, M. J. (2001). Enhancement of protein modeling by human intervention in applying the automatic programs 3D-JIGSAW and 3D-PSSM. *Proteins*, Suppl 5:39–46. 17, 29, 108

Bendtsen, J. D. D., Nielsen, H., von Heijne, G., and Brunak, S. (2004). Improved prediction of signal peptides: SignalP 3.0. *Journal of molecular biology*, 340(4):783–795. 36

Benkert, P., Tosatto, S. C. E. C., and Schomburg, D. (2007). QMEAN: A comprehensive scoring function for model quality assessment. *Proteins*, 71(1):261–277. 18

Berg, J. M., Tymoczko, J. L., and Stryer, L. (2002). *Biochemistry*. W. H. Freeman, fifth edition. 4, 67

REFERENCES

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242. 12, 40, 71, 72

Boomsma, W. and Hamelryck, T. (2005). Full cyclic coordinate descent: solving the protein loop closure problem in Calpha space. *BMC Bioinformatics*, 6(1):159. 146

Bourne, P. and Weissig, H. (2003). *Structural Bioinformatics (Methods of Biochemical Analysis, V. 44)*. Wiley-Liss. 22

Bowie, J. (1997). Helix packing in membrane proteins. *Journal of molecular biology*, 272(5):780–789. 67, 90

Bowie, J. U. (2005). Solving the membrane protein folding problem. *Nature*, 438(7068):581–589. 5, 7

Bradley, P., Malmström, L., Qian, B., Schonbrun, J., Chivian, D., Kim, D. E., Meiler, J., Misura, K. M. S., and Baker, D. (2005). Free modeling with rosetta in casp6. *Proteins*, 61(Suppl 7):128–134. 15

Bugalho, M. and Oliveira, A. (2009). Constant time clash detection in protein folding. *Journal of bioinformatics and computational biology*, 7(1):55–74. 146

Burke, D. F., Deane, C. M., Nagarajaram, H. A., Campillo, N., Martin-Martinez, M., Mendes, J., Molina, F., Perry, J., Reddy, B. V., Soares, C. M., Steward, R. E., Williams, M., Carrondo, M. A., Blundell, T. L., and Mizuguchi, K. (1999). An iterative structure-assisted approach to sequence alignment and comparative modeling. *Proteins*, 37(Suppl 3):55–60. 18

Carter, P. (2006). Potent antibody therapeutics by design. *Nature Reviews Immunology*, 6(5):343–357. 11

Chen, H. and Zhou, H.-X. (2005). Prediction of solvent accessibility and sites of deleterious mutations from protein sequence. *Nucleic Acids Research*, 33(10):3193–3199. 25

Chenna, R., Sugawara, H., Koike, T., Lopez, R., Gibson, T., Higgins, D., and Thompson, J. (2003). Multiple sequence alignment with the clustal series of programs. *Nucleic Acids Research*, 31(13):3497–3500. 20

Choi, Y. and Deane, C. M. (2010). FREAD revisited: Accurate loop structure prediction using a database search algorithm. *Proteins*, 78(6):1431–1440. 111, 113, 121, 136, 140, 142, 146, 160, 162

Clore, G. M., Brünger, A. T., Karplus, M., and Gronenborn, A. M. (1986). Application of molecular dynamics with interproton distance restraints to three-dimensional protein structure determination. A model study of crambin. *Journal of molecular biology*, 191(3):523–551. 32

Cohen, F. E. and Kuntz, I. D. (1989). *Tertiary Structure Prediction*. Prediction of Protein Structure and the Principles of Protein Conformation. Plenum. 31

Crick, F. (1953). The packing of α-helices: simple coiled-coils. *Acta Crystallographica*, 6(8-9):689–697. 68

Cuthbertson, J. M., Doyle, D. A., and Sansom, M. S. (2005). Transmembrane helix prediction: a comparative evaluation and analysis. *Protein Engineering, Design and Selection*, 18(6):295–308. 36

Das, R. and Baker, D. (2008). Macromolecular modeling with rosetta. *Annual review of biochemistry*, 77(1):363–382. 14

Das, R., Qian, B., Raman, S., Vernon, R., Thompson, J., Bradley, P., Khare, S., Tyka, M. D., Bhat, D., Chivian, D., Kim, D. E., Sheffler, W. H., Malmström, L., Wollacott, A. M., Wang, C., Andre, I., and Baker, D. (2007). Structure prediction for casp7 targets using extensive all-atom refinement with rosetta@home. *Proteins*, 69(S8):118–128. 15

Dayhoff, M. O. and Schwartz, R. M. (1978). *A model of evolutionary change in proteins*, volume 5 of *Atlas of Protein Sequence and Structure*. Washington, DC National Biomedical Research Foundation.

