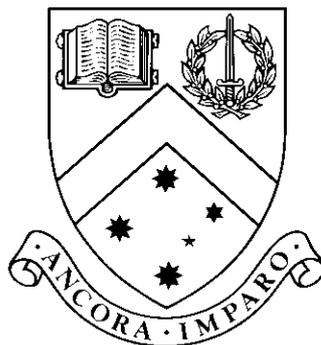


Statistical Inductive Inference of Protein Structural Alignments

by

James Collier



Thesis

Submitted by James Collier

in fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Supervisors:

Dr. Arun Konagurthu & Prof. Maria Garcia de la Banda

**Clayton School of Information Technology
Monash University**

September, 2016

Statistical Inductive Inference of Protein Structural Alignments

Copyright © 2016 James Collier
Some Rights Reserved (see Appendix B)
Clayton School of Information Technology
Monash University
Australia

This thesis is required by the examinations office of Monash University to contain the following copyright notices:

1. Under the Copyright Act 1968, this thesis must be used only under the normal conditions of scholarly fair dealing. In particular no results or conclusions should be extracted from it, nor should it be copied or closely paraphrased in whole or in part without the written consent of the author. Proper written acknowledgement should be made for any assistance obtained from this thesis.
2. I certify that I have made all reasonable efforts to secure copyright permissions for third-party content included in this thesis and have not knowingly added copyright content to my work without the owner's permission.

This thesis was typeset with \LaTeX by the author using GNU Emacs and the $\text{AU}\text{T}\text{E}\text{X}$ macros. $\text{E}\text{T}\text{E}\text{X}$ is a collection of macros for TEX . TEX is a trademark of the American Mathematical Society. The document class used in formatting this thesis was written by Glenn Maughan and modified by Dean Thompson and David Squire of Monash University.

*“Take the risk of thinking for yourself. Much more happiness,
truth, beauty, and wisdom will come to you that way.”*

— Christopher Hitchens

Contents

List of Symbols & Abbreviations	vii
List of Tables	viii
List of Figures	ix
Abstract	xi
Publications	xiii
Acknowledgements	xiv
1 Introduction	1
1.1 Inconsistencies in Protein Structural Alignment Algorithms	2
1.2 Aim of This Thesis	3
1.3 Contributions Made by This Thesis	3
1.3.1 Ancillary Contributions	4
1.4 Summary	5
1.5 Thesis Outline	5
2 Introduction to Proteins and Alignments	7
2.1 Introduction to Proteins	8
2.1.1 Conceptual Hierarchy of Protein Structures	10
2.1.2 Protein Structure Databases and File Formats	10
2.1.3 Structure Determination and Prediction	12
2.1.4 Recurrent Substructural Themes and Classification	13
2.1.5 Geometric Constraints on the Protein Backbone Atoms	15
2.1.6 Topology of Proteins and Structural Plasticity	16
2.2 Alignment of Proteins	17
2.2.1 Protein Structural Alignment	19
2.2.2 Formulation of the Structural Alignment Problem	19
2.2.3 Protein Structural Superposition	20
2.2.4 Internal Representations Used for Structural Alignment	25
2.2.5 Protein Structural Alignment Scoring Functions	27
2.3 Summary	31
3 Introduction to Statistical Inference	33
3.1 Definitions and Notations	34
3.2 Probability, Information and Entropy	35
3.3 Statistical Inference, Model Comparison and Selection	37

3.3.1	Notations Supporting Statistical Inference	37
3.3.2	Statistical Estimators and Common Methods of Parameter Estimation	38
3.3.3	Minimum Message Length Inference	41
3.4	Codewords, Prefix-Free Codes and Universal Codes for Integers	49
3.5	Summary	50
4	An MML Framework for Assessing Alignment Quality	51
4.1	Introduction	52
4.2	Review of Popular Alignment Quality Measures	53
4.3	Structural Alignment as an Inductive Inference Problem	57
4.3.1	An Information Measure of Structural Alignment Quality	58
4.3.2	Statistical Properties of the Information Measure	59
4.4	Formulation of the Alignment Encoding Message Length: $I(\mathcal{A})$	61
4.4.1	Selecting an Alignment Encoding Scheme	64
4.5	Formulation of the Null Model Message Length: $I_{\text{null}}(\cdot)$	64
4.5.1	Selecting a Null Encoding Scheme	67
4.6	Formulation of the Coordinate Compression Model: $I(T S, \mathcal{A})$	68
4.7	Handling Shifts and Rotations	71
4.8	Results and Discussion	72
4.8.1	Selection of Domains from the SCOP Database	73
4.8.2	Experiment 1: Benchmarking Against the SCOP Hierarchy	73
4.8.3	Experiment 2: Level of Disagreement Between Measures of Alignment Quality	77
4.9	Conclusions	79
5	I-value: A Measure of Alignment Quality	81
5.1	Introduction	82
5.2	Improved Encoding Schemes for I -value	83
5.2.1	Improvement to the Alignment Encoding Model: $I(\mathcal{A})$	83
5.2.2	Improved Estimation of the Null Model Message Length: $I_{\text{null}}(\cdot)$	86
5.2.3	Improved Estimation of the Coordinate Compression Model: $I(T S, \mathcal{A})$	93
5.3	Results and Discussion	98
5.4	Conclusions	100
6	Searching for Pairwise Structural Alignments	103
6.1	Introduction	104
6.2	Structural Alignment Search Methods	104
6.2.1	Assembly of Well-Fitting Fragment-Pairs	105
6.2.2	Local Search Methods	106
6.2.3	Search by Alternating Superposition and Alignment	108
6.2.4	Dynamic Programming Based Methods for Structural Alignment	109
6.3	Searching for Structural Alignments Using I -value	112
6.3.1	Dynamic Programming Using the I -value Measure	112
6.3.2	The MMLigner Algorithm	116
6.4	Results and Discussion	124
6.4.1	Identification of Alternate Structural Alignments	124
6.4.2	Performance of MMLigner on Hard Structural Alignment Cases	125
6.4.3	Large Scale Comparison on SCOP Domains with Varying Structural Distance	128

6.5	Conclusions	134
7	Ancillary Methods	135
7.1	An Information Measure for Comparing Top k Lists	136
7.1.1	Information Measure for Comparing Ranked Lists	137
7.1.2	Realising the Information Measure in Practice	140
7.1.3	Results and Discussion	144
7.2	Sufficient Statistics for Least-Squares Superposition	146
7.2.1	Introduction	146
7.2.2	Summary of Least-Squares Superposition	147
7.2.3	Sufficient Statistics	149
7.2.4	Updating Sufficient Statistics	152
7.2.5	Computing the RMSD from Updated Sufficient Statistics	156
7.2.6	Results and Discussion	156
7.3	Conclusions	158
8	Conclusions and Future Directions	161
8.1	Extensions to the Research Presented in this Thesis	163
8.1.1	Evaluating the Quality of Predicted Protein Structures	163
8.1.2	Improvements to the Encoding Models	163
8.1.3	Identifying Closely Competing Structural Alignments	164
8.1.4	Visualisations of Alignment Quality	164
	References	171
	Appendix A List of randomly selected SCOP heirarchy domains	189
	Appendix B Creative Commons Attribution-NoDerivatives 4.0 International	193

List of Symbols & Abbreviations

$\text{Pr}(\cdot)$ Probability

$I(\cdot)$ Shannon information content = $-\log(\text{Pr}(\cdot))$

Å Angstrom. $1\text{Å} = 1 \times 10^{-8}$ cm

DNA Deoxyribonucleic acid

EM Expectation Maximisation

FSA Finite State Automata

IQR Inter-Quartile Range

MML Minimum Message Length

PDB Protein Data Bank

PDF Probability Density Function

PoM Precision of measurement

PoPV Precision of Parameter Value

RMSD Root mean square deviation

SMML Strict Minimum Message Length

wwPDB World-wide Protein Data Bank

List of Tables

2.1	Amino acids	9
2.2	Protein structural alignment quality scoring functions	28
4.1	Adaptive First-Order Markov transmission example	64
4.2	Performance comparison of alignment programs on the SCOP hierarchy	77
6.1	Comparison between alignment programs on calcium binding domains	126
6.2	Performance of alignment programs on difficult to align structures	128
6.3	Performance comparison of alignment programs on the SCOP hierarchy	131
7.1	Time taken to perform exhaustive joint superpositions on a library of well-fitting fragment pairs between two structures from different families.	158

List of Figures

2.1	Amino acid chemical structure	8
2.2	Conceptual hierarchy of protein structure	11
2.3	wwPDB coordinate file formats	12
2.4	Common supersecondary motifs	15
2.5	Protein main-chain torsion angles	16
2.6	Ramachandran-Ramakrishnan-Sasisekharan plot	17
2.7	A Topology of Protein Structure diagram	18
2.8	An example hinge rotation	18
2.9	A pairwise protein structural alignment	20
2.10	Tessellated Representation of Protein Structure	26
2.11	Example tableau representation for a protein structure	27
4.1	Alignment Finite State Automata	61
4.2	The three-state automata used for alignment encoding	63
4.3	Comparison of alignment encoding schemes	65
4.4	Transmitting a coordinate on the surface of a discretised sphere	67
4.5	Comparison of null model encoding schemes	68
4.6	Adaptive superposition	69
4.7	Example of an alignment between structures containing a hinge rotation	71
4.8	Evaluation of scores using SCOP hierarchy	75
4.9	Evaluation of scores using SCOP hierarchy (continued)	76
4.10	Levels of agreement between structural alignment scoring functions	78
5.1	A three-state automata used for alignment encoding	84
5.2	Illustration of an alignment with long terminal gaps	84
5.3	Comparison of alignment encoding schemes	86
5.4	Acidity data containing 155 real valued data points.	87
5.5	Fidelity of mixture models	89
5.6	Measuring the canonical direction of any C_α atom	90
5.7	Performance of improved null model encoding	92
5.8	Global coordinate compression model example	94
5.9	Posterior reweighting of mixtures of directional distributions	96
5.10	Improvement in message length for the \mathcal{R} -model	99
6.1	Alignment quality landscape visualisation	115
6.2	Phases of the MMLigner algorithm	118
6.3	Alternative MMLigner alignments for the SCOP domains <code>d2sasa_</code> and <code>d1jffa_</code>	127
6.4	Large scale benchmarking of MMLigner on SCOP data	129
6.5	Evaluation of alignment programs using the SCOP hierarchy	132

6.6	Evaluation of alignment programs using the SCOP hierarchy (continued)	133
7.1	Examples of the adaptive encoding schemes for bit masks described in the main text.	142
7.2	Translation table for factoradic numbers	143
7.3	Variation of costs over the set of all permutations	144
7.4	Comparison between distance measures on movie rankings	145
7.5	Comparison between distance measures on search engine results	145
7.6	Runtimes for joint superpositions	157
8.1	Landscape for wwPDB 1HH0-A vs. wwPDB 1MBD	165
8.2	Landscape for wwPDB 3CHY vs. wwPDB 5NLL	166
8.3	Landscape for wwPDB 1PHN vs. wwPDB 1MBD	167
8.4	Landscape for wwPDB 1EUD-A vs. wwPDB 1CCW-A	168
8.5	Landscape for wwPDB 1JFJ-A vs. wwPDB 2SAS-A	169

Statistical Inductive Inference of Protein Structural Alignments

James Collier
Monash University, 2016

Supervisors:
Dr. Arun Konagurthu & Prof. Maria Garcia de la Banda

Abstract

Proteins are complex biological molecules that perform a vast array of functions crucial to life. A small set of computational tasks underpin the study of proteins. One of these supports the comparison of proteins using the notion of *alignment*. An alignment between proteins allows biologists to understand their evolutionary relationship. Due to the functional constraints that exist on protein biomolecules, finding reliable alignments requires the comparison of their three-dimensional structures (rather than their sequences). The resulting alignments are called protein structural alignments (rather than sequence alignments). The quality of alignments has important consequences for research in protein biology, as they are the foundation for many aspects of protein research.

The problem of finding reliable structural alignments is commonly posed as a combinatorial optimisation problem, which requires an optimisation strategy (a search method to find the best alignments) and an objective function (a measure of alignment quality). The objective function must arbitrate a trade-off between the structural fidelity of the proteins being aligned, and the complexity of the alignment itself. The alignment search algorithm then finds the alignment that the scoring function considers optimal. Over the past five decades, many alignment methods have been conceived to identify structural alignments between proteins. Concerningly, the alignments obtained by these methods differ substantially and often produce contradictory results. Many comparative studies on methods generating structural alignments have highlighted the absence of a clear consensus on what constitutes a good structural alignment and the lack of a statistically rigorous measure of alignment quality. This has been stated as a leading cause of the observed proliferation of new structural alignment methods, which tend to perform small modifications to previous approaches.

This thesis proposes a fundamental shift in the way structural alignment quality is formalised and measured, and in the way biologically-meaningful alignments are identified. It brings together ideas from fields of information theory, data compression, and statistical inductive inference to develop a statistically rigorous framework to measure structural alignment quality. The resulting alignment quality measure, called *I*-value, is built on the Bayesian framework of minimum message length inference. Furthermore, this thesis develops a search algorithm that employs *I*-value to consistently identify high quality and statistically significant structural alignments. This search method is also able to identify significant alternative structural alignments of comparable quality. The culmination of this work is an open-source pairwise structural alignment program called **MMLigner** (available from <http://lcb.infotech.monash.edu.au/mmligner>). The performance of **MMLigner** is benchmarked against popular alignment programs and alignment scoring functions. **MMLigner** results were found to be highly-competitive compared to other methods, and consistently outperforms other methods in identifying alternative structural alignments, a challenging problem when aligning oligomeric proteins and protein complexes.

Statistical Inductive Inference of Protein Structural Alignments

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

James Collier
September 19, 2016

Publications

Manuscripts in communication:

- **James H. Collier**, Lloyd Allison, Arthur M. Lesk, Peter J. Stuckey, Maria Garcia de la Banda, & Arun S. Konagurthu. Statistical Inference of Protein Structural Alignments. *In communication* (Preprint available at **URL:** <http://biorxiv.org/content/early/2016/06/02/056598>).

Accepted manuscripts:

- **James H. Collier**, Lloyd Allison, Arthur M. Lesk, Maria Garcia de la Banda, Arun S. Konagurthu (2014), A new statistical framework to assess structural alignment quality using information compression. *Bioinformatics*. **30**(17):i512–i518.
(Presented at the 13th Annual European Conference on Computational Biology (ECCB 2014), September 7-10 2014, Strasbourg, France.)
- Arun S. Konagurthu, Parthan Kasarapu, Lloyd Allison, **James H. Collier**, and Arthur M. Lesk (2015), On Sufficient Statistics of Least-Squares Superposition of Vector Sets. *Journal of Computational Biology*. **22**(6):487–497.
- **James H. Collier**, Arun S. Konagurthu (2014), An information measure for comparing top k lists. In *IEEE 10th International Conference on eScience*. pp. 127–134.

Acknowledgements

My sincere and heartfelt thanks go to my advisers Dr. Arun Konagurthu and Prof. Maria Garcia de la Banda. They are inspirational people and I hope that, one day, I am able to live up to the high standards they set. From them I have learned how little I know, but they have instilled in me a confidence and a sense of excitement to explore the unknown. I consider myself extraordinarily lucky to have been given a chance to learn and develop under their careful and patient guidance.

I interacted with Arun almost on a daily basis. He is filled with an enthusiasm for, and joy in science and mathematics. From the day I met him, his enthusiasm has been infectious. He inspires in me a desire to learn more about whichever of the broad subjects we often find ourselves discussing. His deep understanding of a diverse range of fields has been a constant and indispensable aid to my completing this candidature. The kindness and dignity with which he treats students and his honest approach to the research effort are principles that I have taken to heart.

Maria is a truly amazing person. While juggling a busy schedule and life as a mother, researcher, and faculty dean she never once missed (or was even late) to a meeting or a chance to guide me in the right direction. She always gave insightful feedback on any issue I was having, no matter how badly I described it to her. The attention to detail that she has while constantly keeping the big picture in mind is splendid and has ensured I stayed on track.

I also sincerely thank Lloyd Allison for his insights regarding the mathematics, Arthur Lesk from which much of my biological understanding stems, and Peter Stuckey all of whom were extraordinarily helpful. Much of my progress would not have been possible without them. Furthermore, my sincere thanks go to Parthan Kasarapu, who sat with me throughout this candidature, often 'lent me his ear' and is a source of great mathematical prowess.

I am grateful to the Australian Government, as representatives of the Australian people, for their generous funding under the Australian Postgraduate Award (APA) scholarship. Furthermore, I wish to acknowledge Monash University and NICTA for additional funding provided. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. Lastly, thanks go to the Victorian Life Sciences Computation Initiative (VLSCI) for the generous travel funding they provided.

A very special thanks goes to the lovely and delightful Nicole Alers who has been my rock through many of the trials of this candidature, you have a special place in my heart. My dear parents who've lent their unwavering support for me, I cannot thank you enough. My sincere thanks go to James Morgan and Yen Pham Morgan for their extraordinary support and willingness to put up with me. My dear friend Gopika Krishnamurthy deserves very

special thanks for her encouragement, optimism and kindness. I owe a great debt of gratitude to Ehsan Shareghi, Kai Siong Yow, Xuhui Zang, Daniel Bishop, Marion Fumaroli, Meaghan Bruce, Benjamin Lambell, Andrew Read, Shannon Scullin, Michael Eichmair, and many others who have helped and inspired me, in various ways throughout this amazing journey. Each of you have my deepest, most profound and sincere thanks. Without each and every one of you, this thesis would not have come to fruition.

James Collier

Monash University
September 2016

Chapter 1

Introduction

Proteins are large, complex biological molecules that are crucial to all forms of life. They perform a vast range of different functions, which include serving as catalysts of biochemical reactions, translating genetic information into other macromolecular forms, transporting molecular payloads, and acting as storage mechanisms for the essential chemical ingredients of life (Lesk, 2010).

Over the last seventy years, experimental techniques have emerged that allow researchers to accurately determine the underlying amino acid sequence and the three-dimensional (3D) structure of proteins. These experimental techniques have given rise to some of the most prominent data streams in all of molecular biology. The direct result of these experimental advances is the rapid growth of publicly available protein sequence (Universal Protein Resource: Uniprot (2010)) and structure (Worldwide Protein Data Bank: Berman et al. (2002)) databases. The computational analysis of this protein data has become invaluable for biological research.

A critical computational task in protein biology is the comparison of protein sequence and structure using a concept known as *alignment*. Specifically, an alignment is an assignment of one-to-one correspondences between a subset of the amino acid residues in two or more proteins. Proteins evolve continuously, subject to strong functional constraints (Lesk, 2001a), and the correspondences assigned by alignments are used to establish an evolutionary relationship (*homology*) between the proteins under comparison. In particular, an alignment is used to answer two specific questions: *Are two proteins related? And, if so, how?* The identification of this relationship provides a basis for estimating the evolutionary distances between proteins, and for discovering how proteins have evolved (diverged) from a common ancestor, in some cases, over millions of years.

Alignments can be computed using the protein's amino acid sequence information or using its 3D structure (or a combination of both). However, protein structures change more conservatively than their sequences during evolution. Therefore, the structural alignment of proteins offers a more accurate basis to establish homology, and is capable of establishing homology even for very distantly related proteins (Chothia and Lesk, 1986).

The quality of alignments between proteins has important consequences for research in molecular biology. For example, without high quality alignments, the characterisation of important biological processes such as ligand binding would be far more difficult (Marti-Renom et al., 2009). In many cases, protein structural alignments alone are used for automatic hierarchical classification of the known protein fold space. In these cases, high quality alignments are essential for the classification to be useful (Holm et al., 1992; Orengo et al., 1997). As a protein's structure is so intricately tied to its function, high quality structural alignments permit the inference of function from the protein structure (Godzik et al., 2007). Furthermore, it

has been suggested that the prediction of protein structure from sequence is limited mainly by errors in alignments (Zhang and Skolnick, 2005a). And importantly, the experimental solution of protein crystal structure by molecular replacement also depends on high quality structural alignments (Konagurthu et al., 2006).

1.1 Inconsistencies in Protein Structural Alignment Algorithms

Finding a high quality structural alignment for two or more proteins can be considered as an optimisation problem requiring a *measure of quality* to assess any proposed alignment, and a *search method* to find an optimal alignment under the stated quality measure. Over the past five decades, many computational methods have been proposed to identify protein structural alignments, with the number of such new structural alignment methods doubling roughly every five years (Hasegawa and Holm, 2009). These methods differ mainly in how they assess the quality of an alignment. Concerningly, several studies have shown that the alignments obtained by these methods differ substantially (Kolodny et al., 2005; Sippl and Wiederstein, 2008; Hasegawa and Holm, 2009; Slater et al., 2013; Ma and Wang, 2014). In a recent review, Liisa Holm (Hasegawa and Holm, 2009), a leading researcher in the field, and author of DALI (Holm and Sander, 1993), a widely-used structural alignment program, writes:

“A variety of methods use different representations, scoring functions, and optimization algorithms, and they generate contradictory results even for moderately distant structures.”

Manfred Sippl (Slater et al., 2013), another leading expert in the field, comments,

“Lacking a clear definition of what constitutes an optimal alignment, structural bioinformatics has created an ever growing arsenal of computer programs, all dedicated to solve one and the same problem, where each program provides its own numerical criteria and scores to describe the quality of the alignments obtained. From a user’s point of view the situation is most confusing. Research on protein structure requires structure alignment tools but it is unclear which programs produce the most valuable results and how the results obtained have to be interpreted. Obviously some effort for standardization is required.”

The above comments highlight a severe disconnect between the rapidly growing number of methods and the quality of the structural alignments they generate. This disconnect is due to a complete lack of consensus on how to measure structural alignment quality. To resolve this, it is necessary to define a rigorous measure of alignment quality that can objectively discriminate between competing alignments. Without this, the notion of best (or *optimal*) alignment can neither be rigorously defined nor searched for (Hasegawa and Holm, 2009).

The absence of such a rigorous measure has fueled the proliferation of methods based on ad hoc scoring functions. Further, the search algorithms used by existing methods are typically based on greedy heuristics and, thus, often return alignments that are far from optimal, regardless of the chosen scoring function (Konagurthu et al., 2006). Finally, current methods do not provide any framework to meaningfully capture similarities and differences between competing alignments, as they often generate a single alignment result, while ignoring a multitude of other closely-competing alignments. New methods are therefore needed to meaningfully explore the

entire landscape of competing alignments so that biologically interesting relationships are not overlooked.

The traditional approach of formulating an ad hoc scoring function from key alignment quality criteria has been extensively explored over the last four decades. Further development along the same lines is unlikely to provide any major breakthrough. Therefore, the field of structural alignment will stand to benefit by departing from traditional approaches and exploring radically new ones.

1.2 Aim of This Thesis

This thesis aims not merely to tinker along the methodological lines followed by previous attempts, but rather, it aspires to fundamentally shift the way structural alignment quality is formalised and measured, and the way in which biologically-meaningful alignments are identified. This involves achieving the following objectives:

- Use information theory to develop a framework to measure structural alignment quality with statistical properties that provide rigorous evaluation of alignments, objective differentiation between competing alignments, and a robust test of alignment significance.
- Design a search algorithm capable of consistently identifying high quality, statistically significant structural alignments using the above measure of quality.
- Be able to identify considerably different, competing alternative alignments that are statistically significant and should not be overlooked.

1.3 Contributions Made by This Thesis

The first major contribution of this thesis is the development of the *I*-value measure to assess pairwise structural alignment quality. *I*-value treats a structural alignment as a hypothesis of the structural relationship between two proteins, expressed as a one-to-one, order-preserving, correspondence between subsets of residues. Specifically, each alignment hypothesis is seen as an attempt to explain the coordinate data of the pair of proteins. The explanatory power of each alignment is then quantified, using principles of information theory, as the amount of lossless compression obtained from encoding the coordinate data of the proteins using the knowledge of the correspondences provided by the alignment (as compared to encoding the coordinate data without that knowledge). In this context, *I*-value can be thought of as a communication process between an imaginary transmitter-receiver pair. The quality of any structural alignment is measured by *I*-value as the message length required to encode and transmit the protein coordinate data using the alignment hypothesis. The shorter the length of the message, the better the alignment. This measure is backed by mathematical rigour and exhibits important statistical properties, including a natural null hypothesis test for assessing the statistical significance of any proposed alignment. The accuracy of *I*-value depends largely on the specific statistical models of encoding that are used. The development of these encoding schemes represent one of the most complex aspects of this research, and led to an initial set of encoding models, which are described and quantitatively compared in Chapter 4.

In Chapter 5, systematic improvements to the set of statistical models used for coordinate encoding (over the ones used in Chapter 4) are described. These improved encoding schemes build on the statistical models of protein directional data developed by Kasarapu and Allison

(2015). These directional models form the basis of a sophisticated technique to define the relative position and local geometric context of corresponding residues and use this to compress the coordinate data more concisely. This leads to the final implementation of I -value as achieved in this thesis. This implementation can accurately discriminate between closely competing alignments, and can consistently distinguish homologous structures from unrelated structures.

The second major contribution of this thesis is a method to search for high quality alignments of a pair of protein structures, using I -value as the objective for which to optimise. The search for the optimal alignment is an inherently complex problem, which is usually (except in the simplest of cases) solved by heuristic, rather than by complete search methods as it cannot be solved exactly. Several attempts were made at defining a robust search algorithm to optimise alignments using I -value. Early attempts gave rise to a promising method for visualising the entire quality landscape of competing alignments between a pair of structures, which can be seen as a natural development of the dot plot from the field of sequence alignment*. The final search algorithm developed in this thesis, called **MMLigner**, is able to reliably find high quality alignments that achieve significant compression using I -value. **MMLigner** is also able to consistently find a distinct set of statistically significant alternative alignments between the structure pairs, when they exist. This is useful, for example, when aligning multi-domain proteins where a plausible alignment exists for each domain. This gives molecular biologists a powerful tool to examine multiple plausible structural relationships between a pair of proteins, where previous tools often give only a single optimal alignment (see Chapter 6).

1.3.1 Ancillary Contributions

Two other contributions arise from this thesis (described in Chapter 7) providing methodological support to the primary contributions described above. The first one is a highly efficient numerical method to compute least-squares, rigid-body superposition of vector sets using the notion of sufficient statistics. This work derives a set of sufficient statistics for the superposition problem and demonstrates that they possess the crucial property of additivity. This permits a constant time computation of superpositions of vector sets (and their sufficient statistics) when the superpositions of (partial) constituent vector sets are already known, under set addition and symmetric difference operations. This contribution forms the work-horse for generating seed alignments in the final **MMLigner** algorithm (see Chapter 6).

The other ancillary contribution of this thesis is a method to gauge the disorder between ranked, top- k lists. This is traditionally calculated using methods such as the Kendall τ (Kendall, 1938) or Spearman's ρ (Spearman, 1904) rank correlation coefficients or modified versions of these measures (Fagin et al., 2003). The new method presented in this thesis applies information theory (Wallace and Freeman, 1987) to this problem in a similar manner to its application to the alignment problem by I -value. In this instance, one top- k list is used to describe the other, and any differences increase the length of the message. Therefore, two identical lists will require only a short message, whereas, two lists with nothing in common will require a much longer message. This method is, in turn, used to evaluate the degree of agreement between popular measures of structural alignment quality (see Chapter 4).

*Developing this technique further is left to future work.

1.4 Summary

This thesis addresses the protein structural alignment problem by radically departing from the traditional methods of formulating structural alignment scoring functions and finding optimal alignment based on those scoring functions. This is achieved by defining I -value, a structural alignment quality assessment measure with foundations in sound statistical techniques. I -value possesses several useful properties:

1. It achieves an *objective, formal trade-off* between the complexity of an alignment, and the fidelity between the structures defined by that alignment.
2. Alignments can be directly compared using their I -values, and the difference between the I -values for two competing alignments gives their log-odds posterior ratio.
3. I -value provides a natural *null hypothesis* test for statistical significance. An alignment that fails this test can be rejected.
4. I -value is not defined in terms of ad hoc parameters or arbitrary constants.

Furthermore, it goes on to develop and describe **MMLigner**, a search algorithm that uses I -value to optimise a set of distinct structural alignments. Both I -value and **MMLigner** perform highly competitively when benchmarked against state-of-the-art structural alignment methods. The **MMLigner** algorithm is able to consistently identify a range of significant alternative structural alignments, avoids pairing up spurious correspondences. The structural alignments identified by **MMLigner** are highly competitive according to popular measures of structural alignment quality and often succeeds where other alignment programs do not identify any alternative alignments.

1.5 Thesis Outline

The structure of this thesis is as follows:

Chapter 2 provides the biological background necessary for understanding the concepts used throughout this thesis. In particular, it introduces proteins and protein structure, how protein structural data is stored and organised, and how protein structural comparison is performed using protein alignments. The alignment problem is introduced and discussed in terms of three major tasks: representation, scoring, and search. Finally, this chapter reviews representations and scoring functions commonly used by protein structural alignment methods.

Chapter 3 introduces the basics of statistical inference by linking the notions of probability and information. It explores statistical inductive inference by minimum message length (MML) in depth and the practical Wallace-Freeman approximation. Notions of prefix codes and the universal code for positive integers are also briefly introduced here.

Chapter 4 designs and develops the MML based framework for the assessment of protein structural alignment quality. It explores the trade-off between notions of coverage and fidelity, which are made by state-of-the-art alignment quality measures. A rigorous method for making this trade-off is proposed. This chapter then deals with the details of the encoding schemes required to realise the information measure in practise. Finally, this framework of alignment quality is benchmarked over a large set of domain pairs. The quantitative comparison of alignment rankings between popular structural alignment scoring functions highlights the degree of disagreement between structural alignment scoring functions.