8

Deane, C. M. (2000). *Protein structure prediction: amino acid propensities and comparative modelling*. PhD thesis, Dept. of Biochemistry, University of Cambridge. 30

Deane, C. M. and Blundell, T. L. (2001). CODA: a combined algorithm for predicting the structurally variable regions of protein models. *Protein Sci*, 10(3):599–612. 17, 108, 161

Deane, C. M., Dong, M., Huard, F. P., Lance, B. K., and Wood, G. R. (2007). Cotranslational protein folding–fact or fiction? *Bioinformatics*, 23(13):i142–i148. 14

Deane, C. M., Kaas, Q., and Blundell, T. L. (2001). SCORE: predicting the core of protein models. *Bioinformatics*, 17(6):541–550. 17

DeGrado, W. F., Gratkowski, H., and Lear, J. D. (2003). How do helix-helix interactions help determine the folds of membrane proteins? Perspectives from the study of homo-oligomeric helical bundles. *Protein Science*, 12(4):647–665. 5

Deisenhofer, J., Epp, O., Miki, K., Huber, R., and Michel, H. (1985). Structure of the protein subunits in the photosynthetic reaction centre of rhodopseudomonas viridis at 3[angst] resolution. *Nature*, 318(6047):618–624. 65

Dunn, O. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64. 78

Eddy, S. (1998). Profile hidden markov models. *Bioinformatics*, 14(9):755–763. 16

Edgar, R. (2004a). MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5(1):113. 19, 20, 72, 129

Edgar, R. C. (2004b). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797. 20

Eilers, M., Patel, A., Liu, W., and Smith, S. (2002). Comparison of helix interactions in membrane and soluble alpha-bundle proteins. *Biophysical journal.*, 82(5):2720–2736. 68

Eilers, M., Shekar, S., Shieh, T., Smith, S., and Fleming, P. (2000). Internal packing of helical membrane proteins. *Proceedings of the National Academy of Sciences of the United States of America*, 97(11):5796–5801. 68, 101

Ellis, J. J., Huard, F. P. P., Deane, C. M., Srivastava, S., and Wood, G. R. (2010). Directionality in protein fold prediction. *BMC bioinformatics*, 11(1):172. 14

Elofsson, A. and von Heijne, G. (2007). Membrane Protein Structure: Prediction versus Reality. *Annual Review of Biochemistry*, 76(1):125–140. 5, 34, 35, 108

Eswar, N., Webb, B., Marti-Renom, M. A., Madhusudhan, M. S., Eramian, D., Shen, M.-Y. Y., Pieper, U., and Sali, A. (2007). Comparative protein structure modeling using MODELLER. *Current protocols in protein science*, 50:2.9.1–2.9.31. 16, 122

Eyre, T. A., Partridge, L., and Thornton, J. M. (2004). Computational analysis of alpha-helical membrane protein structure: implications for the prediction of 3D structural models. *Protein Engineering, Design and Selection*, 17(8):613–624. vii, 35, 66, 67, 68, 69, 89, 90, 93, 95, 97, 99, 108

Fenosa, A., Fuste, E., Ruiz, L., Veiga-Crespo, P., Vinuesa, T., Guallar, V., Villa, T. G., and Vinas, M. (2009). Role of TolC in Klebsiella oxytoca resistance to antibiotics. *J. Antimicrob. Chemother.*, 63(4):668–674. 110

Forrest, L., Tang, C., and Honig, B. (2006). On the Accuracy of Homology Modeling and Sequence Alignment Methods Applied to Membrane Proteins. *Biophysical Journal*, 91(2):508–517. 110, 114, 124, 140

Fu, R. and Cross, T. (1999). Solid-state nuclear magnetic resonance investigation of protein and polypeptide structure. *Annual review of biophysics and biomolecular structure*, 28(1):235–268. 13

REFERENCES

Ginalski, K., Elofsson, A., Fischer, D., and Rychlewski, L. (2003). 3d-jury: a simple approach to improve protein structure predictions. *Bioinformatics*, 19(8):1015–1018. 25

Gong, S., Worth, C. L., Bickerton, G. R., Lee, S., Tanramluk, D., and Blundell, T. L. (2009). Structural and functional restraints in the evolution of protein families and superfamilies. *Biochemical Society transactions*, 37:727–733. 92

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *Journal of molecular biology*, 162(3):705–708. 18, 29, 84

Gromiha, M. M. and Suwa, M. (2005). A simple statistical method for discriminating outer membrane proteins with better accuracy. *Bioinformatics (Oxford, England)*, 21(7):961–968. 108

Heckmann, D., Meyer, A., Laufer, B., Zahn, G., Stragies, R., and Kessler, H. (2008). Rational design of highly active and selective ligands for the alpha5beta1 integrin receptor. *Chembiochem : a European journal of chemical biology*, 9(9):1397–1407. 11

Henderson, R., Baldwin, J., Ceska, T., Zemlin, F., Beckmann, E., and Downing, K. (1990). Model for the structure of bacteriorhodopsin based on high-resolution electron cryo-microscopy. *Journal of Molecular Biology*, 213(4):899 – 929. 65

Henderson, R. and Unwin, P. N. T. (1975). Three-dimensional model of purple membrane obtained by electron microscopy. *Nature*, 257:28–32. 65

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919. 8, 73, 82

Hertzberg, R. P. and Pope, A. J. (2000). High-throughput screening: new technology for the 21st century. *Curr Opin Chem Biol*, 4(4):445–451. 11

Hill, J. R., Kelm, S., Shi, J., and Deane, C. M. (2011). Environment specific substitution tables improve membrane protein alignment. *Bioinformatics*, 27(13):i15–i23. viii, 62, 80, 81, 83, 84, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 102, 103, 104, 166, 167

Hirosawa, M., Totoki, Y., Hoshida, M., and Ishikawa, M. (1995). Comprehensive study on iterative algorithms of multiple sequence alignment. *Comput. Appl. Biosci.*, 11(1):13–18. 175

Holm, L. and Sander, C. (1996). Mapping the protein universe. *Science*, 273(5275):595–602. 17, 22

Hooft, R. W., Vriend, G., Sander, C., and Abola, E. E. (1996). Errors in protein structures. *Nature*, 381(6580):272. 131, 181

Javadpour, M., Eilers, M., Groesbeek, M., and Smith, S. (1999). Helix packing in polytopic membrane proteins: role of glycine in transmembrane helix association. *Biophysical journal*, 77(3):1609–1618. 68, 101

Jefferys, B. R., Kelley, L. A., and Sternberg, M. J. E. (2010). Protein Folding Requires Crowd Control in a Simulated Cell. *Journal of Molecular Biology*, 397(5):1329–1338. 14

Jones, D. T., Taylor, W. R., and Thornton, J. M. (1994). A model recognition approach to the prediction of all-helical membrane protein structure and topology. *Biochemistry*, 33(10):3038–3049. 10, 35

Just, W. (2001). Computational complexity of multiple sequence alignment with SP-score. *Journal of computational biology : a journal of computational molecular cell biology*, 8(6):615–623. 19

Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637. 27

Karplus, K., Barrett, C., and Hughey, R. (1998). Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856. 25

Kauko, A., Hedin, L. E., Thebaud, E., Cristobal, S., Elofsson, A., and von Heijne, G. (2010). Reposi-

tioning of Transmembrane -Helices during Membrane Protein Folding. *Journal of Molecular Biology*, 397(1):190–201. 5

Kelm, S., Shi, J., and Deane, C. M. (2009). iMembrane: homology-based membrane-insertion of proteins. *Bioinformatics*, 25(8):1086–1088. 56, 111, 113, 114, 165

Kelm, S., Shi, J., and Deane, C. M. (2010). MEDELLER: homology-based coordinate generation for membrane proteins. *Bioinformatics*, 26(22):2833–2840. 56, 108, 169

Khorana, H. G., Gerber, G. E., Herlihy, W. C., Gray, C. P., Anderegg, R. J., Nihei, K., and Biemann, K. (1979). Amino acid sequence of bacteriorhodopsin. *Proceedings of the National Academy of Sciences*, 76(10):5046–5050. 65

Koehl, P. and Delarue, M. (1995). A self consistent mean field approach to simultaneous gap closure and side-chain positioning in homology modelling. *Nature structural biology*, 2(2):163–170. 17, 108