Chapter 5 introduces systematic improvements to the encoding schemes proposed in Chapter 4. Notions of mixture modelling and directional probability distributions are introduced in this chapter and used to construct sophisticated coordinate encoding methods. Quantitative comparisons are carried out between these encoding methods to demonstrate the achieved improvement. The encoding schemes developed in this chapter form the definition of the *I*-value measure of protein structural alignment quality as achieved during this candidature.

Chapter 6 deals with methods to search for protein structural alignments by considering the alignment problem as a combinatorial optimisation problem. Common algorithms to search for structural alignments are explored in detail. The development of methods to search for alignments using *I*-value is reviewed and a method to generate an interactive 3D landscape visualisation of alignment quality is introduced. This chapter culminates in the development of the **MMLigner** program, which identifies high quality and statistically significant structural alignments using the *I*-value alignment quality measure. The **MMLigner** program is benchmarked against popular state-of-the-art structural alignment programs and found to be highly competitive.

Chapter 7 discusses the ancillary methodological contributions supporting the work in this thesis as applied to the work within the earlier chapters. Firstly, an information measure for comparing top-*k* ranked lists in a similar vein to the *I*-value comparison of protein structures. This method uses MML to provide an objective trade-off between criteria that measure the dissimilarity between ranked lists, addressing pitfalls in the existing measures. Secondly, a method to rapidly compute joint superpositions in $O(1)$ time is described. This method provides a set of sufficient statistics for the least-squares superposition problem under the least squares criterion. The results in this chapter demonstrate a drastic improvement in the computational effort required to compute least-squares superpositions.

Chapter 8 concludes the thesis by outlining the key results of the research. Future research directions are explored and discussed that arise naturally from the research presented in this thesis. These future research directions include using **MMLigner** and *I*-value to assess entries to the CASP protein structure prediction competition; specific improvements to the encoding models presented in Chapter 5; and three-dimensional interactive visualisation of the landscape of competing alignments.

Chapter 2

Introduction to Proteins and Alignments

“In the drama of life on the molecular scale, proteins are where the action is.”

— A. M. Lesk (2001a)

The research presented in this thesis applies areas of computational science, information theory and statistics to address a biological problem. This chapter introduces the general biological background required for this research. In particular it introduces proteins and their molecular and structural constitution. It also summarises the general principles and concepts related to protein structure and architecture. Furthermore, it explores how protein structures are compared, and introduces the problem of structural comparison by alignment. The biological definitions and elements presented in this chapter supports the research and discussion presented in subsequent chapters.

2.1 Introduction to Proteins

Proteins are long polymer chains made up by *amino acid residues*.^{*} There are twenty naturally occurring amino acid molecules. Each amino acid is composed of an identical set of *main-chain* atoms: a nitrogen (N) atom which is part of the amine group, a central carbon (C_α) atom, and another (carbonyl) carbon atom which is part of the carboxylic group. The chemical structure of an amino acid is shown in Figure 2.1(a). The twenty amino acids are differentiated by the *side-chain* attached to their central carbon atom. A portion of the protein main-chain and side-chain of two successive amino acids is shown in Figure 2.1(b). Each amino acid is uniquely identified by a one-letter code and also a corresponding three-letter code. Refer to Table 2.1 for the complete list of the naturally occurring amino acids, together with their one- and three-letter codes, and chemical structures.

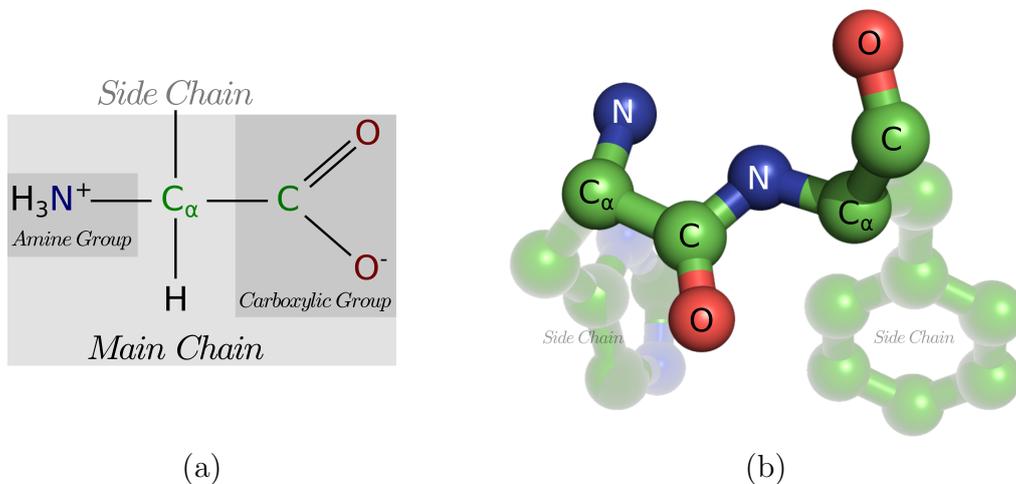


Figure 2.1: The chemical makeup of a protein 3D structure showing the main chain atoms common to all amino acids a N (nitrogen) atom, a C_α (central carbon) atom, and a (carbonyl) carbon atom. (a) The composition of a generic amino acid with its main-chain highlighted. Within the main chain the separate functional amine-, and carboxylic (acid)-groups are also highlighted. (b) A portion of the protein structure containing two bonded amino acid molecules. For each amino acid shown, their side chain atoms, bonded to the C_α atom, are also shown in transparency.

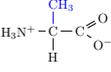
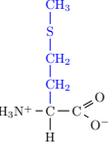
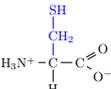
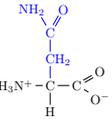
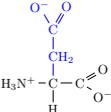
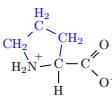
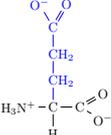
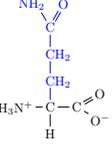
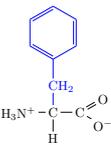
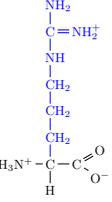
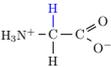
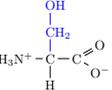
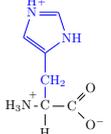
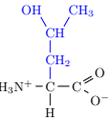
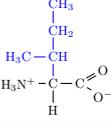
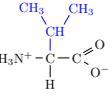
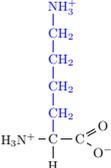
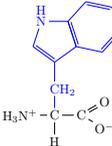
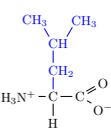
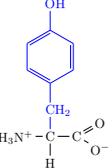
A protein is composed of one or more linear chains (*sequence*) of amino acids.[†] A single protein chain is typically several hundred residues long. The DNA (deoxyribonucleic acid) in each cell of an organism provides the blueprint containing instructions for the construction of its proteins (Crick, 1970). Once the DNA code is translated into a linear chain of amino acid residues, this chain spontaneously *folds*[‡] through a combination of physical and chemical processes, into a precise three-dimensional (3D) shape (also known as *structure*, or *conformation*) (Anfinsen, 1973; Hartl, 1996; Ellis, 2006). The 3D structure of a protein facilitates its biological function (Scheeff and Fink, 2005).

^{*}In protein chemistry, a residue refers to a specific monomer (amino acid) within the protein chain. This thesis will use the terms ‘residue’, ‘amino acid’, and ‘amino acid residue’ interchangeably.

[†]For a protein involving multiple chains of amino acids, each chain accounts for a part of the overall protein molecule.

[‡]Many factors affect protein folding including the environment and the presence of molecular “chaperones” that guide the protein to the correct conformation.

Table 2.1: Amino acids with their abbreviated names and chemical structures. Amino acid side chains appear in blue.

Amino acid	3-letter code	1-letter code	Chemical Structure	Amino acid	3-letter code	1-letter code	Chemical Structure
Alanine	Ala	A		Methionine	Met	M	
Cysteine	Cys	C		Asparagine	Asn	N	
Aspartic Acid	Asp	D		Proline	Pro	P	
Glutamic Acid	Glu	E		Glutamine	Gln	Q	
Phenylalanine	Phe	F		Arginine	Arg	R	
Glycine	Gly	G		Serine	Ser	S	
Histidine	His	H		Threonine	Thr	T	
Isoleucine	Ile	I		Valine	Val	V	
Lysine	Lys	K		Tryptophan	Trp	W	
Leucine	Leu	L		Tyrosine	Tyr	Y	

2.1.1 Conceptual Hierarchy of Protein Structures

Traditionally, the structure of proteins is considered within a conceptual hierarchy of representations: primary, secondary, tertiary and quaternary structure (Linderstrøm-Lang, 1952). The linear chain of amino acids can be represented as a *sequence* of one-letter codes (see Table 2.1). This sequence of one-letter codes is referred to as the *primary structure* of a protein, the first level of this hierarchy. See Figure 2.2(a) for an example of the primary structure. In this thesis, the terms sequence and primary structure are used interchangeably to mean the linear sequence of amino acids.

The second level of the hierarchy consists of the representation of the 3D structure of proteins at the level of standard *secondary structures* consisting of helices and strands of sheet[§] (Pauling and Corey, 1951; Pauling et al., 1951). Secondary structures form local structural segments within the global 3D structure of a protein. Secondary structures arise due to periodic hydrogen bonding patterns between pairs of amino acids, involving amine and carboxylic groups (Kabsch and Sander, 1983; Andersen and Rost, 2009). An example of a protein chain, decomposed into a representation of the local secondary structures elements appears in Figure 2.2(b): helices are coloured in red, while strands are abstracted as yellow arrows pointing towards the C-terminus of the secondary structural region. The green linkers are the remaining (unstructured) parts of the protein chain, termed here as *coils*.

Tertiary structure is the third level of the conceptual hierarchy, where a protein is represented using the details of its 3D atomic coordinates. While structures represented at this level appear very complex, the 3D shape of a tertiary structure is rather simple when considering only its backbone. The *backbone* of any protein refers to the sequence of the following main-chain atoms: $N-C_{\alpha}-C-N-C_{\alpha}-C-N-C_{\alpha}-C-\dots-N-C_{\alpha}-C$. The backbone begins with a nitrogen atom (called the amine- or *N-terminal* of the protein) and ends with a (carbonyl) carbon atom (called the carboxy- or *C-terminal* of the protein). Figure 2.2(c) shows an example of tertiary structure overlaid on its secondary structure.

The fourth and final level of the hierarchy refers to *quaternary structures*, which contains a number of tertiary structures (folded into protein subunits) assembled into a multi-subunit complex. An example of quaternary structure is shown in Figure 2.2(d), where each subunit is coloured differently.

2.1.2 Protein Structure Databases and File Formats

Structure databases underpin almost all research efforts in computational protein structural biology. Publicly available databases collect experimentally determined 3D structures of proteins deposited by protein biochemists and crystallographers from the world over. The central global repository for protein structural data is the Worldwide Protein Data Bank (wwPDB; Berman et al. (2003); <http://www.wwpdb.org>). The entire wwPDB database is freely accessible via the websites of partner organizations, spread across the globe, mainly: The Research Collaboratory for Structural Bioinformatics (RCSB; Berman et al. (2000); <http://www.rcsb.org>, located in the USA), PDBe (Velankar et al. (2011); <http://www.pdbe.org>, a european resource), and PDBj (Kinjo et al. (2012); <http://www.pdbj.org>, from Japan). As of the 10th of May 2016, the wwPDB contains entries for 110,135 protein structures. A breakdown of the proportions of protein structures determined using various experimental methods can be found at the URL: <http://www.rcsb.org/pdb/statistics/holdings.do>.

[§]A sheet is formed from several interconnected (anti-)parallel strands that appear as an, often twisted, pleated sheet.

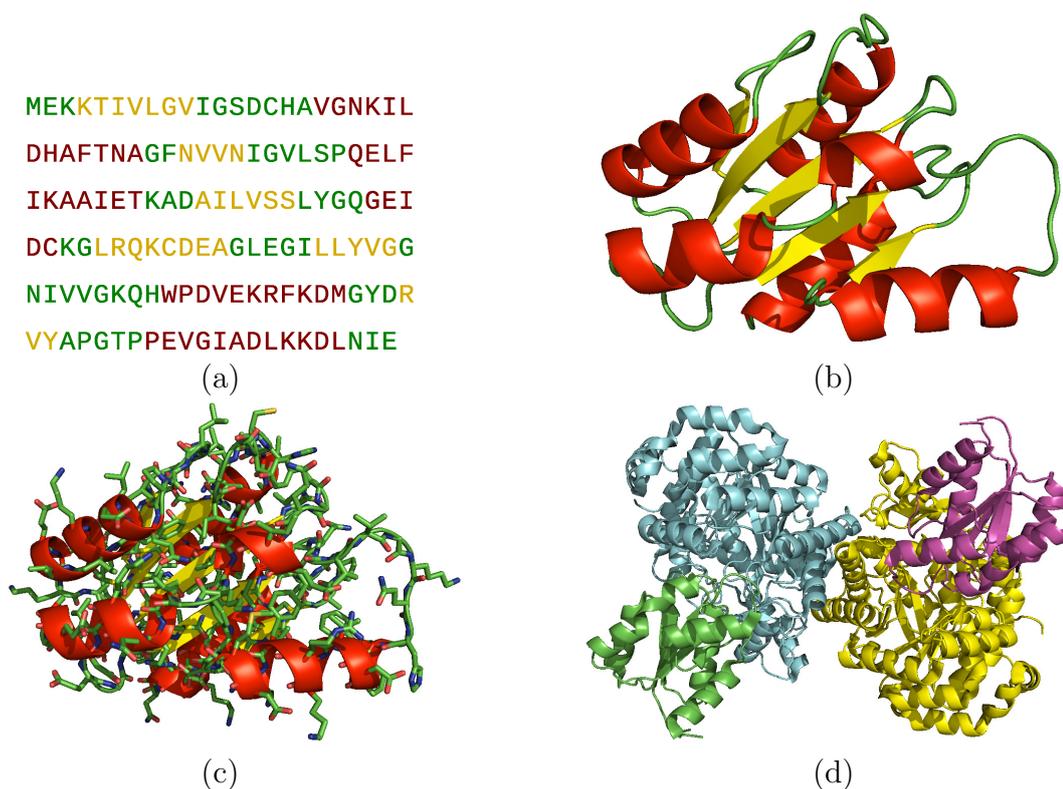


Figure 2.2: A protein structure (wwPDB 1CCW) represented by: (a) primary structure, (b) secondary structure, (c) tertiary structure, and (d) quaternary structure. Primary sequence one-letter codes are coloured according to the secondary structural element they belong to. Secondary helical regions are coloured in red, strand regions are coloured in yellow, and unstructured or coil regions are shown in green. The tertiary structure, overlaid by the secondary structural representation, is shown as a trace of the main-chain and side-chain atoms represented by “sticks”. The quaternary structure is shown using the secondary structure representation for each of its four subunits, where each subunit is coloured differently.

The data for each structure in the wwPDB contains atomic-level 3D (x, y, z) coordinate information, including the details of the experimental process used to determine the structure. Coordinate data from the wwPDB is available in several computer readable formats, including the comprehensive macro-molecular Crystallographic Information File (mmCIF; Bourne et al. (1997)) format from which the other formats are converted, the Brookhaven PDB format (for an example, see Figure 2.3(a); Bernstein et al. (1977)), and the PDBML (an XML format, an example is shown in see Figure 2.3(b); Westbrook et al. (2005)) (Westbrook and Fitzgerald, 2009). Coordinate data is presented in Angstrom (\AA) units, where 1\AA is equal to 1×10^{-10} m.

Every structural entry in the wwPDB has a 4-character unique identifier that is used to retrieve its coordinates. These identifiers are often used throughout the scientific literature (and this thesis) to refer to specific protein structures. For example the structure displayed in Figure 2.2 has the wwPDB identifier: 1CCW. Since a protein may contain multiple chains, each chain is given a single letter identifier (or chain identifier) usually starting from the uppercase letter ‘A’. For example, the protein chain that appears in Figure 2.2(c) is the first chain, with the chain identifier of A, from the structure of coenzyme B12 dependant Glutamate mutase from *Clostridium Cochlearium* (wwPDB 1CCW), which is made up for three other chains B, C, and D. This thesis uses the shorthand notation, wwPDB 1CCW-A to refer to coordinates of chain ‘A’ within the wwPDB 1CCW protein structure coordinate file.

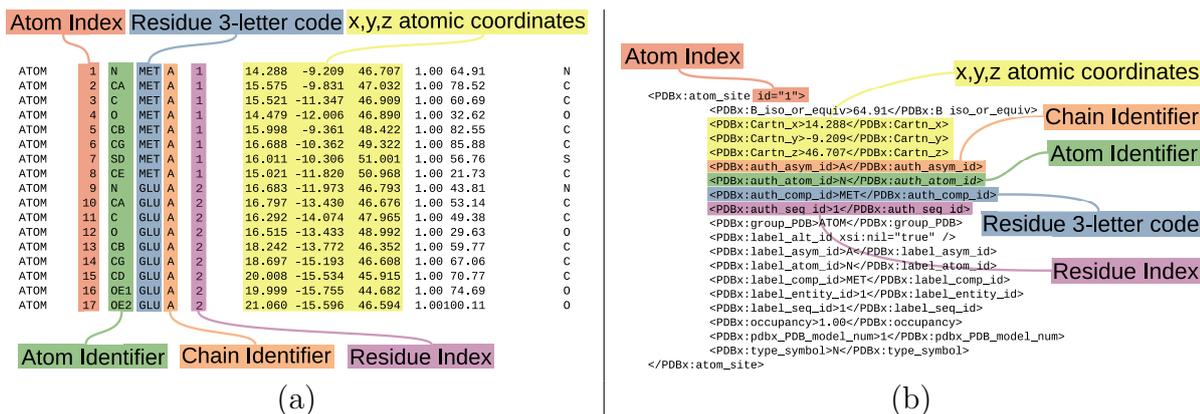


Figure 2.3: An extract of coordinate data from (a) a Brookhaven PDB coordinate data file containing two residues: a Methionine followed by a Glutamic Acid. (b) a PDBML file defining the coordinate data for a single nitrogen atom (the first in this structure) from a Methionine residue. Areas containing important information are highlighted. In red is the index for each atom, the atom identifier is in green: C_α C_β and so forth. The blue area contains the three-letter residue code from Table 2.1 above. This is followed by the chain identifier (in orange) and the residue index (in purple). The yellow area contains the x, y, z coordinates for the atoms, respectively.

2.1.3 Structure Determination and Prediction

The oldest and most common method used to experimentally determine the structure of proteins is X-ray crystallography (Kendrew et al., 1958; Muirhead and Perutz, 1963). Nuclear Magnetic Resonance (NMR; Wagner and Wüthrich (1978)) and Low-temperature Electron Microscopy (Adrian et al., 1984) are other techniques that can be used to resolve the coordinates of protein structures. These techniques are briefly described here along with a brief description of computational methods for predicting the structure of a protein from its sequence.

X-ray Crystallography. This technique begins by solidifying a sample of the protein into a crystal (Kendrew et al., 1958; Muirhead and Perutz, 1963). The crystal is then placed into a monochromatic X-ray beam (*e.g.*, from a synchrotron). As the beam passes through the crystal some of the X-rays are diffracted. The resulting diffraction pattern is recorded as the crystal is rotated in the X-ray beam. These X-ray diffraction patterns are then used to compute the electron densities of the atoms in the crystal. The primary structure and the electron densities are combined to produce a model of the tertiary structure. This model is then refined for several iterations to improve the quality (Lesk, 2010). X-ray crystallography most accurately resolves the position of individual atoms in the protein structure. However, the disadvantage of this method is that only a sub-set of proteins are crystallisable. In this case one of the following two methods may be used.

Nuclear Magnetic Resonance Spectroscopy. This method involves placing a sample of the protein in a homogeneous magnetic field and recording resonance spectra from the protein as determined by the electrical properties of the individual atoms in the protein (Wagner and Wüthrich, 1978). This can be performed in multiple dimensions for large molecules such as proteins. The advantages of this method over X-ray crystallography are that larger structures can be resolved and no crystallisation is required. Also, NMR records the range of natural movements that occur within protein structures. However, NMR cannot deliver the resolution

that X-ray crystallography (Lesk, 2010).

Low-temperature Electron Microscopy. In this method, a sample of the protein is snap-frozen to cryogenic temperatures (Adrian et al., 1984). Images of the protein are then taken from varying orientations to build up a 3D model. This method has several advantages over both crystallography and NMR in that it can resolve large protein complexes. This method can be combined with crystallography to produce very high resolution structures.

Automatic or Computational Methods. Experimental determination of protein 3D structure is significantly more difficult and expensive as compared to the determination of protein primary structure (sequence) (Lesk, 2010). Given this, computational methods to predict protein 3D structure from sequence information alone is an active area of research in computational structural biology (Lesk, 2010). However, this somewhat of a holy grail and is therefore an active research area that has not converged on a reliable method for doing so. Methods for doing so can be broken into two major categories: *ab initio* (*de novo*) modelling and homology (template based) modelling.

Template free modelling describes a method of constructing a protein structure model from a sequence alone. These methods are based on fundamental physical and chemical principles of protein folding and sometimes statistical knowledge of protein folding patterns (Lesk, 2010). Some of the methods in this category are Rosetta@home (Simons et al., 1999) and I-TASSER (Roy et al., 2010).

Template based modelling assembles known structural templates found by matching the template sequence with the target sequence. This method is far less computationally complex than *ab initio* methods and has had great success in accurately predicting protein structures from sequences (Moult et al., 2014). Methods in this category include SWISS-MODEL (Schwede et al., 2003) and FoldX (Schymkowitz et al., 2005).

The testing of tertiary structure prediction algorithms began on a large scale in the Critical Assessment of Structure Prediction (CASP) (Moult et al., 2014) competition, which started in 1994, and later in the Critical Assessment of Fully-Automated Structure Prediction (Fischer et al., 1999) competition. These competitions are primarily interested in testing the state-of-the-art of algorithms to predict the structure of proteins from their amino acid sequences. Both categories of methods for structure prediction mentioned above are tested.

2.1.4 Recurrent Substructural Themes and Classification

The standard secondary structures (generally helices and strands) are observed to combine into substructural units. The amino acid chains of proteins often contain two or more compact substructural units that fold independently into stable 3D conformations. Such substructural units are referred to as *domains* and most proteins are composed of at least two domains (Vogel et al., 2004). A domain is an independent structural unit (Richardson, 1981) of protein function, evolution (Bork, 1991) and folding (Wetlaufer, 1973). The size of domains varies around an average of approximately 100 amino acids (Wheelan et al., 2000).

Since domains can be viewed as the evolutionary units of protein structure, classification is performed at the level of domains. Moreover, separate domains from multi-domain proteins are often similar to other whole proteins (Richardson, 1981). Protein domains are hierarchically classified into related groups. Two important databases for protein domain classification are SCOP (Murzin et al. (1995); Lo Conte et al. (2000); <http://scop.mrc-lmb.cam.ac.uk/scop>) and CATH (Orengo et al. (1997); <http://www.cathdb.info>). This thesis uses the updated

SCOPe (Fox et al. (2013); <https://scop.berkeley.edu>) database which builds on SCOP and adds new structures to the classification.

SCOP database (Murzin et al., 1995; Lo Conte et al., 2000). The Structural Classification Of Proteins database classifies protein domains according to a four-level hierarchical tree with each level given the name (in order, from the root of the tree, from furthest to closest in evolutionary terms): Class, Fold, Superfamily, Family. Domains within a Class share different types of folds, each with a similar type of secondary structure composition. For example a SCOP class may contain domains that only contain arrangements of helical secondary elements. Proteins in the same fold classification share similar secondary structure arrangement, order, and topology. Within a fold classification, domains are further classified into superfamilies. Domains which share a superfamily are likely to share an evolutionary relationship, but their relationship is distant and, hence, share little sequence similarity. An example superfamily may be “Globin-like”. Within the superfamily classification, closely related domains are classified within the same family. Proteins in the same family share a clear evolutionary relationship. An example is the “Globins” family. The human haemoglobin protein, wwPDB 1HH0-A, is a domain classified by SCOP as the following. Class: all helical proteins; fold: globin-like; superfamily: globin-like; and family: globins.

CATH database (Orengo et al., 1997). The Class, Architecture, Topology and Homologous superfamily database is broadly similar in its classification of protein domains to SCOP, however it differs greatly in the details. CATH uses a combination of the SSAP (Taylor and Orengo, 1989) alignment program and human curation to classify known proteins into a four level hierarchical tree: class (equivalent to SCOP class), architecture, topology (equivalent to SCOP fold), and homologous superfamily (equivalent to SCOP superfamily). Proteins in the same class share a similar secondary structure makeup (for example, one CATH class contains domains that have mostly helical secondary structures). Those in the same architecture have a similar arrangement of secondary structures, but no indication of homology (architecture classification is manually performed). Those in the same topology group share specific structural features, while those in the same homologous superfamily are clustered by sequence similarity. The example protein, wwPDB 1HH0-A, is a domain classified by CATH as the following. Class: mainly helical; architecture: orthogonal bundle; topology: globin-like; and superfamily: globins.

According to these two domain classification databases, there are between 1195 and 1375 protein folds at the time of writing (June 2016). Given the large number of known protein structures, it might be surprising that there are relatively few protein folds (or domains of distinct topology in the parlance of the CATH database). In fact, the number of folds was estimated to be approximately 1000 to 1500 by Chothia (1992), or up to several thousand by Orengo et al. (1994). This relatively small number is likely due to the functional constraints that evolution places on protein structures. Proteins of diverse evolutionary origin tend to converge, in parallel, on similar folds according to their function, or diverge from a common ancestor maintaining the same fold through selective pressure to retain function (Richardson, 1981; Chothia and Lesk, 1986; Murzin, 1998; Edwards and Deane, 2015).

At a finer level, regularity also exists at the smaller scale when combining a few secondary structure elements. These common combinations are known as supersecondary motifs. Some examples of common supersecondary motifs are shown in Figure 2.4. Figure 2.4(a) shows what is known as a β -hairpin, a pair of anti-parallel strands connected by a small coil. Figure 2.4(b) shows a helix-turn-helix motif, which is a pair of helices connected by a partial helical turn.

Figure 2.4(c) shows a related motif known as an EF hand a pair of helices connected by a small coil region. The loop is usually 12 residues long. Finally, Figure 2.4(d) shows a helical motif, this is a series of strands formed into a helical pattern with, in this case, three faces (but these motifs may only have two faces). These are examples of some common supersecondary motifs. There are, however, many more and finding these motifs is an active area of research (Unger et al., 1989; Rooman et al., 1990; Camproux et al., 1999; Micheletti et al., 2000; Kolodny et al., 2002; Friedberg and Godzik, 2005; Joseph et al., 2010; Konagurthu et al., 2013; de Oliveira et al., 2015).

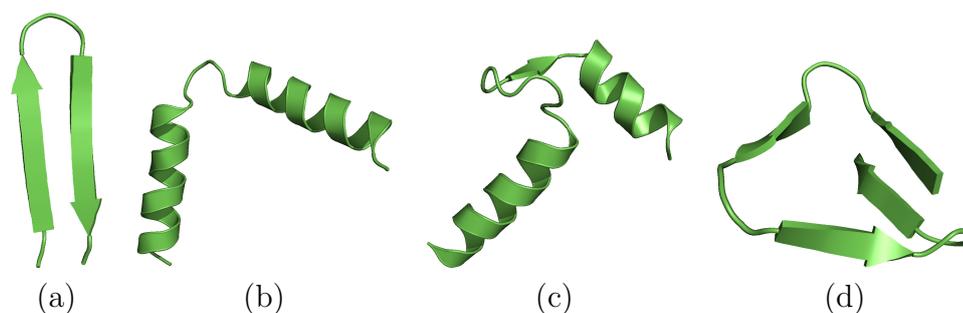


Figure 2.4: Some examples of common supersecondary motifs. (a) A Beta hairpin motif. (b) A helix-turn-helix motif. (c) An EF hand motif. (d) A beta helix motif.

2.1.5 Geometric Constraints on the Protein Backbone Atoms

The topology and geometry of protein structure is subject to physical and chemical constraints. The average bond lengths between atoms within amino acids of proteins was accurately estimated by Marsh and Donohue (1967). They are, N-C $_{\alpha}$: 1.46Å, C $_{\alpha}$ -C: 1.51Å, and C-N: 1.33Å. These lengths are tightly constrained by the nature of the covalent bonds. The geometric relationship between successive main-chain atoms can be defined using these bond lengths, together with the backbone dihedral angles (or torsion angles, see Figure 2.5) between successive residues: ω (torsion about the bond between carbonyl C and amine N), ϕ (torsion about the bond between N and central C $_{\alpha}$) and ψ (torsion about the bond between C $_{\alpha}$ and C). The dihedral angle ω is often very close to $\pm 180^{\circ}$ with the associated nitrogen bonded with a hydrogen (from the amine group) and carbon double bonded with an oxygen (from the carbonyl group) all lying in the same plane. Variation of the other two dihedral angles, ϕ and ψ cause proteins to exhibit the observed variability in their 3D structures (Richardson, 1981).

Figure 2.6 shows the Ramachandran-Ramakrishnan-Sasisekharan plot (Ramachandran et al., 1963) containing the frequency distribution of the dihedral angles ϕ and ψ . In general, the local geometry that standard secondary structure elements (helices and strands of sheet) take, result in the characteristic densities in this plot. Strands of sheet appear in the top-left quadrant, and the most common types of helices appear in the bottom-left quadrant. Furthermore, there are areas of the Ramachandran-Ramakrishnan-Sasisekharan plot, for example in large portion of the bottom-right quadrant, which are forbidden and may indicate a poorly solved protein structure.