Krivov, G. G., Shapovalov, M. V., and Dunbrack, R. L. (2009). Improved prediction of protein side-chain conformations with SCWRL4. *Proteins: Structure, Function, and Bioinformatics*, 77(4):778–795. 17

Krogh, A., Larsson, B., von Heijne, G., and Sonnhammer, E. L. (2001). Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *Journal of molecular biology*, 305(3):567–580. 35

Langosch, D. and Heringa, J. (1998). Interaction of transmembrane helices by a knobs-into-holes packing characteristic of soluble coiled coils. *Proteins.*, 31(2):150–159. 68

Lee, B. and Richards, F. M. (1971). The interpretation of protein structures: estimation of static accessibility. *Journal of molecular biology*, 55(3):379–400. 27

Levitt, M. (1992). Accurate modeling of protein conformation by automatic segment matching. *Journal of molecular biology*, 226(2):507–533. 17, 108

Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics (Oxford, England)*, 22(13):1658–1659. 71, 81, 123

Lindahl, E., Hess, B., and van der Spoel, D. (2001). GROMACS 3.0: a package for molecular simulation and trajectory analysis. *Journal of Molecular Modeling*, 7(8):306–317. 38

Lomize, M. A., Lomize, A. L., Pogozheva, I. D., and Mosberg, H. I. (2006). OPM: orientations of proteins in membranes database. *Bioinformatics*, 22(5):623–625. 36, 81, 122

MacKerell, A., Bashford, D., Bellott, Dunbrack, R., Evanseck, J., Field, M., Fischer, S., Gao, J., Guo, H., Ha, S., Joseph-McCarthy, D., Kuchnir, L., Kuczera, K., Lau, F., Mattos, C., Michnick, S., Ngo, T., Nguyen, D., Prodhom, B., Reiher, W., Roux, B., Schlenkrich, M., Smith, J., Stote, R., Straub, J., Watanabe, M., Wiorkiewicz-Kuczera, J., Yin, D., and Karplus, M. (1998). All-atom empirical potential for molecular modeling and dynamics studies of proteins. *The Journal of Physical Chemistry B*, 102(18):3586–3616. 31

Marassi, F. and Opella, S. (1998). Nmr structural studies of membrane proteins. *Current opinion in structural biology*, 8(5):640–648. 13

Martelli, P. L., Fariselli, P., Tasco, G., and Casadio, R. (2003). The prediction of membrane protein structure and genome structural annotation. *Comparative and functional genomics*, 4(4):406–409. 36

McGuffin, L. and Jones, D. (2003). Improvement of the genthreader method for genomic fold recognition. *Bioinformatics*, 19(7):874–881. 25

Menke, M., Berger, B., and Cowen, L. (2008). Matt: Local Flexibility Aids Protein Multiple Structure Alignment. *PLoS Comput Biol*, 4(1):e10. 22

Michino, M., Abola, E., participants, G. D. ., Brooks, C. L., Dixon, J. S., Moult, J., and Stevens, R. C.

(2009). Community-wide assessment of GPCR structure modelling and ligand docking: GPCR Dock 2008. *Nature Reviews Drug Discovery*, 8(6):455–463. 132

Mizuguchi, K., Deane, C. M., Blundell, T. L., Johnson, M. S., and Overington, J. P. (1998a). JOY: protein sequence-structure representation and analysis. *Bioinformatics*, 14(7):617–623. 27, 63, 72, 113

Mizuguchi, K., Deane, C. M., Blundell, T. L., and Overington, J. P. (1998b). HOMSTRAD: a database of protein structure alignments for homologous families. *Protein Sci*, 7(11):2469–2471. 28, 72

Mizuguchi, K., Sele, M., and Cubellis, M. V. (2007). Environment specific substitution tables for thermophilic proteins. *BMC Bioinformatics*, 8(Suppl 1):S15. 69

Mokrab, Y., Stevens, T., and Mizuguchi, K. (2010). A structural dissection of amino acid substitutions in helical transmembrane proteins. *Proteins*, 78(14):2895–2907. 67, 69, 70, 90, 93, 95

Moult, J., Fidelis, K., Kryshtafovych, A., Rost, B., and Tramontano, A. (2009). Critical assessment of methods of protein structure prediction - Round VIII. *Proteins: Structure, Function, and Bioinformatics*, 77(S9):1–4. 14

Mukherjee, S. and Zhang, Y. (2009). Mm-align: a quick algorithm for aligning multiple-chain protein complex structures using iterative dynamic programming. *Nucleic acids research*, 37(11):e83. 22, 59, 166

Müller, D. J., Wu, N., and Palczewski, K. (2008). Vertebrate membrane proteins: structure, function, and insights from biophysical approaches. *Pharmacological reviews*, 60(1):43–78. 1, 12, 13

Müller, T., Rahmann, S., and Rehmsmeier, M. (2001). Non-symmetric score matrices and the detection of homologous transmembrane proteins. *Bioinformatics*, 17(Suppl 1):S182–S189. 10

Murzin, A. G., Brenner, S. E., Hubbard, T., and Chothia, C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J Mol Biol*, 247(4):536–540. 11, 81

Needleman, S. B. and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453. 18

Ng, P., Henikoff, J., and Henikoff, S. (2000). Phat: a transmembrane-specific substitution matrix. *Bioinformatics*, 16(9):760–766. 10, 82

Notredame, C. (2000). T-coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205–217. 19, 20

Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B., and Thornton, J. M. (1997). CATH–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108. 11

Ortiz, A. R., Strauss, C. E., and Olmea, O. (2002). MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison. *Protein science*, 11(11):2606–2621. 17, 22

Ovchinnikov, Y. A., Abdulaev, N. G., Feigina, M. Y., Kiselev, A. V., and Lobanov, N. A. (1977). Recent findings in the structure—functional characteristics of bacteriorhodopsin. *FEBS Letters*, 84(1):1–4. 65

Pearson, W. (1990). Rapid and sensitive sequence comparison with fastp and fasta. *Methods in enzymology*, 183:63–98. 16, 40

Petrey, D., Xiang, Z., Tang, C. L., Xie, L., Gimpelev, M., Mitros, T., Soto, C. S., Goldsmith-Fischman, S., Kernytsky, A., Schlessinger, A., Koh, I. Y., Alexov, E., and Honig, B. (2003). Using multiple structure alignments, fast model building, and energetic analysis in fold recognition and homology modeling. *Proteins*, 53(Suppl 6):430–435. 17, 108

Punta, M., Forrest, L. R., Bigelow, H., Kernytsky, A., Liu, J., and Rost, B. (2007). Membrane protein

prediction methods. *Methods (San Diego, Calif.)*, 41(4):460–474. 108

Ramos, M. and Fernandes, P. (2006). Atomic-level rational drug design. *Current Computer - Aided Drug Design*, 2(1):57–81. 11

Reddy, C. S. h. S., Vijayasarathy, K., Srinivas, E., Sastry, G. M., and Sastry, G. N. (2006). Homology modeling of membrane proteins: a critical assessment. *Computational biology and chemistry*, 30(2):120–126. 110

Rees, D., DeAntonio, L., and Eisenberg, D. (1989). Hydrophobic organization of membrane proteins. *Science.*, 245(4917):510–513. 65

Renault, M., Cukkemane, A., and Baldus, M. (2010). Solid-state nmr spectroscopy on complex biomolecules. *Angewandte Chemie (International ed. in English)*, 49(45):8346–8357. 13

Rost, B. (1999). Twilight zone of protein sequence alignments. *Protein engineering*, 12(2):85–94. 56, 114, 122

Sadreyev, R. I., Tang, M., Kim, B.-H. H., and Grishin, N. V. (2007). COMPASS server for remote homology inference. *Nucleic acids research*, 35(Suppl 2):W653–W658. 20

Sali, A. and Blundell, T. (1990). Definition of general topological equivalence in protein structures. a procedure involving comparison of properties and relationships through simulated annealing and dynamic programming. *J Mol Biol*, 212(2):403–428. 72

Sali, A. and Blundell, T. L. (1993). Comparative protein modelling by satisfaction of spatial restraints. *Journal of Molecular Biology*, 234(3):779–815. 15, 17, 31, 108, 109

Sali, A. and Overington, J. P. (1994). Derivation of rules for comparative protein modeling from a database of protein structure alignments. *Protein Sci*, 3(9):1582–1596. 110

Sánchez, R. and Sali, A. (1997). Advances in comparative protein-structure modelling. *Current opinion in structural biology*, 7(2):206–214. 17, 62, 64, 85