It must be further noted that the planar nature of the ω dihedral angle, about the bond connecting successive amino acids, imposes a strict constraint on the distance between successive C $_{\alpha}$ atoms along the protein backbone. Successive C $_{\alpha}$ -C $_{\alpha}$ distances has a mean of 3.8Å, deviating rather tightly (often with a standard deviation of 0.2Å) about this mean value. **This**

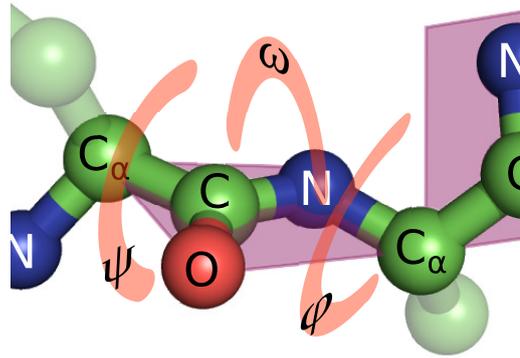


Figure 2.5: A section of protein main-chain showing the three backbone torsion (or dihedral) angles. These angles describe the range of motion for the backbone. The bond between the N and C α atoms can be rotated, this rotation is described by the torsion angle ϕ . The rotation of the bond between C α and C atoms is called the ψ torsion angle. The ω torsion angle between C and N is very limited, making the peptide chain planar between C α atoms (shown with purple planes between C α atoms).

is an important fact that will be used extensively throughout the remainder of this thesis.

2.1.6 Topology of Proteins and Structural Plasticity

The approximate relative positions, orientations and the sequential ordering of secondary structural elements is described by their topology (Levitt and Chothia, 1976; Westhead et al., 1999). This is a simplified description of the protein fold that can be described in a diagram known as a Topology of Protein Structure (TOPS) cartoon. An example of a TOPS cartoon appears in Figure 2.7.

Proteins are not static objects, but are often flexible and prone to plastic deformation and domain motion (Bu and Callaway, 2011). Plasticity refers to the ability of a protein to flex, deviate, and adopt alternate conformations. These conformational changes have a variety of functions which include catalysis, transport, formation of complexes and motion (Gerstein et al., 1994). These changes often occur through the interaction with other molecules known as *ligands* (Frauenfelder et al., 1991). Often these molecules attach (or *bind*) to the protein at a specific place, known as a *binding site*, allowing the protein to accomplish a function. For example, the human haemoglobin protein (wwPDB 1HHO) binds a heme ligand with attached oxygen to transport within the blood stream. These binding sites are the functionally important parts of the protein, the functionally unimportant, or less important parts of the protein are somewhat free to change their conformation. Alternatively, conformational changes can also occur through sequence mutations or the folding environment (Meier and Özbek, 2007). Closely related proteins can appear to deform with the insertion or deletion of only a few residues (Vetter et al., 1996; Simm et al., 2007). All of these conformational changes are constrained to allowed regions in the Ramachandran-Ramakrishnan-Sasisekharan plot, as described above and shown in Figure 2.6.

Secondary structural elements are stable relative to the joining coil regions. These coils are often small and conformational changes within them do not represent a large change to the overall shape of the protein. Larger conformational changes occur between domains in the form of hinge or sheer motions (Gerstein et al., 1994). Gerstein et al. (1994) provide many examples for each of these types of domain motions. Various methods exist to determine the

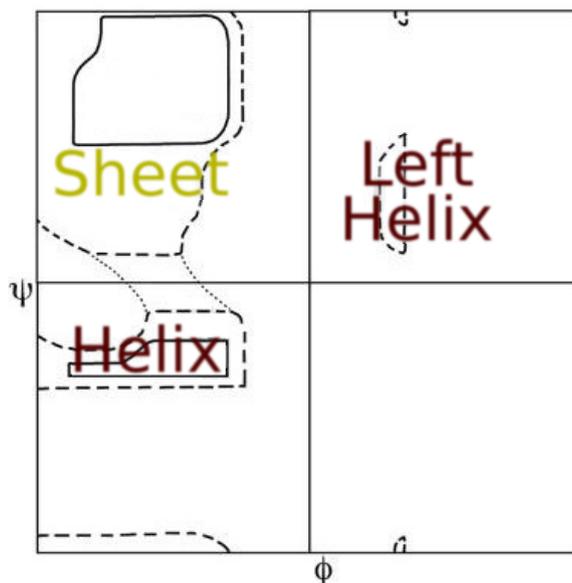


Figure 2.6: Ramachandran-Ramakrishnan-Sasisekharan density plot showing the distribution of combinations of ϕ , ψ dihedral angles. The boundaries appear around areas generally enclosing secondary structural geometry: helices, sheets and left-handed helices. ϕ is on the x -axis and ψ is on the y -axis. (Ramachandran and Sasisekharan, 1968)

locations and to characterise the movement mechanism for domain motion. These methods include HingePROT (Emekli et al., 2008) and DynDom (Hayward and Berendsen, 1998).

Point Hinge motions are the result of large changes in backbone torsion angles in a localised region (Gerstein et al., 1994). This may occur in an extended strand secondary structure (Gerstein and Chothia, 1991), a section of coil, or (to a limited extent due to stricter chemical constraints) in helical secondary structures (Gerstein and Chothia, 1991). These types of motions often happen at the terminals of secondary structural elements (Hayward, 1999). Hinge motions also take the form of planar hinges which are the result of hinge motion of interconnected strands of sheet moving together like the folding of a sheet of paper. The sheet remains packed together in any conformation of the hinged domains. An example of this kind of hinge motion can be observed in Figure 2.8 and is described by (Gerstein et al., 1993). Finally, sheer motions involve a sliding movement between domains. This type of motion does not cause the main chain to deform very much and are controlled by side-chain motion (Gerstein et al., 1994).

2.2 Alignment of Proteins

An alignment (Rao and Rossmann, 1973) of proteins refers to the assignment of one-to-one, (commonly) order-preserving correspondence (or *equivalence*) between a *subset* of their amino acids. Alignments can be computed between two or more proteins. This thesis specifically deals with *pairwise* alignments, or alignments between two proteins. Throughout this thesis the symbols S and T are used to denote the two proteins being aligned. Specifically, S and T are (linearly) ordered sets containing respective amino acid (atomic coordinates of the central carbon, C_α , atoms[¶]), in the order they appear from N- to C-terminal of each protein.

[¶]This description of protein backbone at the level of C_α is common in the field, as the 3D trace of C_α atoms is sufficient to distinguish the structural similarities and differences between proteins. More on protein structural representations can be found in Section 2.2.4.

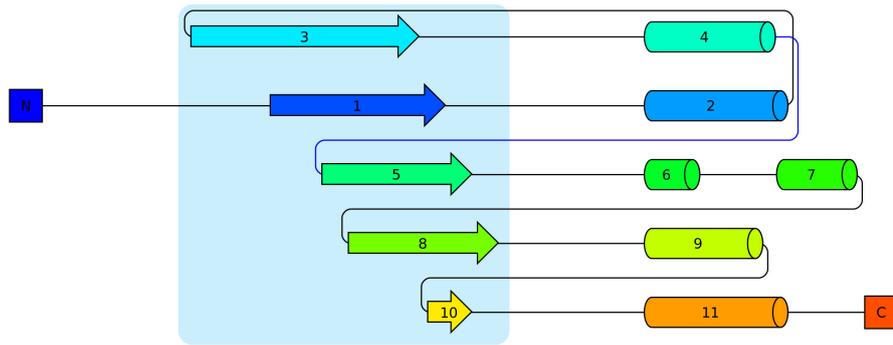


Figure 2.7: A Topology diagram for the protein structure wwPDB 1CCW, chain A. Helices are represented by cylinders, and strands are represented by arrows. The length is proportional to their length in the structure. The N terminus of the protein is labelled with an N in a dark blue box, and the C terminus is labelled with a C in a red box. The connected sheet of anti-parallel strands is highlighted by a blue background. This image was produced by the pro-origami software (Stivala et al., 2011).

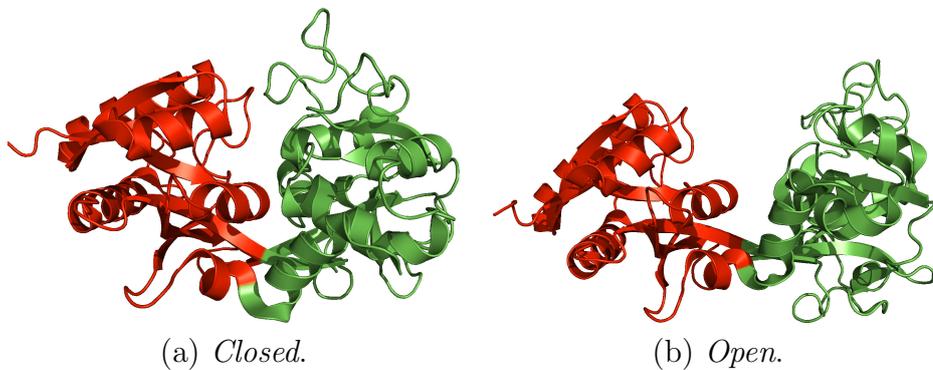


Figure 2.8: An example of a hinge rotation in the iron binding protein lactoferrin. The N-terminal of this protein contains two domains, marked in red and green. The domains *close* together when iron is bound, as in (a). The domains *open* apart when iron is not bound, as in (b). The individual domains are rigid in each conformation. The closed conformation between domains in (a) is given in wwPDB 1LFG. The open conformation between domains in (b) is given in wwPDB 1LFH.

Thus, any pairwise (order-preserving) alignment can be represented as a $2 \times n$ matrix, where the rows represent the two proteins being aligned and the columns[‡] represent the correspondence (or lack thereof) between amino acids of the two proteins. An example of an alignment representation between two proteins appears in Figure 2.9(a).

A column of any alignment has one of the following three states:

Match: defines a one-to-one correspondence between a pair of amino acids, one from each protein. These columns appear without the gap symbol (‘-’).

Delete: defines the absence of a correspondence for an amino acid residue in S . Such columns are represented with the gap symbol in the second row.

[‡]If $|S|$ and $|T|$ denote the sizes of the two proteins, then the number of columns is always in the range $\max(|S|, |T|) \leq n \leq |S| + |T|$.

Insert: defines the absence of a correspondence for an amino acid residue in T . Such columns are represented with the gap symbol in the first row.

These state definitions will be used throughout this thesis.

2.2.1 Protein Structural Alignment

Alignments are often used to infer *homology*, that is, an evolutionary relationship between proteins (Eidhammer et al., 2000). As organisms evolve (and diverge) from their common ancestor, so do their protein domains (Vogel et al., 2004). Therefore, an alignment is often an attempt to identify (and pair-up) amino acids that are derived from the same position in the genetic sequence of a common ancestor.

An alignment between proteins can be computed based on different sources of information:

Sequence Alignment: The assignment or residue-residue correspondences is made primarily using the information from the type and physico-chemical characteristics of the amino acids as part of the protein chain.

Structural Alignment: The assignment of residue-residue correspondences is made primarily using the atomic 3D coordinates of the amino acids in the protein chain (See Section 2.2.4).

When aligning proteins that share a very close evolutionary relationship, both sequence and structural alignments are often consistent with each other. However, sequence and structural alignments radically differ as the relationship between two proteins diverge. As noted earlier in Section 2.1.4, protein structures are directly constrained by selective pressure to retain their structure, but are freer to accrue changes to their sequence. Therefore, structures change more conservatively than their sequences in evolution (Chothia and Lesk, 1986; Abroi and Gough, 2011; Illergård et al., 2009). Most homologous proteins preserve a *common core* within their 3D structure (Chothia and Lesk, 1986). This core comprises one or more regions of residues that retain the same topology of their folding patterns, varying only in their spatial and geometric details (Konagurthu et al., 2006). Peripheral regions outside the common core tend to refold entirely and cannot be meaningfully aligned. Although pairwise **sequence** alignment cannot identify such regions, pairwise **structural** alignment can. Thus, structural alignments can reveal relationships that are otherwise invisible when only aligning sequences (Perutz et al., 1965; Lesk, 2000) and, therefore, offer a more accurate picture of homology, even for very distantly related proteins (Chothia and Lesk, 1986).

2.2.2 Formulation of the Structural Alignment Problem

Given the atomic coordinates of two protein structures S and T , the structural alignment problem involves the following separate tasks:

The alignment task: As stated above, (Section 2.2.1) a structural alignment is the assignment of one-to-one correspondence between amino acids, using the information from their 3D atomic coordinates. This can be posed as a combinatorial optimisation problem requiring an *objective function* (a measure of structural alignment quality) and a *search method* to find an optimal alignment under the stated quality measure. Current techniques for formulating an objective function are reviewed below (Section 2.2.5) as well as being discussed in detail in Section 4.2. A review of alignment methods appears in Section 6.2.

```

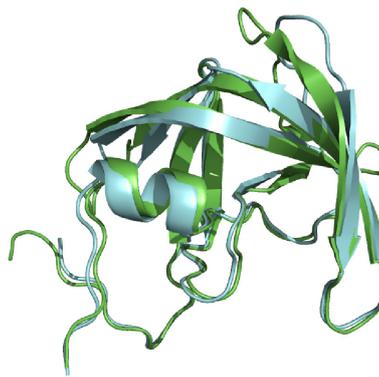
VGTTTTLEKRPEILIFVNGYPIKFLLDTGADITILNRR
--PQITLWQRPLVTIKIGGQLKEALLDTGADDTVLEEM

DFQVKNSIENGRQNMIGVGGGKRGTNYINVHLEIRDEN
SLP-----GRWKPKMIGGIGGFIKVRQYDQILIEICG--

YKTQCIFGNVCVLEDNSLIQPLLGRDNMIKFNIRLVM
---HKAIGTVLVGP---TPVNIIGRNLLTQIGCTLNF

```

(a)



(b)

Figure 2.9: A pairwise protein structural alignment: (a) Structure based sequence alignment of Feline Immunodeficiency Virus Protease (green) and Human Immunodeficiency Virus Protease (blue). (b) Optimal structural superposition given the alignment in (a). The Feline Immunodeficiency Virus Protease protein structure appears again in green, and the Human Immunodeficiency Virus Protease protein structure in blue.

The superposition task: Once the residue-residue correspondence has been established by the alignment task, the coordinates of two proteins are spatially transformed such that the atomic coordinates of equivalent amino acid residues come as close as possible. This allows the structural similarity, as defined by a given structural alignment, to be understood visually and quantitatively. An illustration of a superposition, given an alignment, is shown in Figure 2.9(b). The analytical aspects of the superposition task are discussed in detail in Section 2.2.3 below.

2.2.3 Protein Structural Superposition

As mentioned above, the superposition of protein structures provides a visual and numerical means to evaluate the similarity and differences between protein structures (see Figure 2.9(b)) (Cohen and Sternberg, 1980). The numerical measure provides a descriptive statistic of the distances between corresponding amino acids.

Each atomic coordinate of any protein is a *vector* (or point) in Euclidean space, \mathbb{R}^3 . Therefore, once the pairwise alignment task is completed, the corresponding amino acid residues can be collected into two vector sets with each vector in a set corresponding to its equivalent in the other set. The optimal superposition (under some pre-defined definition of optimality) of these two vector sets poses yet another optimisation problem. A near universal *criterion of optimality* for the superposition problem is the minimisation of the *root mean square deviation* (RMSD) between corresponding vector sets. This criterion arises from the least squares superposition problem, which is solved by finding orthogonal, linear transformations (*rotation* and *translation*) of the vector sets that minimise the squared error between the corresponding vectors in each vector set (Lesk, 2000, 2001b; Eidhammer et al., 2004).

The least squares superposition problem can be formally stated as follows. Given two sets of N_e corresponding vectors in \mathbb{R}^3 :

$$\mathbf{U} = \{\vec{u}_1, \vec{u}_2, \dots, \vec{u}_{N_e}\} \text{ and } \mathbf{V} = \{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_{N_e}\}$$

where each \vec{u}_i is in correspondence with \vec{v}_i between the two sets, \mathbf{U} and \mathbf{V} . Thus, the problem is to find the best orthogonal rotation (representable as a real square symmetric matrix $\mathbf{R}_{3 \times 3}$

with a determinant of +1) and translation (representable as a vector, $\vec{t}_{3 \times 1}$) which minimises the squared error of superimposing \mathbf{U} onto \mathbf{V} :

$$\xi = \min \sum_{i=1}^{N_e} \|\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i\|^2 \quad (2.1)$$

where $\|\vec{x}\|$ is the \mathcal{L}^2 -norm of the vector \vec{x} .

The solution to the optimal translation can be made independently of the solution to the optimal rotation (Alt et al., 1988). This can be found by differentiating ξ with respect to \vec{t} and evaluating it at its extremum:

$$\begin{aligned} \frac{\partial \xi}{\partial \vec{t}} &= \frac{\partial}{\partial \vec{t}} \sum_{i=1}^{N_e} \|\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i\|^2 = 2 \sum_{i=1}^{N_e} \left(\frac{\partial(\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i)}{\partial \vec{t}} (\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i) \right) = 0 \\ \Rightarrow \sum_{i=1}^{N_e} (\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i) &= 0 \Rightarrow \mathbf{R} \sum_{i=1}^{N_e} \vec{u}_i + N_e \vec{t} - \sum_{i=1}^{N_e} \vec{v}_i = 0 \\ \Rightarrow \vec{t} &= \underbrace{\frac{1}{N_e} \sum_{i=1}^{N_e} \vec{v}_i}_{\mathbf{V} \text{ centre-of-mass}} - \mathbf{R} \underbrace{\frac{1}{N_e} \sum_{i=1}^{N_e} \vec{u}_i}_{\mathbf{U} \text{ centre-of-mass}} \end{aligned}$$

Since the optimal transformation involves the optimal translation of the centres-of-mass of the two vector sets, computing the optimal translation can be separated from the computation of the optimal rotation by translating the two vector sets, such that their centres-of-mass coincide with the origin: $\vec{u}'_i = \vec{u}_i - \frac{1}{N_e} \sum_{j=1}^{N_e} \vec{u}_j$, $\vec{v}'_i = \vec{v}_i - \frac{1}{N_e} \sum_{j=1}^{N_e} \vec{v}_j \forall 1 \leq i \leq N_e$. The vectors sets \mathbf{U} and \mathbf{V} can thus be transformed to become: $\mathbf{U}' = \{\vec{u}'_1, \vec{u}'_2, \dots, \vec{u}'_{N_e}\}$ and $\mathbf{V}' = \{\vec{v}'_1, \vec{v}'_2, \dots, \vec{v}'_{N_e}\}$.

It follows that removing the translation term from the previous objective function in Equation 2.1 yields a modified, but equivalent, objective function which is independent of the translation \vec{t} :

$$\xi = \min \sum_{i=1}^{N_e} \|\mathbf{R}\vec{u}'_i - \vec{v}'_i\|^2 \quad (2.2)$$

And the RMSD can thus be defined as:

$$RMSD(\mathbf{U}', \mathbf{V}') = \sqrt{\frac{1}{N_e} \sum_{i=1}^{N_e} \|\mathbf{R}\vec{u}'_i - \vec{v}'_i\|^2} \quad (2.3)$$

Remarkably, this optimisation problem can be both solved exactly and efficiently, and involves a computational effort that grows *linearly* with the number of correspondences in the vector sets, $O(N_e)$. Methods for calculating a superposition of two equal size vector sets have improved over time. The most important of these methods are briefly summarised below.

McLachlan (1972). This procedure assigns a weight to each atom after translating the centre-of-mass of the query structure and the reference to the origin of the coordinate system. The problem is reduced to finding a rotation matrix such that a similarity function is minimised. McLachlan presents two methods (McLachlan, 1972, 1982) to find the rotation matrix. One is an iterative method derived from the Jacobi matrix diagonalisation algorithm, the other is an analytical solution. Both algorithms fail to eliminate potential matches that are clearly

outside a given threshold. Further, the analytical solution contains many special-cases where the approach fails.

Kabsch (1970s). Later, Kabsch (1976, 1978) presents a purely analytical method to solve the superposition problem in matrix form. The solution involves finding a symmetric matrix of Lagrange multipliers to find an orthogonal matrix, \mathbf{U} , that minimises the metric function: $E = \frac{1}{2} \sum_n (U\vec{u}_n - \vec{v}_n)^2$. Eigenvalues are extracted using singular value decomposition and the minimised value for the similarity metric is found. This method does have some exceptions, such as a zero eigenvalue when all vectors in \vec{u}_n and \vec{v}_n are in the same plane. Kabsch does however present a solution for this special case.

Lesk (1986). This algorithm finds the optimal rotation matrix as a function of 4 parameters, where the axis of rotation is defined explicitly. It is noted that this, along with the explicit rotation, eliminates the possibility of inverting the orientation of the coordinate system, resulting in a sub-optimal RMSD value. Rustici and Lesk (1994) build on the ideas presented above by introducing the concept of a lower bound on the RMSD. Calculating a lower bound on the RMSD allows the algorithm to avoid a calculation of the optimal superposition, where the lower bound is larger than the threshold for any particular comparison. They note that their results are good but they expect the algorithm to do better.

Daimond (1988). This is a method similar to that of Lesk (1986). Only half the required angle of rotation is required for the parameterisation of the rotation matrix. The problem becomes that of a 4×4 eigenproblem or an inversion of a 3×3 matrix. This method is a slight improvement on that of Lesk (1986), however it becomes unstable as the angle of rotation approaches π , where an inverse is not defined because the matrix becomes singular (Theobald, 2005).

Kearsley (1989). A very elegant approach to the superposition problem was proposed by Kearsley (1989), which solves the least squares superposition problem as an eigenvalue problem in quaternion parameters, using quaternion algebra (Hamilton, 1844). This approach builds a 4×4 symmetric matrix (from quaternion parameters) and finds the smallest eigenvalue, $\min(\lambda)$ in this matrix, which is related to the RMSD by $\sqrt{\frac{\min(\lambda)}{N_e}}$. This has a number of advantages over the algorithms provided by McLachlan (1972) and Kabsch (1976, 1978). Namely, there are no special cases for the method and it does not suffer from a degeneracy of the axes of rotation for some orientations, known as “gimbal lock” (Kearsley, 1989).

Note that this method is used extensively for any superposition **throughout this thesis**. Provided below is the analytical solution of the least-squares superposition problem using Kearsley’s method (Kearsley, 1989). While understanding the details of the derivation is not necessary to understand the rest of this thesis, it is provided below as a reference for interested readers. This solution informs the derivation of sufficient statistics (Konagurthu et al., 2014) for this problem, which is explained in Section 7.2 of this thesis. Further, it serves as a pillar for the efficient computation of structural alignments using **MMLigner** (Chapter 6) and *I*-value (Chapter 5) as outlined in this thesis.

This derivation begins with some preliminaries on quaternion algebra (Hamilton, 1844; Karney, 2007), which provides a powerful method to describe rigid-body rotations in \mathbb{R}^3 (Mackay,

1984). The set of quaternions is equivalent to a four dimensional vector space over the real numbers. A quaternion, \mathbf{q} , consists of four values: a scalar part, q_1 , and a vector part: $\langle q_2, q_3, q_4 \rangle = \vec{q}$: $\mathbf{q} = (q_1, q_2, q_3, q_4) = (q_1, \vec{q})$. Each quaternion \mathbf{q} has a conjugate: $\bar{\mathbf{q}} = (q_1, -\vec{q})$.

Addition between quaternions, \mathbf{p} and \mathbf{q} is defined as $\mathbf{p} + \mathbf{q} = (p_1 + q_1, \vec{p} + \vec{q})$. Quaternion *multiplication* is associative but not commutative and is defined for the quaternions, \mathbf{p} and \mathbf{q} as: $\mathbf{p}\mathbf{q} = (p_1q_1 - \vec{p} \cdot \vec{q}, p_1\vec{q} + q_1\vec{p} + \vec{p} \times \vec{q})$. The *quaternion-norm* is defined as: $\|\mathbf{q}\| = \sqrt{\mathbf{q}\bar{\mathbf{q}}}$. The *inverse*, \mathbf{q}^{-1} is: $\mathbf{q}^{-1} = \frac{(q_1, -\vec{q})}{\|\mathbf{q}\|^2}$. And if \mathbf{q} is a *unit quaternion*: $\mathbf{q}^{-1} = \bar{\mathbf{q}}$.

Any 3D rotation can be defined by a unit quaternion, given a unit vector representing an axis of rotation, \hat{a} , and a rotation angle, θ , the rotation can be defined as: $\hat{\mathbf{q}} = (\cos(\theta/2), \hat{a} \sin(\theta/2))$. The rotation of a vector, \vec{v} , to yield a rotated vector, \vec{v}' is achieved, using quaternion algebra, by: $\vec{v}' = \hat{\mathbf{q}}^{-1}(0, \vec{v})\hat{\mathbf{q}}$.

Recall from Equation 2.2 that the function to be minimised is the least squares error between N_e corresponding points from the vector sets, \mathbf{U}' and \mathbf{V}' . A quaternion is used to represent the *residual* vectors, or individual error vectors ($\vec{e}_i = \mathbf{R}\vec{u}'_i - \vec{v}'_i$) between coordinates in \mathbf{U}' and \mathbf{V}' rotated by a quaternion, $\hat{\mathbf{q}}$ in \mathbb{R}^3 , $\forall i \in (1, \dots, N_e)$ as:

$$(0, \vec{e}_i) = (0, \vec{v}'_i) - \hat{\mathbf{q}}^{-1}(0, \vec{u}'_i)\hat{\mathbf{q}} \quad (2.4)$$

The problem, then, is to construct a $\hat{\mathbf{q}}$ that *minimises*, $\sum \|\vec{e}_i\|^2 = \sum \|(0, \vec{e}_i)\|^2$. This can be simplified by multiplying Equation 2.4 by $\hat{\mathbf{q}}$:

$$\hat{\mathbf{q}}(0, \vec{e}_i) = \hat{\mathbf{q}}(0, \vec{v}'_i) - (0, \vec{u}'_i)\hat{\mathbf{q}} \quad (2.5)$$

And using this to construct a modified error function:

$$\xi' = \sum_{i=1}^{N_e} \|\hat{\mathbf{q}}(0, \vec{e}_i)\|^2 \quad (2.6)$$

Expanding the modified error function gives: $\xi' = \sum_{i=1}^{N_e} \|(-\vec{q} \cdot \vec{e}_i, q_1\vec{e}_i + \vec{q} \times \vec{e}_i)\|^2 = \sum_{i=1}^{N_e} \|\mathbf{q}\|^2 \|\vec{e}_i\|^2 = \|\mathbf{q}\|^2 \sum_{i=1}^{N_e} \|\vec{e}_i\|^2$. When $\hat{\mathbf{q}}$ defines a *pure* rotation (no dilation), ξ' reduces to the unmodified error function since $\|\hat{\mathbf{q}}\| = 1$. Therefore, substituting Equation 2.5 into the expansion gives:

$$\xi = \sum_{i=1}^{N_e} \|\hat{\mathbf{q}}(0, \vec{v}'_i) - (0, \vec{u}'_i)\hat{\mathbf{q}}\|^2 \quad (2.7)$$

This can be expanded using the quaternion product as:

$$\begin{aligned} \xi &= \sum_{i=1}^{N_e} \|(-\vec{q} \cdot \vec{v}'_i, q_1\vec{v}'_i + \vec{q} \times \vec{v}'_i) - (-\vec{u}'_i \cdot \vec{q}, q_1\vec{u}'_i + \vec{u}'_i \times \vec{q})\|^2 \\ &= \sum_{i=1}^{N_e} \|(-\vec{q} \cdot \vec{v}'_i + \vec{u}'_i \cdot \vec{q}, q_1\vec{v}'_i + \vec{q} \times \vec{v}'_i - q_1 \cdot \vec{v}'_i - \vec{u}'_i \cdot \vec{q})\|^2 \\ &= \sum_{i=1}^{N_e} \|(-\vec{q} \cdot (\vec{v}'_i - \vec{u}'_i), q_1(\vec{v}'_i - \vec{u}'_i) + \vec{q}(\vec{v}'_i - \vec{u}'_i))\|^2 \end{aligned}$$

The squared norm of a quaternion, \mathbf{Q} , is:

$$\begin{aligned}\|\mathbf{Q}\|^2 &= \mathbf{Q}\bar{\mathbf{Q}} = (Q_1, \vec{Q})(Q_1, -\vec{Q}) \\ &= (Q_1^2 + \vec{Q} \cdot \vec{Q}, -Q_1\vec{Q} + Q_1\vec{Q} + \vec{Q} \times -\vec{Q}) \\ &= Q_1^2 + \vec{Q} \cdot \vec{Q}\end{aligned}$$

Therefore, after expanding the respective Cartesian components of \vec{q} , \vec{u} and \vec{v} , the error function becomes:

$$\begin{aligned}\xi &= \sum_{i=1}^{N_e} ((q_2(v_i^x - u_i^x) + q_3(v_i^y - u_i^y) + q_4(v_i^z - u_i^z))^2 \\ &\quad + (q_1(v_i^x - u_i^x) + q_3(v_i^z + u_i^z) - q_4(v_i^y + u_i^y))^2 \\ &\quad + (q_1(v_i^y - u_i^y) + q_4(v_i^x + u_i^x) - q_2(v_i^z + u_i^z))^2 \\ &\quad + (q_1(v_i^z - u_i^z) + q_2(v_i^y + u_i^y) - q_3(v_i^x + u_i^x))^2)\end{aligned}$$

To ensure \mathbf{q} defines a rotation only, $\|\mathbf{q}\|$ is constrained to 1 using the *Lagrangian* as follows:

$$\begin{aligned}\Lambda(\mathbf{q}, \lambda) &= \sum_{i=1}^{N_e} ((q_2(v_i^x - u_i^x) + q_3(v_i^y - u_i^y) + q_4(v_i^z - u_i^z))^2 \\ &\quad + (q_1(v_i^x - u_i^x) + q_3(v_i^z + u_i^z) - q_4(v_i^y + u_i^y))^2 \\ &\quad + (q_1(v_i^y - u_i^y) + q_4(v_i^x + u_i^x) - q_2(v_i^z + u_i^z))^2 \\ &\quad + (q_1(v_i^z - u_i^z) + q_2(v_i^y + u_i^y) - q_3(v_i^x + u_i^x))^2) \\ &\quad + \lambda(1 - q_1^2 - q_2^2 - q_3^2 - q_4^2)\end{aligned}$$

Differentiating $\Lambda(\mathbf{q}, \lambda)$ with respect to each \mathbf{q} component and setting to zero, results in a set of linear equations that can be represented in matrix form as an eigenvalue problem:

$$\begin{pmatrix} \sum(x_m^2 + y_m^2 + z_m^2) & \sum(y_p z_m - y_m z_p) & \sum(x_m z_p - x_p z_m) & \sum(x_p y_m - x_m y_p) \\ \sum(y_p z_m - y_m z_p) & \sum(x_m^2 + y_p^2 + z_p^2) & \sum(x_m y_m - x_p y_p) & \sum(x_m z_m - x_p z_p) \\ \sum(x_m z_p - x_p z_m) & \sum(x_m y_m - x_p y_p) & \sum(x_p^2 + y_m^2 + z_p^2) & \sum(y_m z_m - y_p z_p) \\ \sum(x_p y_m - x_m y_p) & \sum(x_m z_m - x_p z_p) & \sum(y_m z_m - y_p z_p) & \sum(x_p^2 + y_p^2 + z_m^2) \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = \lambda \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix}$$

$\vec{q} = (q_1, q_2, q_3, q_4)^\top$ are the unknown (to be solved) quaternion components associated with a 3D rotation, and λ is an (unknown) eigenvalue. In the eigenvalue problem defined in Equation 2.8, the notation x_m , a scalar quantity, denotes the component-wise difference $v_i^x - u_i^x$ (equivalent notations for y_m and z_m) and the scalar x_p denotes the component-wise sum $v_i^x + u_i^x$ (equivalently, y_p and z_p). Diagonalising this 4×4 symmetric matrix yields four eigenvalues and (corresponding) eigenvectors. (Kearsley, 1989) shows that the eigenvector corresponding to the smallest eigenvalue, λ_{\min} , corresponds to the best rotation producing the least squares error,

and the RMSD is computed as $\sqrt{\frac{\lambda_{\min}}{N_e}}$.

Numerical Considerations and Time Complexity

The RMSD and optimal rotation can be solved for analytically using Kearsley (1989), however the process is not numerically stable. Therefore, stable eigenvalue decomposition algorithms such that of Jacobi (1846) are used (Golub and van der Vorst, 2000).

The computation of the 4×4 square symmetric matrix for N_e corresponding vectors requires $O(N_e)$ effort. The size of the matrix is always constant and implementations of the Jacobi diagonalisation algorithm generally bound the maximum number of iterations by a constant. Despite this, the number of iterations is generally very small anyway. Therefore, the eigendecomposition takes constant, $O(1)$ time. Therefore, the computation of RMSD by this method takes linear, $O(N_e)$ time.

2.2.4 Internal Representations Used for Structural Alignment

In Section 2.2.2, the structural alignment problem was divided into two separate tasks: superposition and alignment. Section 2.2.3 discussed the superposition task. This section begins the discussion of the alignment task by introducing several methods used by alignment programs to store the protein structural data required to compute an alignment.

This is important because the type of structural representation used to compute an alignment has consequences for the trade-off between speed and accuracy of the alignment task. Alignment algorithms may use several of these representations beginning with coarse representations for rapid generation of approximate alignments and advancing to more detailed representations to accurately refine the alignment.

Note that except in specific cases, all structural representations share some common features: the order of amino acids in the protein is conserved,** they are invariant to the coordinate system used to define the protein, and they have a level of robustness against errors in structure determination (Eidhammer et al., 2004).

A coordinate representation. The most accurate methods describe a protein structure in terms of the 3D coordinates of individual atoms. These are expressed as points in \mathbb{R}^3 space, which are superimposed within a common frame of reference, as introduced Section 2.2.3. Most often, only one or two atoms are used to represent each amino acid residue. Almost universally, this is the central C_α carbon atom. However, some alignment programs also include the C_β atom for side chain orientation. This representation is the most common in structural alignment programs and is the most detailed (least abstracted from protein structures themselves) and, thus, also requires more memory to store and more processing time to align. Popular structural alignment programs that use this type of representation include CE (Shindyalov and Bourne, 1998), MaxSub (Siew et al., 2000), FatCat (Ye and Godzik, 2003), LGA (Zemla, 2003), MultiProt (Shatsky et al., 2004), CLICK (Nguyen et al., 2011), TM-Align (Zhang and Skolnick, 2005b) and SPAlign (Yang et al., 2012).

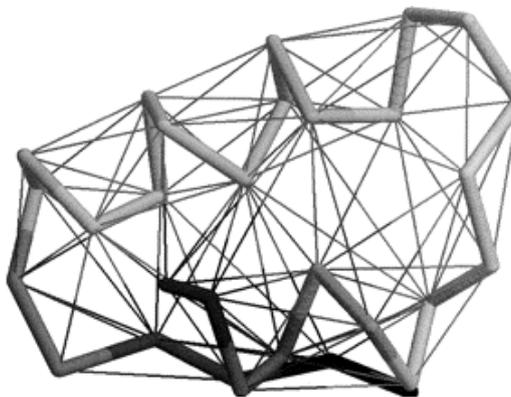
Distance matrices and contact maps. These are 2D representations of the 3D structural information defined by their C_α atoms. With a distance matrix representation, a protein structure, S , is represented by a matrix containing all-against-all inter-atomic distances within a distance threshold. A contact map represents S using a matrix of boolean values where and inter-atomic contact threshold is defined and all inter-atomic distances are either outside, or within the contact distance threshold. In both cases the matrix is of the order $|S| \times |S|$. This representation is independent of the frame of reference and has an additional scope for handling structural plastic deformation. The 3D structure of proteins can be reconstructed from these representations, excepting chirality (Eidhammer et al., 2004). The advantages of these matrices for alignment programs, despite the potentially large memory usage, are that they are naturally independent of the coordinate frame of reference (require no processing to

**Under certain ordering constraints; *e.g.*, cyclical shifting

make them independent), and that similar 3D structures are clearly distinguishable using these representations. Notable examples of structural alignment programs that use distance matrices include, DALI (Holm and Sander, 1993), RAPIDO (Mosca et al., 2008), and GANGSTA+ (Guerler and Knapp, 2008).

A tessellated representation. This involves dividing the space within a protein between residues such that close spatial neighbours share a face, edge, or vertex. Tessellation of the space (as in Figure 2.10) within a protein provides a coordinate system independent representation that also eliminates the requirement for a specified contact threshold. Contacts between residues are well defined and the representation reflects the local geometry of the protein. Different tessellations, including the Delaunay triangulation (Delaunay, 1934) and the Voronoi decomposition (Voronoi, 1908), have been used by structural alignment algorithms. Notable examples of alignment programs that use a tessellated representation include TOPOFIT (Ilyin et al., 2004), which uses a Delaunay tessellation representation, and Vorolign (Birzele et al., 2006), which applies a Voronoi tessellation.

Figure 2.10: An example of the Delaunay tessellation of a Crambin (wwPDB 1CRN) protein. The thick line showing connected C_α atoms represents the backbone of the protein. The thinner lines represent the tessellation. This figure is reproduced from Ilyin et al. (2004).



Coarse grained representations. The above representations are all accurate enough to produce residue level structural alignments. The following representations, beginning with structural profiles, are used to produce coarse, approximate alignments for rapid database matching or later refinement. The general structure, or topology, of the protein can be broken down into a representative string of secondary structural elements, or any discrete alphabet and their relative geometry. This type of representation can be used for rapid pattern matching. An example of this form of representation is the Tableau presented by Lesk (1995) and Konagurthu et al. (2008). A Tableau represents a protein by the relative orientations of its secondary structures. An example Tableau can be found in Figure 2.11. Alternatively, a representative library of protein structural fragments may be used instead of secondary structures as in KL-strings (Friedberg et al., 2007) which represent the entire protein structure using a set of fragments, and each fragment with a symbol resulting in a string which can be used by string pattern matching algorithms.

Finally, and most abstractly, the entire protein structure can be reduced to a single number or string of numbers, a *fingerprnt* (Wolfson and Rigoutsos, 1997; Chu et al., 2008; Sael et al., 2008; Teichert et al., 2007; Zotenko et al., 2007) or histogram (Budowski-Tal et al., 2010). Where similar structures evaluate to similar or identical fingerprints, database searches can be performed extremely quickly. However, their utility in structural alignment is limited and approximate matching is tricky (Hasegawa and Holm, 2009).

β_1	OT	KK	HH	PD	HH	KK
OT	α_A	OT	OT	OT	PE	OT
KK	OT	β_2	HH	PE	HH	KK
HH	OT	HH	β_3	OT	KK	HH
PD	OT	PE	OT	α_B	OS	PE
HH	PE	HH	KK	OS	β_4	RT
KK	OT	KK	HH	PE	RT	β_4

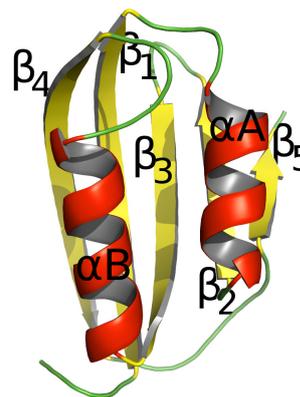


Figure 2.11: (a) An example tableau representation for the wwPDB 2ACY protein structure. The diagonal is labeled with the secondary structure elements, the off-diagonals contain the encoded relative orientations between secondary structural axes. Each relative orientation is described by two letters. The first letter corresponds to one of the following: P-parallel; O-antiparallel; R-crossing right; and L-crossing left. The second letter corresponds to the division of a circle into quadrants for time of day: E-“elevenes”, D-“dinner”, S-“supper”, and T-“tea”. Finally, two extra symbols are used to designate adjacent strands in the same sheet: KK for anti-parallel strands, and HH for parallel strands. (b) The protein structure being encoded: wwPDB 2ACY, the acylphosphatase protein from cow. The diagram contains labels on secondary structural elements. α labels are used for helical secondary structures while, β labels are used for strand secondary structures.

2.2.5 Protein Structural Alignment Scoring Functions

In order to decide if a given alignment is good, a system for assigning a quality value or score to alignments needs to be defined reflect the extent of similarities (or dissimilarities) between the aligned structures. This section presents an overview of some commonly used scoring functions used in the structural alignment literature. This list is definitely not exhaustive, but gathers only either the most popular and widely used measures, or those that are unique in the technique they use. For a more comprehensive listing, the reader may wish to refer to other reviews of structural alignment methods including Kolodny et al. (2005); Sippl and Wiederstein (2008); Hasegawa and Holm (2009); Slater et al. (2013) and Ma and Wang (2014).

A scoring function defines an objective measure by which to gauge the similarity of protein structures and an objective function by which to search for an optimal alignment. Intuitively, the scoring function chosen should give a high score to structures that have large regions that are geometrically similar (see Figure 2.9), and a low score to proteins that bear little structural similarity.

Traditionally, the approach to determine the similarity of protein structures has been based on two key criteria: *coverage* and *fidelity*.

Coverage measures the number of correspondences in an alignment and, in some cases, also considers the number of gaps.

Fidelity measures how similarly positioned the aligned residues are. This is commonly (but not always) based on the RMSD computed after the best rigid-body superposition of corresponding residues (See Section 2.2.3).

To search for the *best* structural alignment, the goal of alignment algorithms is to simultaneously maximise coverage and fidelity. However, these two objectives are in direct conflict with each other. Increasing the number of corresponding residues, or *coverage*, usually leads to a loss of structural *fidelity* and vice versa. Therefore, alignment quality scoring functions must reconcile this conflict by arbitrating, in various ways, between these two key criteria.

A table of various structural alignment scoring functions is presented in Table 2.2. It is not important to fully comprehend each measure listed, rather it is important to note the various ways in which the measures attempt to balance coverage with fidelity. To aid in reading the table, terms for coverage are highlighted in blue and terms for fidelity are highlighted in red. **Where applicable, this thesis will continue to use blue/red colours to highlight the coverage/fidelity terms respectively.**

Table 2.2: A table of scoring functions used by popular alignment methods when determining the quality of an alignment between protein structures S and T . The first column provides the name of the method, the second the mathematical formula for the scoring function, the third criteria by which to decide if the alignment is significant where such criteria could be found in the literature (if the criteria could not be found this column is marked with N/A), and the last column provides references for the details of the scoring function. The symbols used in this table are as follows: N_e is the number of correspondences; N_g is the number of gaps (the number of insertions plus the number of deletions); $N_{\text{contiguous}}$ is a number of contiguous corresponding residue pairs; RMSD is the root mean squared deviation (as above, see Equation 2.3); \vec{S}_i is the vector from the origin to the i^{th} coordinate in the structure S ; L_{\min} is the minimum length between the structures being aligned: $\min(|S|, |T|)$; L_{\max} is the maximum length between the structures being aligned: $\max(|S|, |T|)$; δ_i is the distance between the i^{th} pair of corresponding C_α atoms after rigid body superposition; C is a list of cut-off thresholds; w is a user defined weighting parameter; ϕ and ψ are the dihedral angles; $(\phi, \psi)_i$ is the straight line distance in the Ramachandran-Ramakrishnan-Sasisekharan plot between the (ϕ, ψ) points of the i^{th} pair of equivalent residues; $R(l)$ is the density at a point, l , in the Ramachandran-Ramakrishnan-Sasisekharan plot. Note that alignment quality scores that are non-residue-level, such as secondary structural alignment scores, are not included in this table. Terms for *coverage* are highlighted in blue and terms for *fidelity* are highlighted in red.

Measure	Formulation	Significance Criteria (Same Fold)	References
SSAP	$\sum_{i=1}^{N_e} \sum_{j=1}^{N_e} \frac{500}{10 + \left (\vec{S}_{i-2} - \vec{S}_i) - (\vec{T}_{j-2} - \vec{T}_j) \right }$	$\ln(\text{SSAP}) \times \frac{100}{\ln(50)} \geq 70$	(Taylor and Orengo, 1989) (Orengo and Taylor, 1990) (Orengo and Taylor, 1996)
DALI	$\sum_{i=1}^{N_e} \sum_{j=1}^{N_e} \begin{cases} \Theta^E - \frac{ d_{ij}^S - d_{ij}^T }{d_{ij}^*} e^{(d_{ij}^*/\alpha)^2}, & i \neq j \\ \Theta^E, & i = j \end{cases}$	$L = \sqrt{ S \times T }$ $L' = 7.95 + 0.71L - 0.000259L^2 - 1.92 \times 10^{-6}L^3$ $\frac{\text{DALI} - L'}{0.5 \times L'} > 2$	(Holm and Sander, 1993) (Holm and Sander, 1998)
SAS	$\frac{\text{RMSD} \times 100}{N_e}$	N/A	(Subbiah et al., 1993)

Table 2.2: (continued)

Measure	Formulation	Significance Criteria (Same Fold)	References
STRUCTURAL score (S_g)	$\sum_{i=1}^{N_e} \frac{20}{1 + (\delta_i/5)^2} - 10N_g$	$Z = \begin{cases} \frac{s_s - (c \ln(N_e)^2 + d \ln(N_e) + e)}{f \log(N_e) + g}, N_e < 120 \\ \frac{s_s - (a \ln(N_e) + b)}{f \ln(N_e) + g}, N_e \geq 120 \end{cases}$ $a = 0.872, b = 0.65, c = 0.155,$ $d = -0.619, e = 1.73,$ $f = 0.0922, g = 0.212$ No threshold for fold similarity was found	(Subbiah et al., 1993) (Gerstein and Levitt, 1998) (Levitt and Gerstein, 1998)
MaxSub	$\frac{1}{L_{\max}} \sum_{i=1}^{N_e} \frac{1}{1 + (\delta_i/3.5)^2}$	N/A	(Siew et al., 2000)
PSI (MAMMOTH)	$\frac{N_{\delta \leq 4\text{\AA}}}{L_{\min}}$	N/A	(Ortiz et al., 2002)
GDT_TS	$100 \times \frac{\sum_{i=1}^4 \frac{N_{\delta_i \leq C_i}}{N_e}}{4}, C = \{1, 2, 4, 8\}$	N/A	(Zemla, 2003)
GDT_HA	$100 \times \frac{\sum_{i=1}^4 \frac{N_{\delta \leq C_i}}{N_e}}{4}, C = \{0.5, 1, 2, 4\}$	N/A	(Zemla, 2003)
LCS	$\frac{N_{\text{contiguous: } \delta \leq C}}{N_e}$	N/A	(Zemla, 2003)
LGA_S3	$\frac{2w}{ C (C +1)} \sum_{i=1}^{ C } \left(\frac{ C - i + 1}{ C } \text{GDT}(C_i) \right) + \frac{2(1-w)}{ C' (C' +1)} \sum_{i=1}^{ C' } \left(\frac{ C' - i + 1}{ C' } \text{LCS}(C'_i) \right),$ $C = \{0.5, 1, \dots, 10\}, C' = \{1, 2, 5\}, 0 \leq w \leq 1$ This scoring function is a combination of GDT and LCS.	N/A	(Zemla, 2003)
Q-score	$\frac{N_e^2}{ S T (1 + \text{RMSD}/3)^2}$	N/A	(Krissinel and Henrick, 2003) (Krissinel and Henrick, 2004)
TM-Score	$\frac{1}{L_{\min}} \sum_{i=1}^{N_e} \frac{1}{1 + \left(\frac{\delta_i}{1.24 \sqrt[3]{L_{\max} - 15} - 1.8} \right)^2}$	TM-Score ≥ 0.5	(Zhang and Skolnick, 2004) (Zhang and Skolnick, 2005b) (Xu and Zhang, 2010)
TopoFit	$\frac{N_e - e^{0.84\text{RMSD} + 1.25}}{e^{\text{RMSD} + 1.64} - e^{0.84\text{RMSD} + 1.25}}$	TopoFit ≥ 3	(Ilyin et al., 2004) (Leslin et al., 2007)
GSAS	$\begin{cases} \frac{\text{RMSD} \times 100}{N_e - N_g}, N_e > N_g \\ 99.9, N_e \leq N_g \end{cases}$	N/A	(Kolodny et al., 2005)
SI	$\frac{\text{RMSD} \times L_{\min}}{N_e}$	N/A	(Kleywegt and Jones, November 1994) (Kolodny et al., 2005)
MI	$1 - \frac{1 + N_e}{(1 + \text{RMSD}/1.5)(1 + L_{\min})}$	N/A	(Kleywegt and Jones, November 1994) (Kolodny et al., 2005)
Relative Similarity (TOPMATCH)	$100 \times \frac{2N_e}{ S + T }$	N/A	(Sippl, 2008) (Sippl and Wiederstein, 2008)
TALI	$\frac{1}{N_e} \sum_{i=1}^{N_e} e^{-\int_{(\phi, \psi)_i} R(l) dl}$	N/A	(Miao et al., 2008)
SP-score (SP _b)	$\frac{1}{3 \times L_{\min}^{0.7}} \sum_{\delta_i < 8} \frac{1}{1 + (\delta_i/4)^2} - 0.2$	SP-score ≥ 0.523	(Yang et al., 2012)
SP-score (SP _a)	$\frac{1}{3((S + T)/2)^{0.7}} \sum_{\delta_i < 8} \frac{1}{1 + (\delta_i/4)^2} - 0.2$		

SSAP (Taylor and Orengo, 1989; Orengo and Taylor, 1990, 1996) is one of the earliest automatic methods for protein structural alignment. **SSAP** looks for local structural similarities which it sums up into an aggregate global alignment. Significance criteria, normalised to have a maximum value of 100, is defined (Orengo and Taylor, 1996) as in Table 2.2 above. A value in the range 60–70 refers to a pair of structures approximately in the same Class. A value in the range of 70–80 indicates a pair of structures with a similar Fold, and a value above 80 implies a close structural relationship.

The **DALI** (Holm and Sander, 1993) algorithm introduced an elastic similarity score based on distance matrices. This allows **DALI** to recognise structural relationships which rigid scoring functions cannot find accurately or even at all. The **DALI** score is intended to compensate for structures that are not rigid by using a relative (averaged) distance, rather than an absolute distance between residues. **DALI** also defines an empirical z-score to determine the significance of an alignment. The **DALI z-score** is based on an empirical distribution of **DALI** scores as a function of the size of the protein (to eliminate any length dependence).

The Structural Alignment Score (**SAS**) (Subbiah et al., 1993) and Gapped SAS (**GSAS**) (Kolodny et al., 2005) are simple geometric scores which combine the RMSD after superposition and the number of equivalent residues and the number of gaps in the alignment. Kolodny et al. (2005) define two other simple geometric scores to aid in deciding between alternative alignment quality scoring functions. These are the Similarity Index (**SI**) and the Match Index (**MI**). The **TOPMATCH** (Sippl, 2008) relative similarity score is a similar simplistic geometric score (see Table 2.2).

Subbiah et al. (1993) introduce a scoring function which, in Table 2.2 is referred to as the **STRUCTAL_score**. This score has some statistical basis and has become somewhat of a standard for small modifications such as the **MaxSub** (Siew et al., 2000) score. **TM-Score** (Zhang and Skolnick, 2004) is a further modification to remove the length dependence of the **STRUCTAL** scoring function with an empirical cut-off value. The **TM-Score** is then defined in relation to the length of one of the structures being aligned. The authors of the **SP-Score** (Yang et al., 2012) measure argue that **TM-Score** is not fully length independent, and propose a further modification to the length independent score. Instead of scaling the cutoff value, a normalisation factor is used. The **Q-Score** defined by **SSM** (Krissinel and Henrick, 2003, 2004) is defined in a similar fashion but can be computed faster because it does not require a summation over all correspondences.

Critical Assessment of Structure Prediction (**CASP**) is a regular competition for the prediction of protein structures from their sequences. The scoring function used for ranking the quality of entries to this competition is the Global Distance Test: Total Score (**GDT_TS**) (Zemla, 2003). This scoring function sets four increasing distance thresholds and defines its score as the average proportion of correspondences that fall within these distance thresholds. Zemla (2003) also defines a stricter version of **GDT_TS**, called **GDT_HA** or **GDT: High Accuracy** with a tighter set of distance thresholds (**GDT_TS** thresholds halved). **GDT_TS** is the first of two scoring functions that make up an **LGA** score (referred to in this thesis by **LGA_S3** because this is the output description of the program supplied by Zemla (2003)), the second being the Longest Continuous Segments (**LCS**). **LCS** is defined as the longest contiguous section of corresponding residues that fall below a user-supplied distance threshold. The **LGA_S3** score fixes a set of thresholds for **LCS** and mixes it with **GDT_TS** using a weighted average for thresholds: smaller thresholds having a greater impact on the score than larger thresholds. These scoring functions are easy and fast to compute, but simplistic. **MAMMOTH** (Ortiz et al., 2002) defines a similar structural score called Percentage of Structural Similarity (**PSI**), which is defined as the proportion of correspondences below a distance threshold of 4Å.

Uniquely amongst the scoring function described here is TALI (Miao et al., 2008), which uses a torsion angle distance based on the statistical density of the Ψ and ϕ torsion angles in the Ramachandran-Ramakrishnan-Sasisekharan plot (See Section 2.1.5).

In summary, except for some notable examples that have a statistical basis (especially `STRUCTAL_score` (Subbiah et al., 1993) and TALI (Miao et al., 2008)), the scoring functions mentioned above are ad hoc combinations of heuristic measures of alignment quality. There is no consensus on how to measure structural alignment quality, as each score arbitrarily weighs a measure of coverage against a measure of fidelity to arbitrate between alignments. Recent reviews have shown that, as a result of these ad hoc formulations, the alignments obtained from programs using these measures often contradict each other (Kolodny et al., 2005; Sippl and Wiederstein, 2008; Hasegawa and Holm, 2009; Slater et al., 2013; Ma and Wang, 2014). However, it is recognised that in individual instances, some of these ad hoc measures generate useful alignments for biologists that are able to interpret different aspects of structural relationships through, for example, aligned active and binding sites or similarities of protein-protein interfaces (Hasegawa and Holm, 2009; Grishin and Phillips, 1994). Nevertheless, in the majority of cases, these contradictory results do highlight a severe disconnect between the rapidly growing number of methods and the quality of the structural alignments that programs using them generate. These quality measures do not provide any framework to meaningfully capture similarities and differences between *competing* alignments. This traditional approach to formulating an ad hoc scoring function from key alignment quality criteria, has been extensively explored over the last four decades. The foundations for a radically new method of objectively and rigorously selecting an alignment from a set of alternatives is described in the next section. This is applied to the problem of quantifying alignment quality described in Chapter 4.

2.3 Summary

This chapter has presented general biological background on proteins and protein structure required to understand the research presented in subsequent chapters. It also summarises the field of protein structure comparison methods. There are a very large number of these methods which is doubling approximately every 5 years (Hasegawa and Holm, 2009). All of the scoring functions presented above make a trade-off, in some form, between alignment coverage and goodness of fit. The terms involved in this trade-off are highlighted in the equations in Table 2.2. A cursory examination of this table reveals a lack of agreement on how to measure structural alignment quality. Without this, the notion of best (or *optimal*) alignment can neither be rigorously defined nor searched for (Hasegawa and Holm, 2009).

Chapter 3

Introduction to Statistical Inference

“The supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience.”

— A. Einstein (1933)

This chapter provides a primer on the probabilistic and statistical concepts used in the research presented in the rest of the thesis. In doing so, it introduces key ideas involving the theory of probability and statistical inference. In particular, it provides a broad overview of inductive inference using the Minimum Message Length (MML) principle, a framework that underpins the methodology used for the research introduced in this thesis.

3.1 Definitions and Notations

Random experiment, trial, outcome and event: A random experiment is a procedure that can be repeated as many times as under the same conditions, where the result is uncertain. For example, rolling a die, or tossing a coin. Each repetition of a procedure is called a trial. Each trial has an outcome. The set of outcomes of a random experiment is called an event.

Sample space and random variable: The set of all possible events (outcomes) for a given random experiment (trial) is called the sample space Ω of that experiment (trial). The sample space of an event (outcome) can be a discrete set of possibilities, or a continuous set of possibilities. For example, if the random experiment is to toss a coin twice, let the first trial result in Heads, and the second trial results in Tails. The outcomes of the experiment are (Heads, Tails). The sample space for the trials is {Heads, Tails}. The sample space from the experiment is {(Heads, Heads), (Heads, Tails), (Tails, Heads), (Tails, Tails)}. Based on the underlying procedure, each repetition of a random experiment (trial) can potentially yield a different event (outcome). A random variable \mathbf{X} defines a mapping function $\mathbf{X} : \Omega \rightarrow \mathbb{R}$ that maps an event (outcome) $\omega \in \Omega$ of a random experiment (trial) to a distinct real valued number $e \in \mathbb{R}$, not to be confused with the probability of the event (outcome). If the random variable \mathbf{X} produces an outcome ω that maps to the value e , this is denoted as $\mathbf{X} = e$.

Event probabilities and distributions: Each event (outcome) e from a random experiment (trial) has a probability, denoted by $\Pr(\mathbf{X} = e)$. (For brevity, this is also represented as $\Pr(e)$.) Note that these probabilities, over the sample space of the experiment (trial), are often non-uniform. The function that assigns probabilities for all measurable set of events (outcomes) over the entire sample space of a random experiment (trial) is called a *probability distribution*. By the total probability theorem, $\sum_{\forall e_i \in \mathbf{X}} \Pr(\mathbf{X} = e_i) = 1$. For a sample space that is discrete, the distribution is called the probability mass function. For a sample space that is continuous, the distribution is called probability density function.

Joint, marginal and conditional probabilities: If \mathbf{X} and \mathbf{Y} denote two random variables, which are not necessarily independent, then they are said to be joint random variables. Intuitively, the random variables are joint when an experiment (trial) produces an ordered pair of outcome values e_x and e_y . For example, the probability that a card drawn from a deck of 52 cards is both *red* and an *eight*. The joint probability of the outcome $\mathbf{X} = e_x$ and $\mathbf{Y} = e_y$ is denoted by $\Pr(\mathbf{X} = e_x, \mathbf{Y} = e_y)$. For brevity, this is also represented as $\Pr(e_x, e_y)$. The marginal (or unconditional) probability of the outcome value e_x is:

$$\underbrace{\Pr(\mathbf{X} = e_x)}_{\substack{\text{marginal} \\ \text{probability}}} = \sum_{\forall e_{y_i} \in \mathbf{Y}} \Pr(\mathbf{X} = e_x, \mathbf{Y} = e_{y_i})$$

The conditional probability of the outcome $\mathbf{X} = e_x$ given (or conditioned upon) the outcome $\mathbf{Y} = e_y$ is:

$$\underbrace{\Pr(\mathbf{X} = e_x | \mathbf{Y} = e_y)}_{\text{conditional probability}} = \frac{\Pr(\mathbf{X} = e_x, \mathbf{Y} = e_y)}{\Pr(\mathbf{Y} = e_y)}$$

For example, what is the probability of the card drawn from the 52 card deck being an *eight*, given that it is known to be *red*.