Saxena, A. K., Alam, I., Dixit, A., and Saxena, M. (2008). Internet resources in GPCR modelling. *SAR and QSAR in environmental research*, 19(1-2):11–25. 110

Sayers, E. W., Barrett, T., Benson, D. A., Bryant, S. H., Canese, K., Chetvernin, V., Church, D. M., DiCuccio, M., Edgar, R., Federhen, S., Feolo, M., Geer, L. Y., Helmberg, W., Kapustin, Y., Landsman, D., Lipman, D. J., Madden, T. L., Maglott, D. R., Miller, V., Mizrachi, I., Ostell, J., Pruitt, K. D., Schuler, G. D., Sequeira, E., Sherry, S. T., Shumway, M., Sirotkin, K., Souvorov, A., Starchenko, G., Tatusova, T. A., Wagner, L., Yaschenko, E., and Ye, J. (2009). Database resources of the National Center for Biotechnology Information. *Nucleic acids research*, 37(Suppl 1):D5–D15. 12, 71

Schrama, D., Reisfeld, R., and Becker, J. (2006). Antibody targeted drugs as cancer therapeutics. *Nature Reviews Drug Discovery*, 5(2):147–159. 11

Schulz, G. E. (2002). The structure of bacterial outer membrane proteins. *Biochimica et biophysica acta*, 1565(2):308–317. 108

Schwede, T., Kopp, J., Guex, N., and Peitsch, M. C. (2003). SWISS-MODEL: an automated protein homology-modeling server. *Nucl. Acids Res.*, 31(13):3381–3385. 17, 29, 108

Scott, K. A., Bond, P. J., Ivetac, A., Chetwynd, A. P., Khalid, S., and Sansom, M. S. (2008). Coarse-Grained MD Simulations of Membrane Protein-Bilayer Self-Assembly. *Structure*, 16(4):621–630. 37, 81, 114, 123

Senes, A., Gerstein, M., and Engelman, D. (2000). Statistical analysis of amino acid patterns in transmembrane helices: the gxxxg motif occurs frequently and in association with beta-branched residues at neighboring positions. *Journal of molecular biology*, 296(3):921–936. 68, 101

Shi, J., Blundell, T. L., and Mizuguchi, K. (2001). FUGUE: sequence-structure homology recognition

using environment-specific substitution tables and structure-dependent gap penalties. *Journal of molecular biology*, 310(1):243–257. 10, 16, 17, 25, 27, 29, 74

Shindyalov, I. N. and Bourne, P. E. (1998). Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng*, 11(9):739–747. 17, 22

Siew, N., Elofsson, A., Rychlewski, L., and Fischer, D. (2000). MaxSub: an automated measure for the assessment of protein structure prediction quality. *Bioinformatics*, 16(9):776–785. 23

Simons, K., Bonneau, R., Ruczinski, I., and Baker, D. (1999). Ab initio protein structure prediction of casp iii targets using rosetta. *Proteins*, 37(Suppl 3):171–176. 15

Sippl, M. and Weitckus, S. (1992). Detection of native-like models for amino acid sequences of unknown three-dimensional structure in a data base of known protein conformations. *Proteins*, 13(3):258–271. 25

Skolnick, J., Kihara, D., and Zhang, Y. (2004). Development and large scale benchmark testing of the prospector_3 threading algorithm. *Proteins*, 56(3):502–518. 25

Smith, T. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197. 18

Sneath and Sokal (1973). *Numerical Taxonomy*. W.H. Freeman and Company, San Francisco. 174

Söding, J. (2005). Protein homology detection by hmm-hmm comparison. *Bioinformatics*, 21(7):951–960. 16, 25

Sonnhammer, E. L., von Heijne, G., and Krogh, A. (1998). A hidden Markov model for predicting transmembrane helices in protein sequences. *Proc Int Conf Intell Syst Mol Biol*, 6:175–182. 35

Stevens, T. J. and Arkin, I. T. (1999). Are membrane proteins "inside-out" proteins? *Proteins*, 36(1):135–143. 108

Sutcliffe, M. J., Haneef, I., Carney, D., and Blundell, T. L. (1987). Knowledge based modelling of homologous proteins, Part I: Three-dimensional frameworks derived from the simultaneous superposition of multiple structures. *Protein Eng*, 1(5):377–384. 17, 29

Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994). CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680. 20, 21