Product rule of probability: For joint random variables \mathbf{X} and \mathbf{Y} , the product rule gives the relationship between joint, marginal and conditional probabilities as:

$$\begin{aligned} \underbrace{\Pr(\mathbf{X} = e_x, \mathbf{Y} = e_y)}_{\text{joint probability}} &= \Pr(\mathbf{X} = e_x) \Pr(\mathbf{Y} = e_y | \mathbf{X} = e_x) \\ &= \Pr(\mathbf{Y} = e_y) \Pr(\mathbf{X} = e_x | \mathbf{Y} = e_y) \end{aligned}$$

It is easy to see that, if the two random variables are independent of each other, represented as $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}$, then product rule becomes:

$$\Pr(\mathbf{X} = e_x, \mathbf{Y} = e_y) = \Pr(\mathbf{X} = e_x) \Pr(\mathbf{Y} = e_y)$$

Bayes theorem (Bayes and Price, 1763): This fundamental theorem follows from the product rule and can be used to characterise the probability of an event (outcome), based on conditional probabilities related to that event (outcome) as:

$$\underbrace{\Pr(\mathbf{X} = e_x | \mathbf{Y} = e_y)}_{\substack{\text{posterior probability} \\ \text{of } \mathbf{X} = e_x \text{ after} \\ \text{observing } \mathbf{Y} = e_y}} = \frac{\overbrace{\Pr(\mathbf{X} = e_x)}^{\text{prior probability of } \mathbf{X} = e_x} \overbrace{\Pr(\mathbf{Y} = e_y | \mathbf{X} = e_x)}^{\text{Likelihood of } \mathbf{Y} = e_y}}{\underbrace{\Pr(\mathbf{Y} = e_y)}_{\substack{\text{prior probability} \\ \text{of } \mathbf{Y} = e_y}}}$$

In this Bayesian interpretation of the product rule, each probability term reflects a degree of belief (see Section 3.2 for details). Specifically, $\Pr(\mathbf{X} = e_x)$ and $\Pr(\mathbf{Y} = e_y)$ are the initial degrees of belief (or *priors*) about the outcome $\mathbf{X} = e_x$ and $\mathbf{Y} = e_y$ respectively. The conditional probability term $\Pr(\mathbf{Y} = e_y | \mathbf{X} = e_x)$ gives the degree of belief (or *likelihood*) of the outcome $\mathbf{Y} = e_y$ given that $\mathbf{X} = e_x$ has been observed. Finally, the conditional probability term $\Pr(\mathbf{X} = e_x | \mathbf{Y} = e_y)$ gives the degree of belief (or *posterior probability*) of the outcome $\mathbf{X} = e_x$, having accounted for the outcome $\mathbf{Y} = e_y$.

Expectation of a random variable: For a random variable \mathbf{X} , whose outcome values, e , are mapped to real numbers, each with probability $\Pr(e)$, the expectation of \mathbf{X} , represented as $\mathbb{E}(\mathbf{X})$, defines the average outcome value of \mathbf{X} under the probability distribution of \mathbf{X} . When the random variable has a discrete set of outcomes, then the expectation can be formalised as: $\mathbb{E}(\mathbf{X}) = \sum_{\forall e \in \mathbf{X}} e \Pr(e)$. And for a continuous set of outcomes: $\mathbb{E}(\mathbf{X}) = \int_{\forall e \in \mathbf{X}} e \Pr(e) de$.

3.2 Probability, Information and Entropy

Over the last few centuries, the precise meaning of probability has been interpreted in many different ways. However, two interpretations stand out and are widely used (Hájek, 2003):

Physical interpretation of probability: In this interpretation, the probability of an outcome is objectively determined as the relative frequency of an outcome in a long run

of trials from a random experiment. For example, if an experiment containing 10 trials of tossing a coin produces the string of outcomes: HTHHTTHTT, then the probability of getting Heads in the next toss with the same coin is estimated by this relative frequency definition as: 0.4.

Evidential or Bayesian interpretation of probability In this interpretation, any proposition can be assigned a probability that reflects a subjective plausibility or *degree of belief* about that proposition. This is founded on, and supported by empirical evidence. For example, the proposition “*all swans are white*” can be assigned a probability of (say) 0.85. This degree of belief may arise from the available evidence, and can be updated in the face of new evidence.

The degree of belief (Bayesian) interpretation of probability, for some outcome $\mathbf{X} = e$, is tied to the measure of information $I(e)$ conveyed by the statement of that outcome. But what is information? According to Wallace (2005), information is something that *decreases the uncertainty* about some outcome. Intuitively, in this context the measurement of information of an outcome can be seen as the length of the shortest statement (or message) conveying that outcome.

Such a measure of information is a continuous and decreasing function of probability. For two outcome values $\{e_1, e_2\}$ of the same random variable \mathbf{X} , if $\Pr(e_1) > \Pr(e_2)$, then $I(e_1) < I(e_2)$. In the extreme case, when $\Pr(e) = 1$ then $I(e) = 0$: that is, no (extra) information needs to be conveyed about an outcome that is certain. Further, suppose that e_1 and e_2 are two independent outcomes. Then, it is easy to see that the measure of information of both these outcomes, $I(e_1, e_2)$ is simply the sum of individual measures of information: $I(e_1, e_2) = I(e_1) + I(e_2)$. Since, as stated above, information is a function of probability, $I(e_1, e_2)$ is a function of the joint probability $\Pr(e_1, e_2)$ of the independent event which can be expressed as $\Pr(e_1)\Pr(e_2)$. In other words, information (as the function of probability) is additive when probabilities are multiplicative.

Bringing all of these observations together through the landmark work of Claude E. Shannon (Shannon, 1948), the mathematical function that fits all the above requirements is:

$$\text{Shannon Information content: } I(e) = \log\left(\frac{1}{\Pr(e)}\right) = -\log(\Pr(e)) \quad (3.1)$$

This measure of information is defined as the *Shannon information content* of the outcome e (MacKay, 2003). Equation 3.1 suggests that, given the probability of event e is $\Pr(e)$, the length of the message required to explain the event is equal to the negative logarithm of the probability of e .

The unit of Shannon information content depends on the base of the logarithm used. For base-2 logarithms, the unit is *bits*. Thus, the measure of information in base-2 can be seen as the length of message in bits required to encode the statement of the event e over a noisy binary channel (such as the internet).^{*} To further understand Shannon information content consider, for example, tossing an unbiased coin, giving equally likely outcomes: *heads* or *tails*. The probability of the coin landing on either side is 0.5. Applying Equation 3.1: $-\log_2(0.5) = 1$ *bit*. Therefore, 1 bit of information is required to uniquely state (or *encode*) the outcome of an unbiased coin flip.

Another closely-related concept to Shannon information content is *Shannon entropy*. The entropy of a random variable \mathbf{X} is defined as its average (or expectation of) Shannon information

^{*}The logarithm defined in other bases changes the unit of Shannon’s information content. For instance, using the natural base- e yields the unit *nats* or *nits*, while using base-10 yields *digits* or *hartleys*.

content: $H(\mathbf{X}) = \mathbb{E}(I(\mathbf{X}))$. If \mathbf{X} is discrete, then Shannon entropy of \mathbf{X} takes the form:

$$H(\mathbf{X}) = \sum_{\forall e \in \mathbf{X}} \Pr(e)I(e) = - \sum_{\forall e \in \mathbf{X}} \Pr(e) \log(\Pr(e))$$

When the random variable is distributed continuously, the sum in the above equation is replaced by an integral:

$$H(\mathbf{X}) = \int_{\forall e \in \mathbf{X}} \Pr(e)I(e)de = - \int_{\forall e \in \mathbf{X}} \Pr(e) \log(\Pr(e))de$$

As with Shannon information content, the base of the logarithm indicates the units in which entropy is measured.

3.3 Statistical Inference, Model Comparison and Selection

Statistical inference involves arriving at a hypothesis (or a theory) about the underlying distribution of empirically observed data. However, in many real world settings, the data often comes from a process that has an unknown *true* (probability) distribution. Typically, a hypothesis is defined using well-characterised probability distribution(s) (or model(s)) to explain the observed data. Thus, statistical inference commonly involves comparing a countable set of well-characterised statistical model(s), and selecting a suitable model that best describes the observed data, while simultaneously inferring its statistical parameters. In other words, many real world inference problems are often reformulated as problems of model comparison and selection, and of parameter estimation of the selected models.

Inferring hypotheses on the given data is not merely a data analysis task, it can potentially allow predictions about unobserved (future) data, and also facilitate decision making based on the data to optimise some utility (or objective) function (Oliver and Baxter, 1994). In this regard, the Bayesian framework provides the foundation of model selection and inference, supports prediction of unobserved data, allows updating beliefs about the current hypothesis in light of additional data, and provides a utility function for decision making. Before considering inference in this Bayesian framework, the notation required to support this discussion is defined.

3.3.1 Notations Supporting Statistical Inference

The rest of this chapter will use the following notations, commonly used in the statistical inference literature (Wallace, 2005; Wallace and Boulton, 1975; Wallace and Freeman, 1987; Farr and Wallace, 2002):

Θ denotes the (super)set of all possible hypotheses (also theories/models/parameters).

X denotes the (super)set containing all possible observations of the data.

$\vec{\theta} \in \Theta$ denotes a particular hypothesis (theory/model/parameter) from the set Θ . In this chapter, $\vec{\theta}$ is used interchangeably with \mathcal{H} .

$x \in X$ denotes a particular instance of the observed data. In this chapter x is used interchangeably with \mathcal{D} .

$h(\vec{\theta})$ denotes the prior probability density function of $\vec{\theta}$ before any data has been observed. This gives the prior probability $\Pr(\mathcal{H})$ for any given hypothesis \mathcal{H} .

$g(\vec{\theta}|x)$ denotes the posterior probability density function of $\vec{\theta}$ after the data has been observed. This gives the posterior probability $\Pr(\mathcal{H}|\mathcal{D})$.

$f(x|\vec{\theta})$ denotes the likelihood function. This gives the probability $\Pr(\mathcal{D}|\mathcal{H})$, of data \mathcal{D} assuming that the hypothesis \mathcal{H} is true.

$r(x)$ denotes the marginal (or unconditional, prior) probability function. This gives the unconditional probability of the data, $\Pr(\mathcal{D})$.

$j(\vec{\theta}, x)$ denotes the joint probability density function. This gives the joint probability of the hypothesis and the data, $\Pr(\mathcal{H}, \mathcal{D})$.

3.3.2 Statistical Estimators and Common Methods of Parameter Estimation

A statistical estimator is a function of the observed data, $\hat{\Theta} : X \rightarrow \Theta$, that is used to estimate (guess) an unknown parameter of some statistical model that describes the data. Since this is a function of the observed data $x \in X$, the estimator $\hat{\Theta}(x)$ is the random variable, where the specific values that this function produces $\hat{\theta} = \hat{\Theta}(x)$ are called the *point estimates*.

For some given data \mathcal{D} , let \mathcal{H} define some hypothesis, where a hypothesis implies the supporting statistical models (with statistical parameters). The aim is to estimate the parameters of the hypothesis. The statistical estimator that generates these estimates should possess, amongst others, the following three key properties (Wallace, 2005):

Invariance to reparametrisation (model reparameterisation invariance): Let the parameters of the hypothesis \mathcal{H} be defined in some Θ -space. Consider any reparameterisation that transforms the parameters (invertibly) to some Φ -space under a one-to-one mapping, **Reparameterise** : $\Theta \rightarrow \Phi$. The estimator is said to be invariant to reparameterisation if, for any such transformation, the point estimate in the Φ -space, $\hat{\Phi}(x)$, is *equivalent* (under the inverse of that transformation) to the point estimate in the Θ -space, $\hat{\Theta}(x)$.

Invariance to transformation of the data space (data transformation invariance): Let the observed data defined in some X -space be invertibly transformed into some Y -space under a one-to-one mapping, **Transform** : $X \rightarrow Y$. The estimator is said to be invariant to the transformation of the data if the point estimate of the data in the Y -space, $\hat{\Theta}(y \in Y)$ is *equivalent* (under the inverse of that transformation) to the point estimate of the data in the X -space, $\hat{\Theta}(x \in X)$.

Zero bias of the estimator: Let $\vec{\theta}^*$ be the true parameter of a statistical model generating the observed distribution of data $f(x|\vec{\theta}^*)$. Then the bias of an estimator (with respect to the true parameter $\vec{\theta}^*$) is defined as the difference between the expectation of the statistical estimator over the distribution $f(x|\vec{\theta}^*)$ and the true parameter:

$$\text{Bias}(\hat{\Theta}(x), \vec{\theta}^*) = \mathbb{E}(\hat{\Theta}(x)) - \vec{\theta}^*$$

For an unbiased (or zero bias) estimator, $\mathbb{E}(\hat{\Theta}(x)) = \vec{\theta}^*$.

Given these properties, three popular methods of parameter estimation used to address statistical inference problems are discussed below.

Maximum Likelihood Estimation

The maximum likelihood (ML) approach to parameter estimation involves, as the name suggests, choosing an estimate (parameter value) $\vec{\theta}_{ML}$ that maximises the likelihood function $f(x|\vec{\theta})$. The ML estimator is then:

$$\hat{\theta}_{ML} = \underset{\forall \vec{\theta} \in \Theta}{\operatorname{argmax}} f(x|\vec{\theta})$$

Intuitively, this amounts to choosing the parameter value that is most likely to have resulted in the observed data. This estimator is both, model reparameterisation invariant and data transformation invariant. However, the maximum likelihood estimator is a biased estimator.

Often, when Θ space is continuous, this method involves evaluating the function at its extremum when $\frac{d}{d\vec{\theta}}f(x|\vec{\theta}) = 0$. However, this approach is only useful if each $\vec{\theta} \in \Theta$ is equally probable. In other words, this assumes that the prior density function $h(\vec{\theta})$ is uniform. This is only appropriate when no prior knowledge exists about the distribution of Θ . However, more generally this assumption is very limiting until the prior probabilities are satisfactorily accounted for during estimation.

Bayesian Point Estimation of Posterior Mean, Median and Mode

Bayes theorem (Bayes and Price, 1763), as encountered earlier in Section 3.1, is reformalised in Equation 3.2. This allows restating the degree of belief in an hypothesis in light of new evidence. That is, the conditional probability of the hypothesis \mathcal{H} given the data \mathcal{D} is proportional to the probability of \mathcal{H} multiplied by the conditional probability of \mathcal{D} given knowledge of \mathcal{H} . More precisely, the general form of Bayes theorem used for the inference problems is:

$$\Pr(\mathcal{H}|\mathcal{D}) = \frac{\Pr(\mathcal{H}) \times \Pr(\mathcal{D}|\mathcal{H})}{\Pr(\mathcal{D})} \quad (3.2)$$

Expressing the same in terms of the joint probability ($\Pr(\mathcal{H}, \mathcal{D})$) by applying the product rule results in the following:

$$\begin{aligned} \underbrace{\Pr(\mathcal{H}, \mathcal{D})}_{\text{Joint Pr.}} &= \underbrace{\Pr(\mathcal{H})}_{\text{Prior on } \mathcal{H}} \times \underbrace{\Pr(\mathcal{D}|\mathcal{H})}_{\text{Likelihood}} \\ &= \underbrace{\Pr(\mathcal{D})}_{\text{Prior on } \mathcal{D}} \times \underbrace{\Pr(\mathcal{H}|\mathcal{D})}_{\text{Posterior}} \end{aligned} \quad (3.3)$$

Using the functional notations in Section 3.3.1, the above can be written as:

$$j(\vec{\theta}, x) \propto h(\vec{\theta})f(x|\vec{\theta}) \propto r(x)g(\vec{\theta}|x),$$

or more generally as:

$$g(\vec{\theta}|x) \propto h(\vec{\theta})f(x|\vec{\theta}).$$

There are several traditional Bayesian methods to summarise (using point estimates) the posterior distribution $g(\vec{\theta}|x)$ using this Bayesian formulation:

Mode estimate of the posterior function: This estimate maximises the posterior density function as:

$$\hat{\theta}_{\text{mode}(g(\vec{\theta}|x))} = \underset{\forall \vec{\theta} \in \Theta}{\operatorname{argmax}} g(\vec{\theta}|x)$$

Note that the mode of the posterior distribution is not invariant under non-linear transformations of both model parameters and data.[†] Another issue with this estimator is that it greedily chooses the peak of the posterior distribution, ignoring the probability mass in that region; there may be other peaks in the distribution that holds significantly larger posterior probability mass.

Mean estimate of the posterior function: This estimate gives the expectation of the random variable Θ on the posterior density function $g(\vec{\theta}|x)$ as:

$$\hat{\theta}_{\text{mean}(g(\vec{\theta}|x))} = \mathbb{E}(\Theta) = \int_{\forall \vec{\theta} \in \Theta} \vec{\theta} g(\vec{\theta}|x) d\vec{\theta}$$

As in the case of the mode estimate, this form of estimation is also not invariant under non-linear transformations of either parameters or the data.

Median estimate of the posterior function: This estimate finds the particular value of the hypothesis that satisfies the following property:

$$\hat{\theta}_{\text{median}(g(\vec{\theta}|x))} = \int_{\forall \vec{\theta} < \hat{\theta}_{\text{median}(g(\vec{\theta}|x))} \in \Theta} g(\vec{\theta}|x) d\vec{\theta} - \int_{\forall \vec{\theta} > \hat{\theta}_{\text{median}(g(\vec{\theta}|x))} \in \Theta} g(\vec{\theta}|x) d\vec{\theta} = 0$$

Unlike the mean and mode estimates above, this form of estimation is indeed invariant under non-linear transformation of parameters and data. However, this method does not generalise well to models containing multiple parameters, and can return estimates from the region of the posterior distribution that have low probability.

Information-Theoretic Methods of Point Estimation

A complementary view to Bayesian point estimation and model selection can be derived using the notion of information. In the Bayesian framework, this involves replacing probabilities with the measure of information content. As seen in Section 3.2, the measure of information varies according to the probability. The landmark work of Shannon (1948) showed that the Shannon information content (Section 3.2; Equation 3.1) can be defined as the length of the shortest (optimal) code required to uniquely and losslessly state (*communicate, describe, explain*) an event, e , with a probability $\Pr(e)$ as:

$$I(e) = -\log(\Pr(e))$$

In the nineteen sixties several researchers independently proposed connections between information theory and statistical inference (Solomonoff, 1964; Kolmogorov, 1963; Wallace and Boulton, 1968; Chaitin, 1966). The Minimum Message Length principle (Wallace and Boulton, 1968) gave the first practical demonstration of its application to statistical inference. Wallace and colleagues subsequent work gave rise to a mature branch of statistical inference relying on Shannon's measure of information (Wallace and Boulton, 1968, 1969; Boulton and Wallace, 1973; Wallace and Boulton, 1975; Rissanen, 1978; Wallace and Freeman, 1987, 1992; Allison et al., 1992; Wallace and Patrick, 1993; Dowe et al., 1996; Wallace, 1998; Wallace and Dowe, 1999; Farr and Wallace, 2002).

[†]For instance, this thesis uses set of points that can be expressed as Cartesian coordinates (x, y, z) or equivalently as Spherical coordinates (r, θ, ϕ) . One representation is a non-linear and invertible transformation of the other.

The MML principle provides the basic framework for the research on the structural alignment problem presented in this thesis. The fundamental ideas and methodologies behind the MML principle are briefly explained below. However, for a comprehensive treatment of this topic, refer to Wallace (2005).

3.3.3 Minimum Message Length Inference

Wallace and Boulton (1968) developed the first information theoretic criterion for inductive inference and model selection. Minimum Message Length (MML) is a Bayesian framework that links information theory (Shannon, 1948) and lossless data compression to Bayesian statistics (Bayes and Price, 1763), and provides a practical and reliable way to discriminate between competing models (or hypotheses), and to select good model(s) (and estimate their parameters) that explain the observed data.

The MML principle posits that the best model \mathcal{H} of the observed data \mathcal{D} is the one that can explain (state) \mathcal{D} in the **shortest message length**. This can be seen from an information-theoretic restatement of Bayes theorem in Equation 3.3 above. Applying the notion of Shannon information content (Equation 3.1) to Bayes theorem (Equation 3.3) arrives at:

$$I(\mathcal{H}, \mathcal{D}) = \underbrace{I(\mathcal{H})}_{\text{model-part}} + \underbrace{I(\mathcal{D}|\mathcal{H})}_{\text{data-part}} \quad (3.4)$$

Thus, as a general principle, model selection under the MML framework finding the model that minimises the two-part message length. The first part is the statement of the hypothesis, \mathcal{H} , which is used to describe the observed data, \mathcal{D} . The second part is the statement of the observed data assuming \mathcal{H} to be true.

MML Inference as a Hypothetical Communication Process

MML is best understood as a communication process between a hypothetical transmitter (*Alice*) and receiver (*Bob*) connected over a shannon channel. Alice wishes to encode and send the \mathcal{D} succinctly in such a way that Bob can reconstruct it exactly as Alice observes it. Alice and Bob agree on a *codebook*, a rulebook of communication protocols they both agree on. Alice must then choose a \mathcal{H} based on the data. Once this is done, Alice can encode and transmit the data over a two-part message: in the first, she encodes and transmits the hypothesis to Bob, while in the second she encodes and transmits the data *given* the hypothesis and then transmits it. The two-part message received on Bob's side should be decodable so that the data can be recovered without loss. The goal for Alice is to choose a hypothesis such that the transmission of the data over a two-part message results in the *shortest possible message* over the entire space of hypotheses.

While this communication framework deals with encoding and decoding, in practice, no information is actually transmitted. MML inference deals only with measures of information and not with the actual mechanics of encoding and decoding. Therefore, MML inference is only concerned with the Shannon information content of various terms in the two-part message.

Hypothesis Complexity-versus-Fit Trade-off

Any hypothesis has a certain descriptive complexity. A complex hypothesis (one with more free parameters) can predict (*fit, explain*) a greater variety of observed data than a simpler

hypothesis (one with fewer free parameters). Therefore, in order to choose the best hypothesis for any inference problem, one is confronted with a *trade-off* between hypothesis complexity and how well it predicts the observations.

The MML principle naturally balances hypothesis complexity with the ability of the hypothesis to accurately describe the data. While explaining a complex hypothesis requires a long $I(\mathcal{H})$ message, such a hypothesis may be able to describe (*fit*) the data more concisely, decreasing the length of the $I(\mathcal{D}|\mathcal{H})$ message required to describe the data. Alternatively, asserting a simple hypothesis requires a shorter message $I(\mathcal{H})$, but if it is poor at explaining the data, might increase the length of the message $I(\mathcal{D}|\mathcal{H})$ for stating the data. This trade-off between hypothesis complexity and how well the hypothesis fits the data is perfectly captured by MML and clearly mirrors the trade-off between coverage and fidelity required to solve the protein structural alignment problem (see Section 2.2.5).

Precision of Statement Issues

The MML principle has been explored above in very broad terms, without exposing the reader to many important, subtle and practical issues that arise from its seemingly simple formulation. Specifically, the lossless communication of a message requires some precision of statement for the terms involved in the message. Further, the two-part message involves the statement of statistical parameters (supporting the hypothesis) and the encoding of the data given the parameters. Both of these (as is normally the case) involve the transmission of real-valued entities that have also to be stated to a certain precision (otherwise, the message length can be arbitrarily long if the transmission involves arbitrarily precise real numbers).

For most statistical inference problems, the precision of statement of data (or precision of measurement) is a property of the data.

Precision of Measurement (PoM)

All continuous data has some amount of measurement error. That is, no continuous data can be stated *exactly* to infinite precision. ϵ denotes the precision to which the data can be measured, a value that is a property of the data.

On the other hand, the precision of statement of parameters (or precision of parameter values) must be inferred from the data.

Precision of parameter values (PoPV)

It is important within the MML framework for the chosen model to have all of its parameters stated (the model must be *fully parameterised*). The question then becomes to what precision those parameters should be stated, since Bob does not know the parameters: that is, to what PoPV will Alice need to state the parameters in order to minimise the two part message length.

Therefore, in addition to finding the right model, the MML framework simultaneously needs to address the problem of finding the PoPV. This precision has an important implication to the two-part message length (see Equation 3.4): as the precision of statement of parameters becomes coarse, the first part of the message (parameter-part) shortens, while the second part of the message (data-part) typically grows, because the coarsely specified parameters do not explain the data so well. Here arises the non-trivial problem of finding the *optimal* precision

to which parameters must be stated, that most applications of MML must handle (MacKay, 2003).

The following sections explore the details of model selection in the MML framework, in addition to the treatment of precision of parameter values. This leads to the strict formulation of MML inference.

Strict Minimum Message Length Inference

Consider the set, X , of all possible data. Alice observes some subset $x \in X$ which she wishes to state over a message. One approach to addressing Minimum Message Length inference involves discretising the data space X and mapping each observed data $x \in X$ to a value $\hat{\theta}_{MML} = \hat{\Theta}(x) \in \Theta$, which forms the MML estimate of x (Wallace and Boulton, 1975; Wallace and Freeman, 1987; Farr and Wallace, 2002).[‡]

To establish this discretisation, define the marginal probability of $\hat{\theta}_{MML}$ computed in terms of the sum of marginal probability of a subset of x 's in X as $s(\hat{\theta}_{MML}) = \sum_{\forall x \text{ s.t. } \hat{\Theta}(x)=\hat{\theta}_{MML}} r(x)$. Using this, the two-part message to state some $x \in X$ is as follows: the first part transmits the hypothesis in $I(\hat{\theta}_{MML}) = -\log(s(\hat{\theta}_{MML}))$ bits; the second part transmits the data given the hypothesis in $I(x|\hat{\theta}_{MML}) = -\log(f(x|\hat{\theta}_{MML}))$ bits. However, this two-part message cannot be decoded by the receiver (Bob) since the encoding of the model parameters in the first part depends on the specific data x that is being encoded in the second part. To work around this, the strict minimum message length (SMML) inference aims to minimise, instead, the *expectation* of the two-part message length, over all $x \in X$ (Farr and Wallace, 2002):

$$I_{SMML}(\hat{\theta}, x) = - \sum_{\hat{\theta} \in \hat{\Theta}} s(\hat{\theta}) \log(s(\hat{\theta})) - \sum_{x \in X} r(x) \log(f(x|\hat{\theta})).$$

Thus, the goal of SMML inference is finding the mapping or discretisation $\hat{\Theta} : X \rightarrow \Theta$ that minimises the above expected message length. The algorithmic complexity issues of undertaking this discretisation are handled in Farr and Wallace (2002), which proves that SMML inference is, in general, an NP-hard problem. Hence, this strict formulation of MML on practical inference problems becomes intractable. Nevertheless, very good approximations of SMML estimates have been previously proposed (Wallace and Boulton, 1968; Wallace and Freeman, 1987) and are briefly described in the subsequent section.

Wallace-Freeman Approximation of SMML

Wallace and Freeman (1987) give a quadratic approximation of SMML inference. This is shown below for a model with a vector of parameters (Oliver and Baxter, 1994). Following this derivation, several approximations used to simplify the MML two part message length computation are introduced. Finally, a derivation for a specific model is undertaken after this as a motivating example.

Consider N observations of identical and independently distributed (i.i.d.) data, $x \in X$, each with a precision of measurement (PoM) of ϵ , being modeled using a statistical distribution with a d -dimensional parameter vector $\vec{\theta}$. Beginning with the general MML statement for the two-part message length for x using $\vec{\theta}$:

$$I(\vec{\theta}, x) = I(\vec{\theta}) + I(x|\vec{\theta}) \tag{3.5}$$

[‡]Note that since the data space is being discretised, this mapping need not be one-to-one between X and Θ . Often, multiple subsets of data are mapped to a single estimate, $\hat{\theta}_{MML}$.

Unlike SMML inference where it is required to discretise the data space X , the approximation of Wallace and Freeman (1987) considers a discretisation of the parameter space Θ .

Let the precision of the parameters values (PoPV) of $\vec{\theta}$ describe a discretised region $\mathcal{V}(\vec{\theta})$ of imprecision whose volume is given by $V(\vec{\theta})$. Then, considering the message length of the first part, $I(\vec{\theta})$, the Wallace and Freeman (1987) approximation assumes that the prior density function $h(\vec{\theta})$ is constant within the volume of imprecision of the parameters, and is represented as:

$$I(\vec{\theta}) = -\log(\Pr(\vec{\theta})) = -\log(V(\vec{\theta})h(\vec{\theta})) \quad (3.6)$$

Expanding the set of notation introduced in Section 3.3.1, let the *negative* log likelihood function be defined as $\mathcal{L}(\vec{\theta}) = -\log f(x|\vec{\theta})$. Furthermore, to compute the second part of the message $I(x|\vec{\theta})$, consider its expectation by integrating the likelihood $f(x|\vec{\theta})$ in the region corresponding to $V(\vec{\theta})$:

$$\mathbb{E}[I(x|\vec{\theta})] = \mathbb{E}[-\log(\Pr(x|\vec{\theta}))] = \frac{1}{V(\vec{\theta})} \int_{\mathcal{V}(\vec{\theta})} [\mathcal{L}(\vec{\theta} + \vec{\vartheta}) - N \log \epsilon] dv$$

where $\vec{\theta} + \vec{\vartheta}$ are the set of all parameter values defined by the discretised region $\mathcal{V}(\vec{\theta})$.

The integral above can be approximated by considering the Taylor series expansion of the negative log likelihood function up to the quadratic term:

$$\begin{aligned} \mathbb{E}[I(x|\vec{\theta})] &\approx -N \log \epsilon + \frac{1}{V(\vec{\theta})} \int_{\mathcal{V}(\vec{\theta})} \left(\mathcal{L}(\vec{\theta}) + \vec{\vartheta} \frac{\partial \mathcal{L}(\vec{\theta})}{\partial \vec{\theta}} + \frac{1}{2} \vec{\vartheta}^\top \frac{\partial^2 \mathcal{L}(\vec{\theta})}{\partial \vec{\theta} \partial \vec{\theta}^\top} \vec{\vartheta} \right) dv \\ &\approx -N \log \epsilon + \frac{1}{V(\vec{\theta})} \left(\int_{\mathcal{V}(\vec{\theta})} \mathcal{L}(\vec{\theta}) dv + \int_{\mathcal{V}(\vec{\theta})} \vec{\vartheta} \frac{\partial \mathcal{L}(\vec{\theta})}{\partial \vec{\theta}} dv + \frac{1}{2} \int_{\mathcal{V}(\vec{\theta})} \vec{\vartheta}^\top \frac{\partial^2 \mathcal{L}(\vec{\theta})}{\partial \vec{\theta} \partial \vec{\theta}^\top} \vec{\vartheta} dv \right) \\ &\approx -N \log \epsilon + \mathcal{L}(\vec{\theta}) + \frac{1}{2V(\vec{\theta})} \int_{\mathcal{V}(\vec{\theta})} \vec{\vartheta}^\top \frac{\partial^2 \mathcal{L}(\vec{\theta})}{\partial \vec{\theta} \partial \vec{\theta}^\top} \vec{\vartheta} dv \end{aligned} \quad (3.7)$$

Since:

$$\frac{1}{V(\vec{\theta})} \int_{\mathcal{V}(\vec{\theta})} \mathcal{L}(\vec{\theta}) dv = \mathcal{L}(\vec{\theta}) \quad \text{and} \quad \frac{1}{V(\vec{\theta})} \int_{\mathcal{V}(\vec{\theta})} \vec{\vartheta} \frac{\partial \mathcal{L}(\vec{\theta})}{\partial \vec{\theta}} dv = 0$$

Substituting Equations 3.6 and 3.7 into Equation 3.5 gives:

$$I(\vec{\theta}, x) \approx -\log(V(\vec{\theta})h(\vec{\theta})) - N \log \epsilon + \mathcal{L}(\vec{\theta}) + \frac{1}{2V(\vec{\theta})} \int_{\mathcal{V}(\vec{\theta})} \vec{\vartheta}^\top \frac{\partial^2 \mathcal{L}(\vec{\theta})}{\partial \vec{\theta} \partial \vec{\theta}^\top} \vec{\vartheta} dv \quad (3.8)$$

where $\frac{\partial^2 \mathcal{L}(\vec{\theta})}{\partial \vec{\theta} \partial \vec{\theta}^\top}$ is the matrix of second order differentials of $\mathcal{L}(\vec{\theta})$ which is also known as the *observed* Fisher information matrix (Casella and Berger, 2001). However, this matrix is dependent on observed data of which the receiver has no knowledge. To avoid an infinite regress of deciding the precision of the parameters based on the data, replace the *observed* Fisher with the *expected* Fisher, $\mathcal{F}(\vec{\theta})$, as per Equation 3.9, thus making the message decodable, as $V(\vec{\theta})$ will be independent of the observed data.

$$\mathcal{F}(\vec{\theta}) = \frac{\partial^2 \mathcal{L}(\vec{\theta})}{\partial \vec{\theta} \partial \vec{\theta}^\top} \approx \mathbb{E} \left[\frac{\partial^2 \mathcal{L}(\vec{\theta})}{\partial \vec{\theta} \partial \vec{\theta}^\top} \right] \quad (3.9)$$

This gives:

$$I(\vec{\theta}, x) \approx -\log(V(\vec{\theta})h(\vec{\theta})) - N \log \epsilon + \mathcal{L}(\vec{\theta}) + \frac{1}{2V(\vec{\theta})} \int_{\mathcal{V}(\vec{\theta})} \vec{\vartheta}^\top \mathcal{F}(\vec{\theta}) \vec{\vartheta} d\mathbf{v}$$

Since $\mathcal{F}(\vec{\theta})$ is a real square symmetric matrix, its eigen decomposition is given as: $\mathcal{F}(\vec{\theta}) = S\Lambda S^\top$, where Λ is a $d \times d$ diagonal matrix of eigenvalues. Note that $\Lambda^\top = \Lambda$. Therefore, the integrand in the above message length equation can be simplified by defining the following transformation:

$$\begin{aligned} \vec{\vartheta}^\top \mathcal{F}(\vec{\theta}) \vec{\vartheta} &= \vec{\vartheta}^\top S\Lambda S^\top \vec{\vartheta} \\ &= B\Lambda B^\top, \text{ where } B = \vec{\vartheta}^\top S \\ &= B\Lambda^{1/2}\Lambda^{1/2}B^\top, \text{ where } \Lambda^{1/2}\Lambda^{1/2} = \Lambda \\ &= y^\top y, \text{ where } y = \Lambda^{1/2}B^\top \end{aligned}$$

Therefore, y in terms of $\vec{\vartheta}$ is:

$$y = \Lambda^{1/2}S^\top \vec{\vartheta} \quad (3.10)$$

Let $p(\vec{\phi})$ be the transformed prior density of $h(\vec{\theta})$, and let $V'(\vec{\phi})$ be the transformed volume of uncertainty (imprecision) associated with $V(\vec{\theta})$:

$$p(\vec{\phi}) = h(\vec{\theta}) \frac{dV(\vec{\theta})}{dV'(\vec{\phi})} \quad (3.11)$$

Thus, in ϕ -space, the message length becomes:

$$I(\vec{\theta}, x) \approx -\log(V'(\vec{\phi})p(\vec{\phi})) - N \log \epsilon + \mathcal{L}(\vec{\theta}) + \frac{1}{2}\mathbb{E}[y^\top y] \quad (3.12)$$

where the model has two parameters, the expected value of $y^\top y$ can be expressed in terms of the d -dimensional optimal quantising lattice constant (Conway and Sloane, 1984), κ_d , as in the following:

$$\mathbb{E}(y^\top y) = d\kappa_d V'(\vec{\phi})^{\frac{2}{d}} \quad (3.13)$$

Substituting Equation 3.13 into the expression for the message length gives:

$$I(\vec{\theta}, x) \approx -\log(V'(\vec{\phi})p(\vec{\phi})) - N \log \epsilon + \mathcal{L}(\vec{\theta}) + \frac{d}{2}\kappa_d V'(\vec{\phi})^{\frac{2}{d}} \quad (3.14)$$

At this point it is necessary to find the optimal discretisation of the prior distribution, or PoPV ($V(\vec{\theta})$) that minimises $I(\vec{\theta}, x)$. This is achieved by differentiating Equation 3.14 with respect to $V'(\vec{\phi})$ and setting it to zero:

$$\frac{\partial I(\vec{\theta}, x)}{\partial V'(\vec{\phi})} = -\frac{1}{V'(\vec{\phi})} + \kappa_d V'(\vec{\phi})^{\frac{2}{d}-1} = 0$$

This can be solved as:

$$\begin{aligned} \frac{1}{V'(\vec{\phi})} &= \kappa_d V'(\vec{\phi})^{\frac{2}{d}-1} \\ V'(\vec{\phi}) &= \kappa_d^{-\frac{d}{2}} \end{aligned}$$

Substituting this optimal value for $V'(\vec{\phi})$ into Equation 3.13 gives:

$$\mathbb{E}(y^\top y) = d$$

Thus, from Equation 3.12, the message length expression becomes:

$$I(\vec{\theta}, x) \approx -\log(V'(\vec{\phi})p(\vec{\phi})) - N \log \epsilon + \mathcal{L}(\vec{\theta}) + \frac{d}{2}$$

From Equation 3.10, $V'(\vec{\phi})$ can be expressed in terms of $V(\vec{\theta})$:

$$\begin{aligned} V'(\vec{\phi}) &= \text{Jacobian}(\Lambda^{1/2} S^\top) V(\vec{\theta}) = \text{Jacobian}(\Lambda^{1/2}) V(\vec{\theta}) \\ &= \prod_{i=1}^d \sqrt{\lambda_i} V(\vec{\theta}) = \sqrt{\prod_{i=1}^d \lambda_i} V(\vec{\theta}) \\ &= \sqrt{\det(\mathcal{F}(\vec{\theta}))} V(\vec{\theta}) \end{aligned}$$

Therefore:

$$\frac{dV(\vec{\theta})}{dV'(\vec{\phi})} = \frac{1}{\sqrt{\det(\mathcal{F}(\vec{\theta}))}}$$

which, substituted into Equation 3.11, gives:

$$p(\vec{\phi}) = \frac{h(\vec{\theta})}{\sqrt{\det(\mathcal{F}(\vec{\theta}))}}$$

Finally, substituting this and the optimal value of $V'(\vec{\phi})$ into the expression for the message length gives the Wallace-Freeman approximation for the message length with two parameters:

$$I(\vec{\theta}, x) \approx \underbrace{\frac{d}{2} \log(\kappa_d) - \log(h(\vec{\theta})) + \frac{1}{2} \log(\det(\mathcal{F}(\vec{\theta})))}_{\text{first part: } I(\vec{\theta})} - \underbrace{N \log \epsilon + \mathcal{L}(\vec{\theta}) + \frac{d}{2}}_{\text{second part: } I(x|\vec{\theta})} \quad (3.15)$$

The Wallace and Freeman (1987) MML estimate is, therefore, given by minimising Equation 3.15.

Derivation of Wallace and Freeman (1987) Approximation of Parameters for the Normal Distribution

This section reproduces, from Wallace (2005), the derivation of the Wallace-Freeman approximation of data modelled using a Normal distribution with unknown parameters for mean, μ , and standard deviation, σ . Let the data be $\vec{x} = x_1, x_2, \dots, x_N$, and likelihood function be:

$$f(\vec{x}|\mu, \sigma) = \prod_{i=1}^N \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\sum_{i=1}^N \frac{(x_i - \mu)^2}{2\sigma^2}\right)$$

Assume that each datum is measured to the precision (PoM) of ϵ and $\epsilon \ll \sigma$. The negative log likelihood function is:

$$\mathcal{L}(\mu, \sigma) = -\log f(\vec{x}|\mu, \sigma) = \frac{N}{2} \log(2\pi) + N \log \sigma + \sum_{i=1}^N \frac{(x_i - \mu)^2}{2\sigma^2}$$

The first and second derivatives of the negative log likelihood function with respect to the parameters are:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mu} &= -\sum_{i=1}^N \frac{(x_i - \mu)}{\sigma^2} \\ \frac{\partial^2 \mathcal{L}}{\partial \mu^2} &= \frac{N}{\sigma^2} \\ \frac{\partial \mathcal{L}}{\partial \sigma} &= \frac{N}{\sigma} - 1/\sigma^3 \sum_{i=1}^N (x_i - \mu)^2 \\ \frac{\partial^2 \mathcal{L}}{\partial \sigma^2} &= -\frac{N}{\sigma^2} + \frac{3}{\sigma^4} \sum_{i=1}^N (x_i - \mu)^2 \\ \frac{\partial^2 \mathcal{L}}{\partial \mu \partial \sigma} &= (1/\sigma^2) \sum_{i=1}^N (x_i - \mu) \end{aligned}$$

For the Normal distribution, $\mathbb{E}[x_i - \mu]$ and $\mathbb{E}[(x_i - \mu)^2]$ are 0 and σ^2 respectively. Thus,

$$\begin{aligned} \mathbb{E} \left[\frac{\partial^2 \mathcal{L}}{\partial \sigma^2} \right] &= -\frac{N}{\sigma^2} + (\sigma 3\sigma^4) N \sigma^2 = 2\frac{N}{\sigma^2} \\ \mathbb{E} \left[\frac{\partial^2 \mathcal{L}}{\partial \mu \partial \sigma} \right] &= 0 \\ \mathcal{F}(\mu, \sigma) &= \begin{bmatrix} \frac{N}{\sigma^2} & 0 \\ 0 & \frac{2N}{\sigma^2} \end{bmatrix} \\ \det(\mathcal{F}(\mu, \sigma)) &= \frac{2N^2}{\sigma^4} \end{aligned}$$

Substituting this and the above expression for \mathcal{L} into Equation 3.15 gives:

$$\begin{aligned} I((\mu, \sigma), \vec{x}) &= \log(\kappa_2) - \log(h(\mu, \sigma)) + \frac{1}{2} \log \left(\frac{2N^2}{\sigma^4} \right) + \frac{N}{2} \log(2\pi) \\ &\quad + N \log \sigma - N \log \epsilon + \sum_{i=1}^N \frac{(x_i - \mu)^2}{2\sigma^2} + 1 \quad (3.16) \end{aligned}$$

Assuming that μ and σ have independent priors. With little prior knowledge, assume there is no reason to prefer any particular location for the distribution and, therefore, the prior density for μ is uniform within a range, R_μ . Furthermore, assume a bland prior on σ that expects σ to be small, with a density proportional to $\frac{1}{\sigma}$. That is, $\log \sigma$ has a uniform prior density over a range, R_σ . In this scenario:

$$h(\mu, \sigma) = \left(\frac{1}{R_\mu} \right) \left(\frac{1}{\sigma R_\sigma} \right)$$

Thus,

$$\begin{aligned} -\log(h(\mu, \sigma)) &= -\log\left(\left(\frac{1}{R_\mu}\right)\left(\frac{1}{\sigma R_\sigma}\right)\right) \\ &= \log R_\mu + \log \sigma R_\sigma \\ &= \log \sigma + \log R_\mu R_\sigma \end{aligned}$$

Which, substituted into Equation 3.16 above and letting $v^2 = \sum_n (x_n - \mu)^2$, gives:

$$I((\mu, \sigma), \vec{x}) = \log(\kappa_2) + \log \sigma + \log R_\mu R_\sigma + \frac{1}{2} \log\left(\frac{2N^2}{\sigma^4}\right) + \frac{N}{2} \log(2\pi) + N \log \sigma - N \log \epsilon + \frac{v^2}{2\sigma^2} + 1$$

Note that the *unbiased* estimates for the mean and standard deviation are: $\mu' = \frac{1}{N} \sum_n x_n$, and $\sigma' = \sqrt{\frac{v^2}{(N-1)}}$. Which, substituted into the above equation gives:

$$\begin{aligned} I((\mu, \sigma), \vec{x}) &= \log(\kappa_2) + \log R_\mu R_\sigma + \frac{1}{2} \log(2N^2) + \frac{1}{2}(N-1) \log\left(\frac{v^2}{N-1}\right) \\ &\quad + \frac{N}{2} \log\left(\frac{2\pi}{\epsilon^2}\right) + \frac{N-1}{2} + 1 \end{aligned} \quad (3.17)$$

Equation 3.17 is the expression for the message length of N data points modeled by a normal distribution using the Wallace-Freeman approximation for uniform priors. The MML estimates for μ and σ that minimise $I((\mu, \sigma), \vec{x})$ correspond to the solutions of $\frac{\partial I}{\partial \mu} = 0$ and $\frac{\partial I}{\partial \sigma} = 0$. These are:

$$\hat{\mu}_{MML} = \frac{1}{N} \sum_{i=1}^N x_i \quad \text{and} \quad \hat{\sigma}_{MML} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \hat{\mu})^2$$

MML Inference When an Explicit Fisher Information Matrix for a Hypothesis Cannot be Defined

The previous section described how model parameters can be estimated using the quadratic approximation method of Wallace and Freeman (1987). This method is directly applicable when the hypotheses are backed by statistical models with real-valued parameters. However, not all statistical inference problems define hypotheses of this kind. For instance, consider the problem of hierarchical classification of N items with K attributes. A hierarchical classification hypothesis will have to specify (at least) the following five pieces of information (Wallace and Boulton, 1968):[§]

1. The number of classes.
2. The hierarchical tree of these classes.
3. The model (or distribution function) for each class in the classification.
4. The class to which each of the N items belong (and this could be either a deterministic or a probabilistic membership depending on the type of the classification problem).

[§]Recall from Section 3.2, that information is additive.

5. The deviations of each item from its assigned class in the proposed/hypothesised classification.

Comparing these enumerated items against the MML's two-part message framework (see Equation 3.4), one can conclude that the first three items form the first part (model-part) of the message, while the last two items contribute to the second part (data-part) of the message. It may be true that each item corresponding to the classification problem (or any inference problem in general) can individually be supported by statistical distributions (with unknown parameters) and, hence, their expected Fisher information matrix may be computable and applied only individually for each item (information term). However, finding the Fisher information collectively for all the terms involved in the hypothesis can be intractable, if not impossible, in some cases. Therefore, in such cases, the MML objective is framed to minimise the two-part message by individually accounting for the message length terms of all its constituent pieces of information (Wallace and Boulton, 1968). Therefore, the MML inference in such cases is done by breaking down each of the two-part message into individual terms (message blocks), whose Shannon information content are computed separately, under the objective that the total (as sum-of-the-parts) message length is minimised. **The problem of structural alignment considered in this thesis falls into this category.**

3.4 Codewords, Prefix-Free Codes and Universal Codes for Integers

While the MML framework requires only the measure of information of various terms (and does not actually require any encoding or decoding scheme), for completeness, this introduction of MML is concluded by providing a brief overview of codes and encoding.

Over a binary Shannon channel, each message length term is a long sequence of bits containing many codewords. Each codeword is bit string (often of variable length) containing an encoded piece of information that the sender (Alice) wants to transmit, losslessly, to the receiver (Bob). The length of a codeword is simply the number of bits that go into forming that codeword. Therefore, a lossless message is a concatenation of a set of codewords that is decodable at the receiver's end.

Information theorists and computer scientists are interested in constructing codewords for a set of possible symbols generated from some source that is producing these symbols. Each codeword should exhibit the following properties:

1. Uniquely encode a symbol,
2. Contain enough information so that the encoded symbol is decodable, and
3. Be non-redundant so that the code is efficient (or contain more information than is required).

As seen in Section 3.2, theoretically, the shortest lossless code for a symbol is given by the negative logarithm of the probability of that symbol. In practice, the length of the codewords vary according to the entropy in many useful codes. Such codes are not only of variable-length (dictated by the statistical model supporting the encoding of the symbols) but also *prefix-free*. A prefix-free code has the property that no two codewords share a common prefix bit string.[¶]

[¶]Prefix-free codes are uniquely decodable, but not all decodable codes are prefix-free.

The Huffmann code is an optimal (non-redundant), variable-length, prefix-free code (Huffman, 1952). The most common use of prefix-free codes is for encoding integers (or any countable set of things over an infinite range).

Provided the range of values to be expressed is known and each value has a uniform probability, the optimal code length to state any integer in the range is simply the logarithm of the size of the range. However, if the range of values is not known, an alternative method is required. Elias (1975); Rissanen (1983) and Wallace and Patrick (1993) give methods to state positive integers from an unknown range. Each essentially chooses a variable length integer representation and transmits it. However, since the range is unknown, the length of the value being transmitted should be transmitted first, and the length of that length and so on. This regression, however, decreases in size very rapidly as it takes the form:

$$\log^*(n) = \log(n) + \log(\log(n)) + \dots$$

for all positive terms, where n is the size of the original integer representation to transmit. Rissanen (1983) gives:

$$I_{\text{integer}}(n) = \log^*(n) + \log_2(2.865) \quad (3.18)$$

where 2.865 is a normalising constant such that the sum of all values from the distribution is 1: $\sum_{n=1}^{\infty} 2^{-I_{\text{integer}}(n)} = 1.0$. In some cases, when using this encoding scheme, the range of integers that must be encoded begins at zero: $0 \leq n \leq \infty$. In this case, $I_{\text{integer}}(n)$ is defined as: $I_{\text{integer}}(n) = \log^*(n + 1) + \log_2(2.865)$. **This method of variable-length encoding for integers is used extensively throughout this thesis.**

3.5 Summary

Statistical inference is the process of selecting a hypothesis based on some observed data, facilitating the prediction of unobserved data. In this context, a hypothesis is a fully parameterised statistical model. The parameters of the statistical model are estimated using an estimator that should have at least the three key properties of: invariance to reparameterisation, invariance to transformation of the data space, and zero bias. Fitting these criteria, The Minimum Message Length (MML) framework for inductive inference provides a practical, reliable, and objective method for model selection. MML does this by linking Bayesian statistics and information theory with data compression. The next chapter will present a method for applying MML to the problem of selecting the best protein structural alignment hypothesis. This method is grounded in rigorous statistical methods and lossless data compression.

Chapter 4

A Framework for Assessing Alignment Quality Using Information Theory

“We can only see a short distance ahead, but we can see plenty there that needs to be done.”

— A. M. Turing (1950)

This chapter presents an MML based framework for assessing protein structural alignment quality. This framework treats a structural alignment as an instance of the general class of inference problems, where an alignment is a hypothesis explaining the structural relationship between two proteins. Each alignment hypothesis is seen as an attempt to explain the coordinate data of the pair of proteins. The explanatory power of each alignment is then quantified, using principles of information theory, as the amount of *lossless compression* obtained from encoding the coordinate data of the proteins using the knowledge of the correspondences provided by the alignment. This framework is then benchmarked, using a large set of SCOP domain pairs, against other popular structural alignment scoring functions. Additionally, a set of alternative alignments suggested by Zu-Kang and Sippl (1996), which are ambiguous in terms of RMSD and number of correspondences, are examined in detail. Finally, the level of agreement between scoring functions on the SCOP data is quantified and found to be minimal, except when the scoring functions are closely related. This confirms the findings of Kolodny et al. (2005); Hasegawa and Holm (2009); Sippl and Wiederstein (2008); Slater et al. (2013) and Ma and Wang (2014) which indicate that the current state-of-the-art structural alignment programs produce contradictory results.

This chapter is based on the paper: Collier, J. H., Allison, L., Lesk, A. M., Garcia de la Banda, M., Konagurthu A. S. (2014). A new statistical framework to assess structural alignment quality using information compression, *Bioinformatics* **30**(17): i512–i518.

4.1 Introduction

As introduced in Section 2.2, a pairwise alignment is an order-preserving assignment of one-to-one correspondences between the amino acids of two proteins. While alignments can be computed based on sequence information alone, structural alignments (which are computed based on the conformational similarity) are generally accepted to be more reliable, since structure changes more conservatively than sequence in the evolution of protein domains (Richardson, 1981; Chothia and Lesk, 1986; Murzin, 1998; Edwards and Deane, 2015). Indeed, structural alignments are often used as the standard by which to judge the quality of sequence alignments (Mizuguchi et al., 1998; Walle et al., 2005; Edgar, 2010).

The problem of aligning protein *sequences* is very well understood: many rigorous statistical models have been proposed to quantitatively assess sequence alignment quality (Karlin and Altschul, 1990; Altschul, 1991; Allison et al., 1992). This has, in turn, helped standardise the task of measuring sequence alignment quality and, thus, the task of generating meaningful sequence alignments. In fact, it has been argued that newer sequence alignment methods yield diminishing (if any) improvements, in so far as an evolutionary relationship can be inferred from sequence information alone (Edgar, 2004).

The mature statistical basis of sequence alignments stands in stark contrast to the current state of the art for structural alignment methods. The last four decades have seen the development of many methods aimed at producing biologically meaningful structural alignments, with the number of new methods estimated to be doubling every five years (Hasegawa and Holm, 2009). Several comparative studies have observed many inconsistencies and paradoxes when comparing the alignments generated by existing methods. Noteworthy among these studies are those by Kolodny et al. (2005); Hasegawa and Holm (2009); Sippl and Wiederstein (2008); Slater et al. (2013); and Ma and Wang (2014). A common theme emerging from all of these studies is the need for a systematic framework to assess the quality of structural alignments. While a handful of quantitatively rigorous statistical models for structure comparison have been proposed for this (for example, see Levitt and Gerstein (1998)), there is no consensus regarding their usefulness.

Guided by good biological insights, current structural alignment methods define a *scoring function* to *quantify* the structural alignment quality. This has traditionally been achieved by combining the contributions of a small number of important criteria into an easy-to-compute scoring function, as seen in Section 2.2.5. Broadly, in their different manifestations, these current scoring functions use two key criteria: *coverage* and *fidelity*. Typically, coverage measures the number of correspondences in an alignment and, in some cases, also considers the number of gaps. Fidelity, measures how similarly positioned the aligned residues are. Fidelity is commonly (but not always) based on the root-mean-square deviation (RMSD) computed after the least-squares superposition of corresponding residues is found (see Section 2.2.3).

To search for the *best* structural alignment, the goal of alignment programs is to simultaneously maximise coverage and fidelity. However, these two objectives are in direct conflict with each other (Irving et al., 2001). Most of the proliferation of quality scores for protein structural alignments arise from attempts to reconcile this conflict, with scoring functions differing mainly in how they make the trade-off between these two criteria. This has led to a situation where existing scoring functions produce conflicting results, even when aligning structures that have only moderately diverged in evolution (Kolodny et al., 2005; Hasegawa and Holm, 2009; Slater et al., 2013). Since this traditional approach to formulating a scoring function has been explored extensively over the last four decades, further development along the same methodological lines is unlikely to provide any major breakthrough.

In this chapter, a radically new approach to assessing the quality of a protein structural alignment is proposed. This approach uses the information-theoretic Minimum Message Length (MML) (Wallace and Boulton, 1968; Wallace, 2005) criterion. This method has a statistically rigorous foundation and it overcomes the reliance of existing approaches on formulating an *ad hoc* trade-off between *coverage* and *fidelity*. This MML based method, as explained in Section 4.3, can make a natural and objective trade-off between these two objectives. This chapter is best contextualised using the background information for the structural alignment problem provided in Section 2.2.2 and that for MML inference provided in Section 3.3.

4.2 Review of Popular Alignment Quality Measures

This chapter uses nine of the most widely accepted structural alignment scoring functions for benchmarking: DALI **z-score** (Holm and Sander, 1993), GDT_TS and LGA_S3 (Zemla, 2003), MI and SI (Kleywegt and Jones, November 1994; Kolodny et al., 2005) SAS (Subbiah et al., 1993), GSAS (Kolodny et al., 2005), STRUCTAL_score (Subbiah et al., 1993; Gerstein and Levitt, 1998; Levitt and Gerstein, 1998), and TM-Score (Zhang and Skolnick, 2005b). While these scoring functions were introduced in Section 2.2.5, they are described in more detail below as this is necessary to fully analyse the results presented in Section 4.8. For some of the measures described below, the criteria for statistical significance is unclear. However, a method such as that used by Holm and Sander (1998) or Levitt and Gerstein (1998) to compute an empirical z-score is widely applicable to all alignment methods. As in Table 2.2, coverage and fidelity terms will be highlighted in blue and red respectively.

DALI_score introduced the concept of *elasticity* based on the alignment of distance matrices (see Section 2.2.4). The score is intended to compensate for plastic deformation in structures by using a relative intra-structure distance between C $_{\alpha}$ atoms, rather than the distances between corresponding residues after least-squares superposition. The formulation of the DALI_score is as follows:

$$\text{DALI_score} = \sum_{i=1}^{N_e} \sum_{j=1}^{N_e} \begin{cases} \Theta^E - \frac{\|d_{ij}^S - d_{ij}^T\|}{d_{ij}^*} e^{(d_{ij}^*/\alpha)^2}, & i \neq j \\ \Theta^E, & i = j \end{cases}$$

In this formulation, N_e is the number of correspondences, i and j are indexes of the i^{th} and the j^{th} pair of corresponding residues, d^S and d^T are the intra-protein distance matrices for proteins S and T , Θ^E is a similarity threshold, set to 0.2, allowing for deviations between adjacent matched secondary structural elements, and $\alpha = 20\text{\AA}$, is an empirical value for the size of a typical domain. Each matrix cell in d^S and d^T contains the intra-structural distance between the residues within a structure, where d_{ij}^* is the average of distance between the i^{th} and j^{th} corresponding C $_{\alpha}$ atoms in S and T : $d_{ij}^* = 0.5(d_{ij}^S + d_{ij}^T)$. The DALI_score evaluates coverage by summing the score over all corresponding pairs in S and T , and fidelity by taking the difference in intra-residue distances between the two structures.

The DALI program, which uses DALI_score to evaluate structural alignments, defines an empirical, length-independent **z-score** to evaluate the statistical significance of an alignment. Given a pair of protein structures, $\langle S, T \rangle$, with lengths, $|S|$ and $|T|$, a length term of $L =$

$\sqrt{|S| \times |T|}$ is first computed. Using this, the DALI z-score is defined as follows:

$$L' = 7.95 + 0.71L - 0.000259L^2 - 1.92 \times 10^{-6}L^3$$

$$\text{z-score} = \frac{\text{DALI_score} - L'}{0.5 \times L'}$$

An alignment between a pair of structures that share a similar fold or topology are usually assigned a DALI z-score greater than 2 (Holm and Sander, 1998). The z-score is measured as the number of standard deviations from the mean of the distribution of DALI_scores from an all-vs.-all comparison. This value is theoretically unbounded. As the DALI z-score is a transformation of the raw DALI_score and a measure of significance, it will be used as benchmark alignment quality measure instead of the raw DALI_score. Without accounting for the construction of the distance matrices, the DALI_score is a quadratic, $O(N_e^2)$, time operation.

STRUCTAL_score (Subbiah et al., 1993; Gerstein and Levitt, 1998; Levitt and Gerstein, 1998) was developed to outperform methods relying on intra-protein distance matrices such as DALI (above). The STRUCTAL_score measures fidelity as the distances, δ 's, between corresponding C_α atoms. Alignment coverage is indirectly accounted for by summing over all corresponding residues, and also more directly, by subtracting the number of gaps (N_g) from the score. The STRUCTAL_score has the following form:

$$\text{STRUCTAL_score} = \sum_{i=1}^{N_e} \frac{M}{1 + (\delta_i/d_0)^2} - 10N_g$$

In the above equation, N_e is the number of assigned correspondences, M is the maximum score given to any given correspondence which is empirically defined to be 20\AA , d_0 is a threshold distance which is set to 5\AA . The range of values from the STRUCTAL_score is unbounded. Though Levitt and Gerstein (1998) provide a method for computing a z-score (see Table 2.2), no threshold for similarity is set. Therefore it is unclear what values of the STRUCTAL_score constitute a statistically significant alignment. It is easy to see from its definition that the STRUCTAL_score is a linear, $O(N_e)$, time operation in the number of correspondences.