Tusnády, G. E., Dosztányi, Z., and Simon, I. (2005). PDB_TM: selection and membrane localization of transmembrane proteins in the protein data bank. *Nucleic acids research*, 33(Suppl 1):D275–D278. 12, 54, 71, 81, 123

Ulmschneider, M. B. and Sansom, M. S. (2001). Amino acid distributions in integral membrane protein structures. *Biochimica et biophysica acta*, 1512(1):1–14. 67, 68, 90, 101, 108

Vakser, I. A. (1995). Protein docking for low-resolution structures. *Protein Eng.*, 8(4):371–378. 12

van der Spoel, D., Lindahl, E., Hess, B., Groenhof, G., Mark, A. E., and Berendsen, H. J. (2005). GROMACS: Fast, flexible, and free. *J. Comput. Chem.*, 26(16):1701–1718. 38

von Heijne, G. (1986). The distribution of positively charged residues in bacterial inner membrane proteins correlates with the trans-membrane topology. *The EMBO journal*, 5(11):3021–3027. 35, 65

von Heijne, G. (1992). Membrane protein structure prediction: Hydrophobicity analysis and the positive-inside rule. *Journal of Molecular Biology*, 225(2):487 – 494. 65

von Heijne, G. (1994). Membrane Proteins: From Sequence to Structure. *Annual Review of Biophysics and Biomolecular Structure*, 23(1):167–192. 35, 65

von Heijne, G. (1995). Membrane protein assembly: Rules of the game. *Bioessays*, 17(1):25–30. 66

von Heijne, G. (2007). The membrane protein universe: what's out there and why bother? *Journal of*

Internal Medicine, 261(6):543–557. 1, 66

Wallin, E. and von Heijne, G. (1998). Genome-wide analysis of integral membrane proteins from eubacterial, archaean, and eukaryotic organisms. *Protein science : a publication of the Protein Society*, 7(4):1029–1038. 108

Wallner, B. and Elofsson, A. (2005). All are not equal: A benchmark of different homology modeling programs. *Protein Science*, 14(5):1315–1327. 109, 124

Wang, G. and Dunbrack, R. (2003). Pisces: a protein sequence culling server. *Bioinformatics*, 19(12):1589–1591. 151

Wang, L. and Jiang, T. (1994). On the complexity of multiple sequence alignment. *Journal of computational biology : a journal of computational molecular cell biology*, 1(4):337–348. 19

Ward, J. H. J. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244. 78

Wu, S. and Zhang, Y. (2008). Muster: Improving protein sequence profile-profile alignments by using multiple sources of structure information. *Proteins*, 72(2):547–556. 25

Wuthrich, K. (1989). Protein structure determination in solution by nuclear magnetic resonance spectroscopy. *Science*, 243(4887):45–50. 13

Yang, L. and Nolan, J. P. (2007). High-throughput screening and characterization of clones selected from phage display libraries. *Cytometry A*, 71(8):625–631. 11

Yang, Q., Du, L., Wang, X., Li, M., and You, Q. (2008). Modeling the binding modes of Kv1.5 potassium channel and blockers. *Journal of Molecular Graphics and Modelling*, 27(2):178–187. 110

Yarov-Yarovoy, V., Schonbrun, J., and Baker, D. (2006). Multipass membrane protein structure prediction using Rosetta. *Proteins*, 62(4):1010–1025. 109

Yuen, C., Davidson, A., and Deber, C. (2000). Role of aromatic residues at the lipid-water interface in micelle-bound bacteriophage m13 major coat protein. *Biochemistry*, 39(51):16155–16162. 68

Zemla, A., Venclovas, C., Moult, J., and Fidelis, K. (1999). Processing and analysis of CASP3 protein structure predictions. *Proteins: Structure, Function, and Genetics*, 37(S3):22–29. 23, 122

Zhang, Y. and Skolnick, J. (2004). Scoring function for automated assessment of protein structure template quality. *Proteins*, 57(4):702–710. 23

Zhang, Y. and Skolnick, J. (2005a). The protein structure prediction problem could be solved using the current PDB library. *Proceedings of the National Academy of Sciences of the United States of America*, 102(4):1029–1034. 17, 22, 123

Zhang, Y. and Skolnick, J. (2005b). TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic Acids Research*, 33(7):2302–2309. 81

Zhou, H. and Zhou, Y. (2005). Fold recognition by combining sequence profiles derived from evolution and from depth-dependent structural alignment of fragments. *Proteins*, 58(2):321–328. 25