TM-Score (Template Modelling Score; Zhang and Skolnick (2004)) is a modification of the STRUCTAL_score. TM-Score attempts to remove the dependence on the length of the protein chains being aligned. It defines a normalisation constant, d_0 , as a *function* of the length of the largest structure: $d_0 = 1.24 \sqrt[3]{\max(|S|, |T|) - 15} - 1.8$. The constants are obtained empirically in order to ensure consistent length normalisation. The formulation is similar to that of the STRUCTAL_score above:

$$\text{TM-Score} = \frac{1}{\min(|S|, |T|)} \sum_{i=1}^{N_e} \frac{1}{1 + (\delta_i/d_0)^2}$$

In this equation, N_e is the number of assigned correspondences, δ_i is the distance between the i^{th} pair of corresponding C_α atoms after least-squares superposition. The authors of TM-Score do not clearly define a threshold for statistical significance (Xu and Zhang, 2010). However, a TM-Score larger than 0.3 is said to have a p-value of less than 0.001, and a TM-Score greater than 0.5 is said to come from alignments of structural pairs in the same fold (Xu and Zhang,

2010). The time complexity of **TM-Score** is linear in the number of equivalences: $O(N_e)$.

GDT_TS (Global Distance Test: Total Score; Zemla (2003)) is a score originally designed for use in measuring the quality of structures predicted from sequence. **GDT_TS** measures the average number (coverage) of well-fitting residues (fidelity) as a percentage of the total number of correspondences (N_e). ‘Well-fitting’ in this case means the corresponding pairs of C_α atoms with distances (after least-squares superposition) below a set of distinct pre-specified thresholds which are set to 1Å, 2Å, 4Å, and 8Å. There is an alternative definition of the global distance test (GDT) measure, with tighter distance thresholds, called **GDT high accuracy (GDT_HA)**. In the high accuracy version, the distance thresholds are halved to: 0.5Å, 1Å, 2Å, and 4Å. **GDT_TS** takes the following formulation:

$$\text{GDT_TS} = 100 \times \frac{\sum_{i=1}^4 \frac{N_{\delta_i \leq C_i}}{N_e}}{4}, \quad C_i \in \{1.0, 2.0, 4.0, 8.0\} \text{Å}$$

In the above equation $N_{\delta_i \leq C_i}$ is the number of correspondences superimposed with a distance below a threshold, $C_i \in \{1.0, 2.0, 4.0, 8.0\} \text{Å}$. The result is normalised such that the range of possible values is between 0 and 100. However, the criteria for determining the statistical significance for an alignment is unclear. **GDT_TS** requires a least-squares superposition which can be accomplished in $O(N_e)$ time, and the count of well-fitting correspondences takes $O(N_e)$ time. Therefore, **GDT_TS** is a linear, $O(N_e)$ time computation.

LGA_S3 (Local Global Alignment Score; Zemla (2003)) is also designed for measuring the quality of structures predicted from sequence. **LGA_S3** is a weighted combination of two other scoring functions. **LGA_S3** simply provides these two scoring functions parameters in the form of a set of increasing distance thresholds, D , for the distance between corresponding C_α atoms. **LGA_S3** then weights the results. In practise, this means that tighter (smaller) distance threshold parameters are given greater weight. The weighting function, W , takes the following form:

$$W(F, D) = \frac{\sum_{i=1}^{|D|} \frac{|D|-i+1}{|D|} F(D_i)}{(1 + |D|) \times |D|/2} \quad (4.1)$$

In this equation, F is one of the two scoring functions (discussed specifically below) used by **LGA_S3**, and D is a set of increasing distance thresholds. The final result is normalised based on the number of thresholds, $|D|$. The set of distance thresholds is iterated over, at each iteration a threshold is provided, as a parameter, to F and the result is weighted (between 0 and 1). The weighting becomes smaller for each iteration, as the thresholds become larger. For example, if D contains 10 thresholds, on the first iteration the result from F will be given the weight 1.0. On the 5th iteration the assigned weight will be 0.6, and on the final iteration the weight will be 0.1.

The two scoring functions used by **LGA_S3** are called **GDT** (Global Distance Test, which provides the global aspect of the score) and **LCS** (Longest Continuous Segments, which provides the local aspect of the score). The definition of **GDT** is similar to that of **GDT_TS** as described above. Instead of a set of threshold parameters, only one, t , is provided at a time:

$$\text{GDT}(t) = 100 \times \frac{N_{\delta_i \leq t}}{N_e}$$

LCS computes the longest set of contiguous correspondences (with intervening gaps) that superimpose below a given threshold. This can be defined mathematically by letting M be the set of all possible sets of contiguous correspondences. Then M_i is a particular set of contiguous correspondences, and M_{ij} is the j^{th} correspondence in the i^{th} contiguous set of correspondences. Using this notation, LCS can be defined as:

$$\text{LCS}(t) = \max(\{\|M_i\| \mid \forall j, M_{ij} < t\}) \quad \forall M_i \in M$$

In this equation, t is a threshold parameter. Sets of contiguous correspondences in M are selected if they match the condition that every correspondence in the set, M_i , is less than the threshold. The length of longest of the sets that match this condition is the result of the LCS scoring function.

Finally, LGA_S3 combines GDT and LCS, using the weighting function (W , see Equation 4.1 above) with a user defined weight, w , as follows:

$$\text{LGA_S3} = w \cdot W(\text{GDT}, \{0.5, 1.0, \dots, 10.0\}) + (1 - w) \cdot S(\text{LCS}, \{1.0, 2.0, 5.0\})$$

LGA_S3 first applies the weighting function, W , as defined above to the GDT scoring function with the set of thresholds, $C_i = \{0.5, 1.0, \dots, 10.0\}$. Then the weighting function is applied to the LCS scoring function with the set of thresholds, $C_i = \{1.0, 2.0, 5.0\}$. The results of these are summed according to the user defined weighting parameter, w , which has a range of $0 \leq w \leq 1$. The range of values from LGA_S3 is 0 to 100. However, the criteria for determining the statistical significance for an alignment is unclear.

The time complexity of LGA_S3 is bound by the computation time for LCS and GDT_TS. As discussed above, GDT_TS takes linear time, $O(N_e)$. The time complexity of LCS is quadratic in the number of correspondences, $O(N_e^2)$ because it needs to find all contiguous corresponding segments. There are a fixed constant number of thresholds to iterate over, therefore, the time complexity of LGA_S3 is quadratic in the number of correspondences, $O(N_e^2)$.

SAS (Structural Alignment Score; Subbiah et al. (1993)) and GSAS (Gapped Structural Alignment Score; Kolodny et al. (2005)) were devised as independent measures of alignment quality to be used for the benchmarking of many protein structural alignment programs. This benchmarking was carried out by Kolodny et al. (2005). The SAS makes a simple trade-off between the fidelity as measured by the RMSD after least-squares superposition, and the coverage as measured by the number of assigned correspondences as follows:

$$\text{SAS} = \frac{100 \times \text{RMSD}}{N_e}$$

GSAS extends SAS in order to account for gaps by subtracting the number of gaps (N_g) from the coverage term. In effect, this penalises gaps in the alignment. In the case where the number of gaps is larger than the number of matches, GSAS concludes that the alignment is poor and assigns it the worst possible score: 99.9. For all cases where $N_e \geq N_{\text{gaps}}$, GSAS is defined as:

$$\text{GSAS} = \frac{100 \times \text{RMSD}}{N_e - N_g}$$

The best possible alignment is given a *cost* of 0 from both SAS and GSAS, there is no upper bound for SAS. The worst possible score for GSAS is 99.9. However, the criteria for determining the statistical significance for an alignment is unclear. The time complexity for both SAS and

GSAS is linear in the number of correspondences: $O(N_e)$. This is due to the computation needed for a least-squares superposition (see Section 2.2.3).

MI and **SI** (Match Index and Similarity Index; Kleywegt and Jones (November 1994)) were devised for use by the **LSQMAN** (Kleywegt, 1996) alignment program. Along with **GSAS** above, these scores are also used as independent measures for the evaluation of protein structural alignment methods (Kolodny et al., 2005). **SI** has a similar formulation to **SAS** with the same coverage and fidelity terms:

$$\text{SI} = \frac{\min(|S|, |T|) \times \text{RMSD}}{N_e}$$

MI is a normalised score, taking values between 0 and 1. Following the notation of Kolodny et al. (2005), **MI** is defined to be $1 -$ the original version of the score. In this sense, values of **MI** closer to 0 indicate better alignments. The formulation takes the following form:

$$\text{MI} = 1 - \frac{1 + N_e}{(1 + w \times \text{RMSD})(1 + \min(|S|, |T|))}$$

In this equation, w is a user defined weighting parameter. Kleywegt and Jones (November 1994) suggest values for w in the range of 0.1 to 1. Kolodny et al. (2005) use the specific value of $w = 2/3$, and this value is also used throughout this thesis. The criteria for determining the statistical significance for an alignment using either of these measures is unclear. The time complexity for both **SI** and **MI** is bounded by the computation of the RMSD term, which takes $O(N_e)$ time.

Each of the protein structural alignment scores discussed above treat coverage and fidelity in ad hoc terms. The next section will present a new way of thinking about the alignment problem, where a statistically rigorous foundation is built on information theory and statistical inference.

4.3 Structural Alignment as an Inductive Inference Problem

The goal of inductive inference is to propose a *theory* (or *hypothesis*) that is best able to explain the observed data. The selection of the best structural alignment fits naturally into the general class of inference problems, since a structural alignment is a hypothesis that attempts to *explain* the residue-residue relationships between protein structures whose observed data are the 3D atomic coordinates.

MML (Wallace and Boulton, 1968; Wallace, 2005) provides a practical information-theoretic criterion for hypothesis selection based on observed data (see Section 3.3.3). A framework based on MML is used in this chapter to formulate an objective assessment criterion for structural alignment quality, one that can reliably differentiate between the quality of competing alignments. As explained in Section 3.3.3, MML attempts to transmit, *losslessly*, the observed data using a hypothesis. The shorter the combined two-part (hypothesis + data) message is, the more favourable the hypothesis is regarding the data. Conceptually, this transmission occurs between some hypothetical transmitter and receiver. However, no actual encoding or transmission is required in order to compute the length of the two-part message. The data here are the

3D C_α atomic coordinates of a pair of protein structures: $\langle S, T \rangle$,* and the hypothesis is the alignment, \mathcal{A} .

Two possible scenarios arise from this description:

- (1) If the two structures are *unrelated* to each other, one cannot do better than to encode and transmit the information of the two structures *independently*, one after another. That is, knowledge of one structure does not inform the receiver about the other and, thus, knowledge of the atomic coordinates in S cannot be used to compress the atomic coordinates in T . This form of independent transmission is called the *null model* message.
- (2) On the other hand, if the two structures are *related* (*i.e.*, if there is a meaningful alignment between the two), then knowledge of the coordinates in S informs the receivers expectations of the coordinates in T . The more closely related the structures, the more information one reveals about the other. The transmitter can therefore use this similarity to *compress* the coordinates in T using what the receiver already knows about the coordinates in S . In order for the receiver to decode the coordinates in T losslessly (*i.e.*, to the precision to which the transmitter sees the original data), they will require the coordinates in S *plus* the proposed relationship (*i.e.*, the structural alignment) to T . This allows the transmitter to encode T more concisely than stating T using a null model. Herein, this form of transmission will be referred to as the *related model* (\mathcal{R} -model) message.

This MML framework for structural alignment is *intuitive*. If the proposed alignment relationship is a poor one, then the encoded \mathcal{R} -model message will be inefficient (*i.e.*, long). Alternatively, if the alignment relationship is a good one, then the \mathcal{R} -model message length becomes efficient (*i.e.*, short). Therefore, the total message length of the lossless transmission of coordinate data using an alignment hypothesis forms an excellent measure to assess the quality of this structural alignment. It follows that *the best structural alignment is the one with the shortest total two-part message length of lossless transmission*.

4.3.1 An Information Measure of Structural Alignment Quality

Formally, let \mathcal{A} denote some alignment between the coordinate data of a pair of protein structures, $\langle S, T \rangle$. The proposed measure estimates the *Shannon information content* (Shannon (1948); see Section 3.2) which states that, given an event E with probability $\Pr(E)$, the information content of E is $-\log_2(E)$ bits.

In this context, the information content (or length) of the null model message, where S and T are assumed to be *unrelated* can be simply stated as:

$$\begin{aligned} I_{\text{null}}(\langle S, T \rangle) &= -\log_2(\Pr(S)\Pr(T)) \\ &= I_{\text{null}}(S) + I_{\text{null}}(T) \quad \text{bits.} \end{aligned} \tag{4.2}$$

Similarly, the information content (or length) of the \mathcal{R} -model message $I(\mathcal{A}, \langle S, T \rangle)$, is the negative logarithm of the joint probability of the alignment hypothesis and the data:

$$I(\mathcal{A}, \langle S, T \rangle) = -\log_2(\Pr(\mathcal{A}, \langle S, T \rangle)) \quad \text{bits.}$$

* S and T are used to refer specifically to the 3D coordinate data in two structures.

Using the product rule of probability over the events \mathcal{A} , S and T :

$$\Pr(\mathcal{A}, \langle S, T \rangle) = \underbrace{\Pr(\mathcal{A})}_{\text{Prior of alignment}} \underbrace{\Pr(\langle S, T \rangle | \mathcal{A})}_{\text{Likelihood}} \quad (4.3)$$

$$\Pr(\mathcal{A}, \langle S, T \rangle) = \underbrace{\Pr(\langle S, T \rangle)}_{\text{Prior of data}} \underbrace{\Pr(\mathcal{A} | \langle S, T \rangle)}_{\text{Posterior}} \quad (4.4)$$

where $\Pr(\mathcal{A}, \langle S, T \rangle)$ is the joint probability of the alignment \mathcal{A} and the structure coordinates in S and T . This product rule can be restated in terms of Shannon's information content by applying a negative logarithm to both sides of the equation:

$$\begin{aligned} \underbrace{-\log_2(\Pr(\mathcal{A}, \langle S, T \rangle))}_{I(\mathcal{A}, \langle S, T \rangle)} &= \underbrace{-\log_2(\Pr(\mathcal{A}))}_{I(\mathcal{A})} + \underbrace{-\log_2(\Pr(\langle S, T \rangle | \mathcal{A}))}_{I(\langle S, T \rangle | \mathcal{A})} \\ &= \underbrace{-\log_2(\Pr(\langle S, T \rangle))}_{I(\langle S, T \rangle)} + \underbrace{-\log_2(\Pr(\mathcal{A} | \langle S, T \rangle))}_{I(\mathcal{A} | \langle S, T \rangle)} \end{aligned}$$

Therefore, the total \mathcal{R} -model message length for transmitting \mathcal{A} , S , and T is:

$$\begin{aligned} I(\mathcal{A}, \langle S, T \rangle) &= \underbrace{I(\mathcal{A})}_{\text{First part}} + \underbrace{I(\langle S, T \rangle | \mathcal{A})}_{\text{Second part}} = I(\mathcal{A}) + I(S | \mathcal{A}) + I(T | S, \mathcal{A}) \\ &= \underbrace{I(\mathcal{A})}_{\text{First part}} + \underbrace{I_{\text{null}}(S) + I(T | S, \mathcal{A})}_{\text{Second part}} \quad \text{bits.} \end{aligned} \quad (4.5)$$

where transmitting the alignment takes $I(\mathcal{A})$ bits, transmitting the coordinate data from S takes $I_{\text{null}}(S)$ bits, and transmitting coordinates from the target structure T using \mathcal{A} and S takes $I(T | S, \mathcal{A})$ bits. In these terms, the first part of the message, $I(\mathcal{A})$, measures the alignment hypothesis complexity (shown in blue). The second part of the message measures the fidelity (shown in red). Note that $I(S | \mathcal{A}) = I_{\text{null}}(S)$ because S is assumed to be independent: \mathcal{A} does not inform S .

4.3.2 Statistical Properties of the Information Measure

The I -value measure has the following three key properties:

1. The difference between the lengths of the messages needed to transmit the structures S and T using any two alignments, gives their log-odds posterior ratio.

Formally, given any two competing alignment hypotheses, \mathcal{A}_1 and \mathcal{A}_2 , the difference in total message length between these is:

$$\begin{aligned} I(\mathcal{A}_2, \langle S, T \rangle) - I(\mathcal{A}_1, \langle S, T \rangle) &= -\log(\Pr(\mathcal{A}_1, \langle S, T \rangle)) + \log(\Pr(\mathcal{A}_2, \langle S, T \rangle)) \\ &= \log \left(\frac{\Pr(\mathcal{A}_2, \langle S, T \rangle)}{\Pr(\mathcal{A}_1, \langle S, T \rangle)} \right) \end{aligned}$$

From the product rule of probability, for an alignment hypothesis, \mathcal{A} :

$$\Pr(\mathcal{A}, \langle S, T \rangle) = \Pr(\langle S, T \rangle) \Pr(\mathcal{A} | \langle S, T \rangle)$$

Therefore, the log-odds posterior ratio between any two competing alignment hypotheses \mathcal{A}_1 and \mathcal{A}_2 is:

$$\begin{aligned} I(\mathcal{A}_2, \langle S, T \rangle) - I(\mathcal{A}_1, \langle S, T \rangle) &= \log \left(\frac{\Pr(\langle S, T \rangle) \Pr(\mathcal{A}_2 | \langle S, T \rangle)}{\Pr(\langle S, T \rangle) \Pr(\mathcal{A}_1 | \langle S, T \rangle)} \right) \\ &= \log \left(\frac{\Pr(\mathcal{A}_2 | \langle S, T \rangle)}{\Pr(\mathcal{A}_1 | \langle S, T \rangle)} \right) \end{aligned}$$

This proposition is important when comparing competing alternative alignments between protein structures S and T , they can now be compared based on their message lengths: the best alignment hypothesis, \mathcal{A}^* , is the one that results in the shortest message length for $I(\mathcal{A}^*, \langle S, T \rangle)$.

2. The information measure permits a *natural null hypothesis test* where the statistical significance of any proposed alignment hypothesis can be estimated by comparing the \mathcal{R} -model message length with the null model message length. Any alignment hypothesis, \mathcal{A} , with an \mathcal{R} -model message length worse (longer) than that of the null model message length *must be rejected*:

$$\text{If } I(\mathcal{A}, \langle S, T \rangle) \geq I_{\text{null}}(\langle S, T \rangle) = I_{\text{null}}(S) + I_{\text{null}}(T) \quad \text{Reject } \mathcal{A}$$

This is because the null model hypothesises no relationship between the pair of structures, $\langle S, T \rangle$. The coordinates of S and T are stated independently of one another. Therefore, their transmission does not benefit from any compression. If a proposed alignment, \mathcal{A} , causes the message required to state $I(\mathcal{A}, \langle S, T \rangle)$ to be longer, then it is a less likely hypothesis by the reasoning from property 1 stated above.

This property is important in deciding whether S and T are related *at all*. Or, more generally, in deciding whether (and to what extent) a given alignment defines a relationship between S and T .

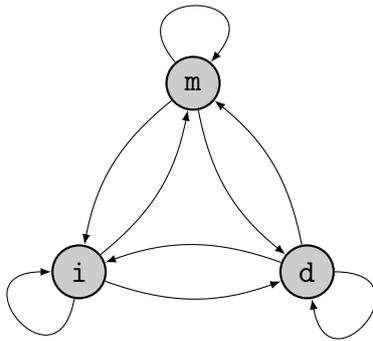
3. The structural alignment problem involves a trade-off between the conflicting objectives of maximising the coverage and fidelity. Note that coverage (in its various manifestations in existing scoring functions) is a crude *approximation* of the alignment hypothesis complexity. Similarly, the fidelity of a structural alignment is *approximated* using the RMSD after least-squares superposition or using some distance measure.

The information measure provides an objective, formal trade-off between the coverage, as measured by the complexity of the alignment hypothesis $I(\mathcal{A})$ and the fidelity of the structures given the proposed alignment ($I(T|S, \mathcal{A})$). Unlike previous attempts, these terms are not *ad hoc* approximations, as they represent rigorous estimations of the Shannon information content based on lossless encoding and compression.

The following sections explore the details of computing the message lengths of each term in Equation 4.5, $I(\mathcal{A})$, $I_{\text{null}}(S)$, and $I(T|S, \mathcal{A})$, using various encoding schemes. Note that these encoding schemes are improved further in Chapter 5.

4.4 Formulation of the Alignment Encoding Message Length: $I(\mathcal{A})$

This section describes four possible methods of encoding the alignment string. While describing these, the term *index* will be used to describe the number of columns encountered since the beginning of the alignment, and the term *offset* will be used to describe the number of residues in either S or T that have been encountered since the last **match** state in S or T , respectively.



(a)

S	-	-	-	E	K	K	-	T	V	L	G	V	G	S	C
T	R	G	T	V	S	R	S	-	-	-	G	T	L	T	-
FSA str	i	i	i	m	m	m	i	d	d	d	m	m	m	m	d

(b)

Figure 4.1: (a) Order-preserving protein structural alignments can be encoded from as a string derived from three-state automaton. (b) An example alignment of two protein sequences containing all possible state transitions. Note that the transition from a delete state to an insert state is not present because it is equivalent to the insert to delete transition. This example contains two contiguous blocks of correspondences (highlighted).

An order-preserving alignment, \mathcal{A} , between the pair of protein structures, $\langle S, T \rangle$, defines a state string over three states: **match** (m), **insert** (i), and **delete** (d) (see Section 2.2). This string is derived from a Finite State Automata (FSA), as shown in Figure 4.1, and encoded as a message. To be decoded, the receiver needs to know the length of the alignment ($|\mathcal{A}|$), *plus* the contents of the string. The length of the alignment will be stated using the universal code for positive integers as described in Section 3.4. This code transmits the length in $I_{\text{integer}}(|\mathcal{A}|)$ bits.

Fixed Width Encoding: This method encodes the alignment as a string of states. In the example from Figure 4.1(b) the string of states would be: “**iiimmmiddmmmm**d”. Each state in the state string can be encoded over a uniform probability distribution. As an alignment can be in three possible states, a state can be encoded in $\log_2(3) \approx 1.58$ bits. This will result in the message length for any alignment, \mathcal{A} , being the number of states in the alignment *plus* $\log_2(3)$ bits for each state:

$$I_{\text{Fixed Width}}(\mathcal{A}) = I_{\text{integer}}(|\mathcal{A}|) + |\mathcal{A}| \log_2(3) \quad \text{bits.}$$

As demonstrated in Section 4.4.1 below, this encoding method is too inefficient and is not considered any further.

Time complexity: for computing $I(\mathcal{A})$ using this method is constant since only the length of the state string is required.

Corresponding Block Encoding: This encoding method treats an alignment as an ordered set of contiguous, monotonic blocks of correspondences that do not contain any gaps. In the example from Figure 4.1(b), such blocks are highlighted in gray. Each block of correspondences can be identified by a tuple of three values: the offset of the first corresponding residue in S , likewise for the first in T , and finally the length of the block of correspondences. For the example in Figure 4.1(b), the alignment has two blocks of correspondences which are highlighted: $(0, 3, 3)$ and $(3, 1, 4)$. To make the message decodable, the transmitter must also state three preliminary values: the number of (tuples representing) matched blocks in the alignment, and the number of trailing **i** and **d** states that occur after the final matched block. Thus the encoding of the example alignment in Figure 4.1(b) is: $2, 0, 1, (0, 3, 3), (3, 1, 4)$ as it has two matched blocks, zero trailing **insert** states and one trailing **delete** state. All values are transmitted using the integer code (see Section 3.4) resulting in:

$$I_{\text{Block}}(\mathcal{A}) = I_{\text{integer}}(2) + I_{\text{integer}}(0) + I_{\text{integer}}(1) + I_{\text{integer}}(0) + I_{\text{integer}}(3) + I_{\text{integer}}(3) \\ + I_{\text{integer}}(3) + I_{\text{integer}}(1) + I_{\text{integer}}(4) \quad \text{bits.}$$

Time complexity: of this **Block** alignment encoding scheme is linear in the length of the state string. This is because the string needs to be iterated over to find the indexes of the blocks of correspondences.

Run Length Encoding (RLE): As before, this method encodes the alignment as a string of states. In the example from Figure 4.1(b) above, the string of states would be: “**iiimmmiddmmmd**”. Biological macromolecules retain local similarities between related structures such as those created by helical secondary structures. These are often aligned together as a contiguous *run* of correspondences with contiguous runs of gap states (**i** or **d**) in between. Therefore, it may be assumed that biologically significant alignments are generated by automata that only infrequently transition between different states and instead prefer to continue in the same state, (see Figure 4.1(a)) usually resulting in long blocks of equal states.

This knowledge is used by the RLE encoding method to efficiently encode an alignment state string as follows. Firstly, state the total number of contiguous runs in the alignment, which is 6 in the example from Figure 4.1(b). Then state the type and length of each run. Thus, the example alignment state string in Figure 4.1(b) would be encoded as: $6, (\mathbf{i}, 3), (\mathbf{m}, 3), (\mathbf{i}, 1), (\mathbf{d}, 3), (\mathbf{m}, 4), (\mathbf{d}, 1)$. All integer values are transmitted using the integer code (see Section 3.4), while states are transmitted using a fixed-width code in $\log_2(3) \approx 1.58$ bits.

Time complexity: of the RLE method is linear in the number of states in the alignment state string. Each state in the string needs to be examined to find the length of each run.

Adaptive First-Order Markov Model Encoding: An alternate method is to treat the alignment state string as a sequence of transitions between states, rather than a sequence of states. In the example from Figure 4.1(b), this would mean encoding the alignment state string, “**iiimmmiddmmmd**”, as a series of state transitions: $(\mathbf{i} \rightarrow \mathbf{i}), (\mathbf{i} \rightarrow \mathbf{i}), (\mathbf{i} \rightarrow \mathbf{m}), (\mathbf{m} \rightarrow \mathbf{m}), (\mathbf{m} \rightarrow \mathbf{m}), (\mathbf{m} \rightarrow \mathbf{i}), (\mathbf{i} \rightarrow \mathbf{d}), (\mathbf{d} \rightarrow \mathbf{d}), (\mathbf{d} \rightarrow \mathbf{d}), (\mathbf{d} \rightarrow \mathbf{m}), (\mathbf{m} \rightarrow \mathbf{m}), (\mathbf{m} \rightarrow \mathbf{m}), (\mathbf{m} \rightarrow \mathbf{m}), (\mathbf{m} \rightarrow \mathbf{d})$. For the sake of brevity, this thesis uses the notation, **md** to refer to the $(\mathbf{m} \rightarrow \mathbf{d})$ transition and so on. This series of state transitions can be encoded very efficiently using an approach similar to the *adaptive encoding* method used by Wallace and Boulton (1969) over a 3-state Markov chain. This model permits 9 possible state transitions, **mm**, **mi**, **md**, **im**, **ii**, **id**,

dm , di , and dd . Associated with each possible state transition is a transition probability, as in Figure 4.2, $\Pr(mm)$, $\Pr(mi)$, $\Pr(md)$, and so on. The transition probabilities are constrained such that the sum of all probabilities along edges leaving a state must sum to 1. For example, $\Pr(mm) + \Pr(mi) + \Pr(md) = 1$.

While the above probabilities can be computed for any given alignment on the transmitters side, the receiver needs to know these probabilities in advance to be able to decode the alignment. An adaptive code is an efficient approach to construct a decodable message over this first order Markov model. The adaptive encoding here requires maintaining nine running counters, one for each possible transition probability, all initialised to 1. As a boundary case, the first state is transmitted with a uniform probability of $1/3$. Traversing the alignment string left to right, for every observed transition, the transmitter estimates its probability by dividing the current value of the corresponding transition counter by the sum of all counters from previous to any state. After estimating the probability, the transmitter encodes the current alignment state using this probability and then increments the corresponding counter by 1.

The code length to encode each state is the negative logarithm of its estimated probability. Summing this over all state transitions in the alignment FSA string, and adding it to the code length required to transmit the size of the alignment over the integer code results in the estimation of $I(\mathcal{A})$ using this method. From the adaptive transition probability estimates in Table 4.1 below, the example in Figure 4.1(b) would be encoded with the following message length:

$$\begin{aligned} I_{\text{Markov}}(\mathcal{A}) &= I_{\text{integer}}(15) - \log_2(1/3) - \log_2(1/3) - \log_2(2/4) - \log_2(1/5) - \log_2(1/3) - \log_2(2/4) \\ &\quad - \log_2(1/5) - \log_2(1/6) - \log_2(3/7) - \log_2(4/8) - \log_2(2/9) - \log_2(3/7) - \log_2(4/8) \\ &\quad - \log_2(5/9) - \log_2(2/10) \end{aligned}$$

The example alignment in Figure 4.1(b) is encoded using the adaptive Markov method as described above in Table 4.1.

Time complexity: for computing $I(\mathcal{A})$ using this method takes linear time in the length of the state string. This is because each transition between states in the state string needs to be examined in order.

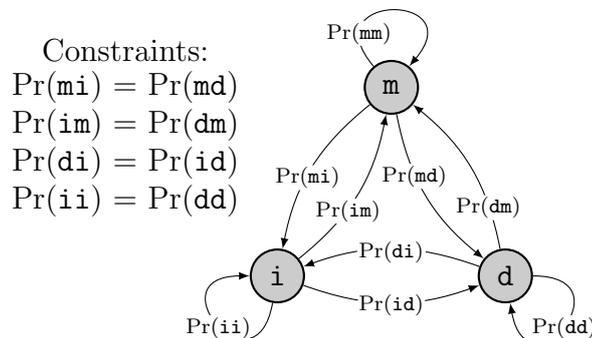


Figure 4.2: A three-state automata, with transition probabilities marked, used for adaptive first-order Markov encoding of an alignment. Symmetrical transition probabilities are enforced according to the constraints listed on the left.

Table 4.1: An illustrated example of the transmission of the alignment state string in Figure 4.1(b), “*iiimmiddmmmmmd*” using the Adaptive First-Order Markov encoding method. A star (*) is placed next to a counter value when it is updated after being used to encode a state transition. The state transition probability is estimated before the transition counters are updated. These values are given in parentheses. Note that this update occurs in symmetrical rows according to the four symmetry constraints listed in Figure 4.2.

Alignment transitions:	$\rightarrow i$	$i \rightarrow i$	$i \rightarrow i$	$i \rightarrow m$	$m \rightarrow m$	$m \rightarrow m$	$m \rightarrow i$	$i \rightarrow d$	$d \rightarrow d$	$d \rightarrow d$	$d \rightarrow m$	$m \rightarrow m$	$m \rightarrow m$	$m \rightarrow m$	$m \rightarrow d$
$m \rightarrow m$ counter	1	1	1	1	(1)2*	(2)3*	3	3	3	3	3	(3)4*	(4)5*	(5)6*	6
$m \rightarrow i$ counter	1	1	1	1	1	1	(1)2*	2	2	2	2	2	2	2	3*
$m \rightarrow d$ counter	1	1	1	1	1	1	2*	2	2	2	2	2	2	2	(2)3*
$m \rightarrow^*$ total	3	3	3	3	(3)4	(4)5	(5)7	7	7	7	7	(7)8	(8)9	(9)10	(10)12
$i \rightarrow m$ counter	1	1	1	(1)2*	2	2	2	2	2	2	3*	3	3	3	3
$i \rightarrow i$ counter	1	(1)2*	(2)3*	3	3	3	3	3	4*	5*	5	5	5	5	5
$i \rightarrow d$ counter	1	1	1	1	1	1	1	(1)2*	2	2	2	2	2	2	2
$i \rightarrow^*$ total	3	(3)4	(4)5	(5)6	6	6	6	(6)7	8	9	10	10	10	10	10
$d \rightarrow m$ counter	1	1	1	2*	2	2	2	2	2	2	(2)3*	3	3	3	3
$d \rightarrow i$ counter	1	1	1	1	1	1	1	2*	2	2	2	2	2	2	2
$d \rightarrow d$ counter	1	2*	3*	3	3	3	3	3	(3)4*	(4)5*	5	5	5	5	5
$d \rightarrow^*$ total	3	4	5	6	6	6	6	7	(7)8	(8)9	(9)10	10	10	10	10
Estimated probability	1/3	1/3	2/4	1/5	1/3	2/4	1/5	1/6	3/7	4/8	2/9	3/7	4/8	5/9	2/10

4.4.1 Selecting an Alignment Encoding Scheme

As illustrated above, there are many possible encoding schemes available to encode alignments. The aim is to define the best encoding method, that is able to state an alignment using the shortest possible message. To determine which of the above schemes is best, they were evaluated using pairwise benchmark alignments provided by the SABmark (Walle et al., 2005) database. SABmark provides 29,759 pairwise alignments in May 2016, when this evaluation was conducted.

Each of the alignment encoding schemes above was used to compute the length of the alignment encoding, $I(\mathcal{A})$, for all of the benchmark pairwise alignments provided by SABmark. The results are plotted in Figure 4.3 using a notched box-and-whisker plot to show the distribution of encoding lengths. Based on these results it is clear that the **Markov** encoding scheme provides the best compression among the methods discussed, closely followed by the **Block** encoding method.

The Adaptive First-Order **Markov** scheme has a median encoding length for $I(\mathcal{A})$ of 111.3 bits with an inter-quartile range of 68.09 bits. The next best is the **Block** encoding scheme which achieves a median encoding length for $I(\mathcal{A})$ of 131.9 bits with an inter-quartile range of 78.33 bits. The RLE encoding scheme achieves a median encoding length for $I(\mathcal{A})$ of 145.4 bits with an inter-quartile range of 90.7 bits. Finally, as expected, the naïve fixed width encoding scheme is much worse than the others. It achieves a median encoding length for $I(\mathcal{A})$ of 324.9 bits with an inter-quartile range of 259.9 bits. **Therefore, the Adaptive Markov encoding scheme is used, for the remainder of this chapter, as the scheme for the estimation of $I(\mathcal{A})$ since it results in the shortest encoding length of the benchmark alignments and the smallest deviation from its median.**

4.5 Formulation of the Null Model Message Length: $I_{\text{null}}(\cdot)$

The null model message transmits the protein coordinates in S and T independently, that is, without an alignment hypothesis. This is done to either transmit S (for later transmission of T using an alignment hypothesis) or to test the alignment hypothesis against the null model

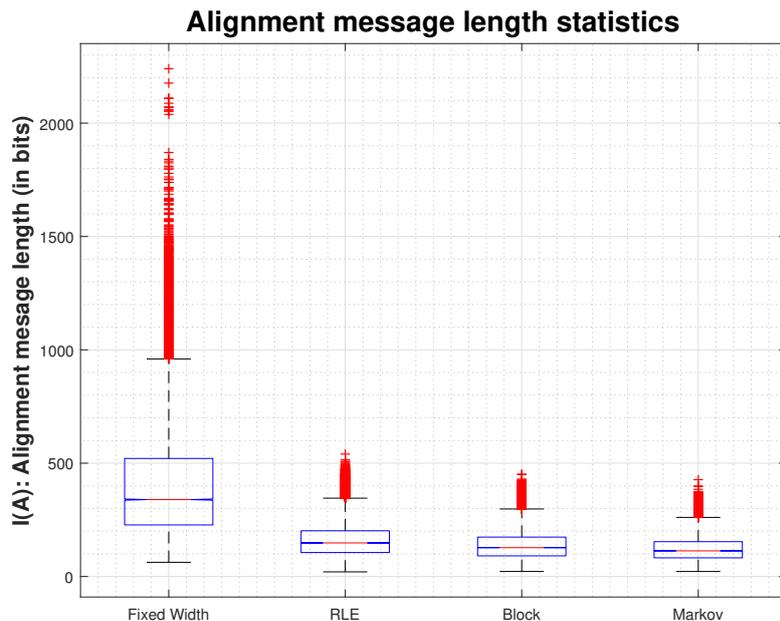


Figure 4.3: A comparison of the various alignment encoding schemes presented in Section 4.4. The plot breaks the range encoding lengths into four parts: the bottom line of the box marks the 25th percentile of the message lengths, the line through the center of the box marks the median or 50th percentile of the message lengths, while the top line in the box marks the 75th percentile of the message lengths. The length of the bottom and top whiskers represent the range of the first and fourth quartiles, respectively. The size of the notch shows the 95% confidence interval for the median.

statement of S and T . Recall that in this work, only the C_α atomic coordinates are considered (see Section 4.3). Recall also, that protein coordinate data are recorded to a precision of three decimal places in the PDB (see Section 2.1.2). Even though this is not the experimental precision to which protein structures are determined. In this work, the coordinate data is stated to three decimal places so that the receiver can reconstruct the coordinate data to the same precision, ϵ , that the transmitter sees in PDB coordinate data files (see Section 2.1.2).

This section presents several potential null model encoding schemes. Note that a useful null encoding scheme should be independent of the coordinate frame-of-reference: it should not depend on the position and orientation of the structure in space.

Fixed-width encoding: The most naïve Null encoding treats the (x, y, z) coordinate data as a string of symbols drawn from a dictionary of 12 elements: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -, .\}$. For example, a C_α atom with coordinates $x = -1.739, y = -34.450, z = 10.061$ can be expressed as the string: “-1.739-34.45010.061”,[†] which is 19 symbols long. Letting C denote a chain of C_α coordinates, a particular coordinate in C , C_i is represented as a string of symbols containing $|C_i|$ such symbols. The transmitter simply states the number of symbols in the string in $I_{\text{integer}}(|C_i|)$ bits and then the transmitter encodes the string itself in $|C_i| \log_2(12)$ bits. Using this measure, the chain of C_α coordinates, C is encoded using a message with a length of: $I_{\text{fixed-width}}(C) = I_{\text{integer}}(|C|) + |C| \times (I_{\text{integer}}(|C_i|) + |C_i| \log_2(12))$ bits.

Each symbol is assigned a unique fixed-size code of $\log(12) \approx 3.6$ bits. Therefore, the message length required to encode the example above, using the fixed-width encoding method is $I_{\text{integer}}(19) + 19 \log_2(12) \approx 77$ bits.

This encoding scheme has problems that make it undesirable as a null encoding model. Firstly, this encoding method does not take advantage of crucial aspects of the protein structure, especially the nature of the data as *chain* of coordinates. Furthermore, each symbol in the dictionary is not equally likely and thus transmitting each symbol over a uniform distribution

[†]The receiver can delimit x from y from z coordinates using the position of the decimal point. The next coordinate begins after encountering three symbols after a decimal point.

(where each symbol in the dictionary is assigned an equal probability) is inefficient, giving a poor estimate for $I_{\text{null}}(\langle S, T \rangle)$.

Time complexity: The time required to compute the null model message length depends on the number of coordinates, and the number of symbols per coordinate. Since the number of symbols per coordinate is essentially constant, computing the null model message length with the fixed-width encoding model takes linear time in the number of coordinates in the chain, $O(|C|)$.

Bounding-box encoding: A more efficient encoding method bounds the space around a protein chain with a cuboid, or *bounding box*. The space within the bounding box is then divided into elements of size $\epsilon \times \epsilon \times \epsilon$. Any C_α coordinate can then be stated as the coordinates of an element to the correct precision of measurement. Firstly the transmitter establishes preliminaries. The size of the bounding box may be encoded using the fixed-width encoding described above. And the number of C_α coordinates in the chain of atoms, C , can be encoded using the integer code in $I_{\text{integer}}(|C|)$ bits.

To encode the positions of each C_α atom, let the bounding box have total edge lengths: δx , δy , and δz . The x coordinate of any element within the bounding box can then be stated in $\log(\frac{\delta x}{0.001}) \equiv \log(\delta x \times 1000)$ bits, and equivalently for the y and z coordinates. Therefore, a chain of C_α coordinates, C , is encoded using this scheme in: $I_{\text{bounding-box}}(C) = I_{\text{integer}}(|C|) + I_{\text{fixed-width}}(\delta x) + I_{\text{fixed-width}}(\delta y) + I_{\text{fixed-width}}(\delta z) + |C| \times \log_2(\delta x \delta y \delta z \times 1000^3)$ bits.

While more efficient in terms of message length than the fixed-width encoding scheme, this bounding-box approach has similar problems to the fixed-width encoding scheme, including not taking advantage of the fact that the protein is a chain of coordinates. The next encoding scheme to be discussed rectifies this issue by accounting for the protein chain.

Time complexity: The time complexity for computing the null model message length using this approach takes constant time, $O(1)$. This is because no iteration is required.

Uniform-directional encoding: A better model that takes advantage of the nature of protein chains is defined by Konagurthu et al. (2012). This scheme represents the coordinates of C_α atoms using a spherical coordinate system with three axes: $(r, \langle \theta, \phi \rangle)$. This coordinate system is orthogonal and equivalent to the Cartesian coordinate system.

Firstly, recall that the distance between successive C_α atoms in a protein chain is highly constrained around 3.8 \AA (see Section 2.1.5). Given a chain of coordinates, C_1, C_2, \dots, C_n , any coordinate C_j can be transmitted given the previous C_{j-1} , by first transmitting the radius r_j between C_{j-1} and C_j using a normal distribution $\mathcal{N}(r; \mu, \sigma)$ stated to $\epsilon = 0.001 \text{ \AA}$ precision, with $\mu = 3.8 \text{ \AA}$ and $\sigma = \pm 0.2 \text{ \AA}$. The length of the message to encode r_j is $I(r_j)$.

$$I_{\text{radius}}(r_j) = -\log_2(\mathcal{N}(r_j; \mu, \sigma) \cdot \epsilon) \quad \text{bits.} \quad (4.6)$$

This is the standard method to transmit the distance between consecutive C_α atoms which is used throughout this thesis.

With this information transmitted, the receiver now knows that C_j lies on a sphere of radius r_j centred at C_{j-1} , but does not yet know exactly where on the sphere C_j lies. Assuming that C_j is distributed uniformly over the surface of the sphere, the transmitter can discretise the surface of the sphere into cells, each of area ϵ^2 . Using this discretisation, C_j can simply be transmitted as a cell index number, c_j , over a uniform distribution. With the knowledge of

C_{j-1} , r_j and c_j , the receiver can reconstruct C_j to ϵ precision. Stating the cell index number uniformly on a sphere with surface area equal to $4\pi r_j^2$ takes:

$$I(c_j) = -\log_2 \left(\frac{\epsilon^2}{4\pi r_j^2} \right) = \log_2(4\pi r_j^2) - 2 \log_2 \epsilon \quad \text{bits.} \quad (4.7)$$

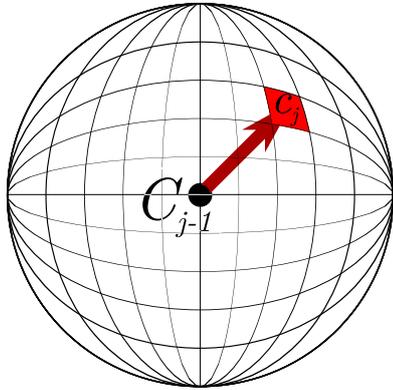


Figure 4.4: A stylised representation of the uniform directional encoding model. The direction of an unknown coordinate, C_j , from the position of a known coordinate, C_{j-1} , is encoded on the surface of a sphere dissected into cells with area ϵ^2 . Each cell is numbered (or indexed) according to an agreed upon scheme. The index number of the cell containing C_j is transmitted with uniform probability.

To transmit the chain of C_α coordinates C_1, C_2, \dots, C_n using this encoding model, the message starts with the number of C_α atoms in the chain, $|C|$, followed by incrementally transmitting (using the method above) the chain of coordinates. $|C|$ can be transmitted using the integer code in $I_{\text{integer}}(|C|)$ bits (refer to Section 3.4). From Equation 4.6 and Equation 4.7, the message length required to state a coordinate, C_j , is:

$$I_{\text{null}}(C_j) = I_{\text{radius}}(r_i) + I(c_i) \quad \text{bits.} \quad (4.8)$$

The total message length required to send a chain, C , of coordinates is therefore:

$$I_{\text{uniform-sphere}}(C) = I_{\text{integer}}(|C|) + \sum_{i=2}^{|C|} I_{\text{null}}(C_i) \quad \text{bits.}$$

Time complexity: The time required to compute the null model message length using this scheme depends on the computation of the radius and the position on the surface of the sphere. Both of these (constant time) operations, are performed $|C|$ times for each C_α atom in the coordinate chain. Therefore, the uniform-directional encoding model is a linear time, $O(|C|)$, operation.

4.5.1 Selecting a Null Encoding Scheme

To determine which of the three null encoding schemes described above is best, they were evaluated using 3000 randomly selected SCOP domains (see Section 4.8 and Appendix A for details). Each SCOP domain was encoded using each of the three encoding models and the resulting message lengths divided by the length (number of C_α atoms) of the domain. This calculation gives an average message length to encode a C_α coordinate in that domain. The results were plotted using a notched box-and-whisker plot to show the distribution of average message lengths in Figure 4.5. The fixed-width encoding method encodes a C_α coordinate with a median average message length of 51.24 bits and inter-quartile range of 0.75 bits. The bounding-box method requires a median 45.4 bits on average to encode a C_α coordinate with an inter-quartile range

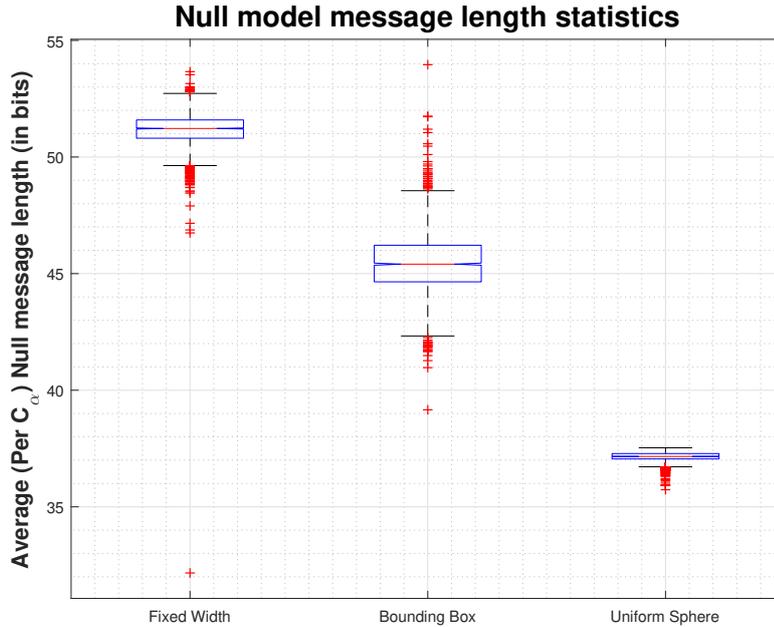


Figure 4.5: A comparison between the average (per C_α) message lengths of various coordinate encoding models. The notched box-and-whisker plots divide the data as follows: the bottom of the box is the 25th percentile of average message lengths the, line through the center of the box is the median or 50th percentile of average message lengths, and the top line in the box is the 75th percentile of average message lengths. The length of the bottom and top whiskers represent the range of the first and fourth quartiles respectively. The size of the notch indicates the 95% confidence interval for the median. Outliers are shown as red points.

of 1.52 bits. Clearly, the most efficient method of encoding is the uniform-direction method encodes which encodes a C_α coordinate with a median average message length of 37.16 bits and inter-quartile range of 0.23 bits. **Therefore, the uniform-direction encoding scheme is used as the null model for all experiments in this chapter.**

4.6 Formulation of the Coordinate Compression Model:

$$I(T|S, \mathcal{A})$$

With the information of S and \mathcal{A} known to the receiver, the transmitter can now use that information to encode the coordinates of T . Intuitively, encoding of T is based on the fact that when scanning \mathcal{A} from left to right, T contains runs of coordinates that alternate between blocks of insertions and matches with respect to S and the stated alignment. Note that, in this transmission, all deletion blocks (with respect to S) in T are ignored as they contain no information to be transmitted about T (this can be seen in Figure 4.1(b)). More formally, the insertion blocks are encoded as follows. Let \mathcal{A} yield $\{\mathcal{I}_1, \dots, \mathcal{I}_m\}$ insertion blocks, where any \mathcal{I}_k represents a consecutive run of coordinates that are inserted in T (with respect to S). Each insertion block is transmitted, using Equation 4.5, as a null message taking:

$$I_{\text{ins}}(T|S, \mathcal{A}) = \sum_{k=1}^m \sum_{i=1}^{|\mathcal{I}_k|} I_{\text{null}}(\mathcal{I}_{ki})$$

For the example in Figure 4.1(b), the alignment yields two **insert** blocks. The first containing three residues from T , $\{R, G, T\}$. The second contains only one residue from T , $\{S\}$. As described above, these four coordinates are transmitted using the uniform-direction model.

What remains to be sent are the coordinates in the blocks of coordinates in T that are aligned to corresponding C_α coordinates in S . Which, for the example in Figure 4.1(b), are $\{V, S, R\}$, and $\{G, T, L, T\}$. Let $\{\vec{s}_1, \vec{s}_2, \dots, \vec{s}_{N_e}\}$ and $\{\vec{t}_1, \vec{t}_2, \dots, \vec{t}_{N_e}\}$ denote the ordered set of corresponding coordinates in S and T , respectively. The receiver already knows S and the alignment. From the alignment information the receiver can infer the indexes of the aligned residue-residue correspondences between S and T . In the example from Figure 4.1(b), these correspondences are, (E, V) , (K, S) , (K, R) , (G, G) , (V, T) , (G, L) , and (S, T) . Thus, the following procedure can be used to transmit the aligned coordinates in T . To start the procedure, the first[‡] three matched coordinates in T $\{t_{j_1}, t_{j_2}, t_{j_3}\}$ are sent over the null model message taking:

$$I_{\text{startup}}(T|S, \mathcal{A}) = I_{\text{null}}(\{\vec{t}_1, \vec{t}_2, \vec{t}_3\}) \quad \text{bits.}$$

For the example in Figure 4.1(b), these first three matched coordinates in T are, $\{V, S, R\}$. The transmission of this startup message ensures that orientation information is available to the receiver. The remaining aligned coordinates in T are then transmitted *incrementally* so that the receiver does not need to know the orientation of T after least-squares superposition ahead of time. Instead, the receiver computes a superposition *adaptively*, based on the information it has at any point in time. This is achieved by the following method. To transmit the current matched coordinate \vec{t}_{j+1} , the transmitter considers only the set of matched coordinates $\{\vec{t}_1, \vec{t}_2, \dots, \vec{t}_j, \vec{t}_{j+1}\}$. For simplicity, let this set represent the transformed state after least squares superposition with the corresponding coordinates in S (see Section 2.2.3). In this procedure, an *adaptive superposition* is repeated for each pair of aligned coordinates. Throughout this procedure the set of coordinates from S is treated as fixed (under orthogonal transformation) while all coordinates in T are rotated and translated. An example of adaptive superposition is illustrated in Figure 4.6.

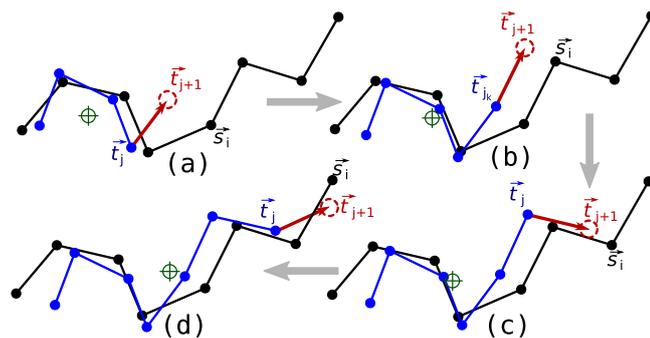


Figure 4.6: An idealised example of the adaptive superposition used to send the matched residues in T (in blue) incrementally given the knowledge of S (in black). Both structures have 8 points and are assumed here to be in one-to-one correspondence. Assume that the receiver already knows the first 3 points of T . The transmitter sends the fourth point in T by superposing all previously matched points between the two structures. (Green crosshairs shows the rotational center of superposition.) This orients the fourth point (in red) in T or, more generally, t_{j+1} , whose deviation from its corresponding s_{i+1} can be encoded over a von Mises-Fisher spherical distribution.

[‡]Only the first, not the first in every matched block.

Using this setup, \vec{t}_{j+1} is transmitted over a directional distribution on a sphere. This is achieved by first transmitting the radius $r_j = |\vec{t}_{j+1} - \vec{t}_j|$ over a normal distribution as in Equation 4.6. This allows the transmitter to state \vec{t}_{j+1} as a point on a sphere with radius r_j centred at \vec{t}_j . However, it is not stated over a uniform distribution (which would make it a null model description), since the knowledge of the correspondence between \vec{t}_{j+1} with s_{i+1} informs the receiver about its position on the sphere (provided the assigned correspondence is a ‘good’ one). Since the receiver already knows the corresponding point s_{i+1} , after transmitting r_j , a von-Mises Fisher (vMF) directional probability distribution is used to state \vec{t}_{j+1} more concisely. In directional statistics, the vMF distribution (see Section 5.2.2) gives a probability density function on the surface of any sphere. In 3D, the probability is distributed symmetrically around a mean direction with circular density contours. This effectively allows the receiver to predict the direction of the next coordinate, gaining compression when the prediction is accurate.

Using this distribution to transmit \vec{t}_{j+1} , \hat{x}_{j+1} is computed as the direction cosines of the vector $\vec{t}_{j+1} - \vec{t}_j$, and $\hat{\mu}_{j+1}$ as the direction cosines of the vector $\vec{s}_{i+1} - \vec{t}_j$. The probability of stating \vec{t}_{j+1} to the required precision using the vMF distribution over the surface of a 3D sphere of unit radius is then given by:

$$\Pr(\hat{x}) = \epsilon'^2 \frac{\kappa}{2\pi(e^\kappa - e^{-\kappa})} e^{\kappa \hat{\mu} \cdot \hat{x}}$$

where $\epsilon'^2 = \frac{\epsilon^2}{r_j^2}$, accounting for the scaling of the sphere of radius r_j to a unit sphere. Transmission of each \vec{t}_{j+1} requires the concentration parameter κ .

The *maximum-likelihood* estimator is used based on the available superposition (see Mardia and Jupp (1999)). The above procedure works only when the receiver and transmitter are encoding and decoding the points using exactly the same concentration parameter κ . To avoid stating κ explicitly as a part of the message, and to minimise the computational complexity of the procedure, a *maximum likelihood estimate*, κ_{ml} , can be inferred based on the previous \hat{x}_i 's observed so far in the incremental procedure described above using a method by Banerjee et al. (2005). In fact, neither transmitter nor the receiver need to store all previous values of \hat{x} since $\sum \hat{x}_i$ forms the *sufficient statistic* for κ_{ml} and this can be updated efficiently as the procedure iterates. If there are N previous \hat{x} observations, let $\bar{R} = \frac{\sum_{i=1}^N \hat{x}_i}{N}$, then:

$$\kappa_{\text{ml}} = \frac{\bar{R}(3 - \bar{R}^2)}{1 - \bar{R}^2}$$

Since the encoding of a current \hat{x}_i is based on a κ_{ml} , the receiver can decode the coordinates by computing the same κ_{ml} that the transmitter uses to encode the coordinates. The code length to state \hat{x} using a vMF distribution on a unit sphere is then: $I_{\text{vmf}}(\hat{x}) = -\log(\Pr(\hat{x}))$ bits.

Each \vec{t}_{j+1} is transmitted iteratively over this procedure, which is termed *adaptive* superposition. An illustration of this adaptive superposition procedure is shown in Figure 4.6. The message length required to transmit the matched coordinates in T with respect to the matched coordinates in S is:

$$I_{\text{match}}(T|S, \mathcal{A}) = I_{\text{startup}}(T|S, \mathcal{A}) + \sum_{i=4}^{N_e} I_{\text{radius}}(r_i) + I_{\text{vmf}}(\hat{x}_i) \quad \text{bits.}$$

Combining the message lengths of transmitting coordinates in the insertion and matched blocks gives:

$$I(T|S, \mathcal{A}) = I_{\text{ins}}(T|S, \mathcal{A}) + I_{\text{match}}(T|S, \mathcal{A}) \quad \text{bits.}$$

4.7 Handling Shifts and Rotations

So far, the information measure has been estimated under a rigid model of structural alignment. The rigid model treats the structures being aligned as rigid under least-squares superposition. This model can be generalised to handle certain common types of plastic deformations commonly observed in protein evolution, such as hinge rotations and shifts (see Section 2.1.6 for an introductory treatment of plastic deformations in protein structures). Handling these deformations requires a modification in the way $I(T|S, \mathcal{A})$ and $I(\mathcal{A})$ are estimated, yielding a *flexible* model of transmission. A flexible encoding model does not treat the structures as rigid, but allows for shifts and rotations.

Without loss of generality, assume that T contains a certain number of shifts and rotations, with respect to S , associated with its residues. In computing $I(T|S, \mathcal{A})$, alignment \mathcal{A} is partitioned at the residues in T where the shifts and rotations occur. For example, consider the alignment in Figure 4.7 containing a hinge rotation. Then, the alignment can be partitioned into two separate parts: before the hinge and after the hinge. In this example, the hinge is at residue 10 of T .

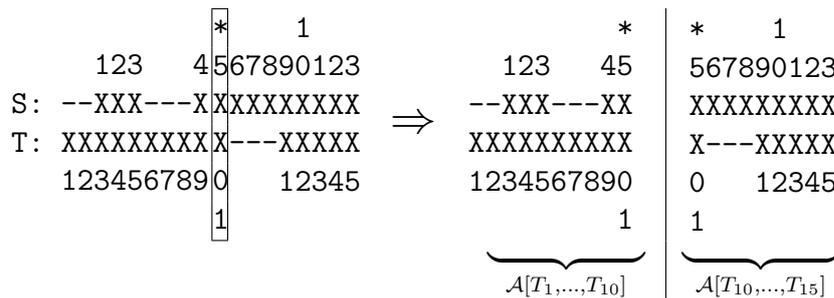


Figure 4.7: An example alignment to be encoded with a hinge in T at residue offset of 10. This column is framed and marked by the star (*) on the left. The flexible alignment is then treated as two separate partial alignments on the right, where a partial alignment contains residues T_1 to T_{10} , and the other contains residues T_{10} to T_{15} .

Let these partial alignments be denoted as $\mathcal{A}[T_1, \dots, T_{10}]$ and $\mathcal{A}[T_{10}, \dots, T_{15}]$, identifying the start and end residue indexes in T at which the partition is defined. Then $I(T|S, \mathcal{A})$ is computed as $I(T|S, \mathcal{A}[T_1, \dots, T_{10}]) + I(T|S, \mathcal{A}[T_{10}, \dots, T_{15}])$ using the unmodified coordinate compression model from Section 4.6 above. More generally, if there are k residues in T about which shifts or hinge rotations are defined, the full alignment \mathcal{A} is partitioned into $k + 1$ partial alignments: $\mathcal{A}[T_1, \dots, T_{i_1}]$, $\mathcal{A}[T_{i_1}, \dots, T_{i_2}]$, \dots , $\mathcal{A}[T_{i_k}, \dots, T_{|T|}]$, where $i_1 < i_2 < \dots < i_k < |T|$. Given these partitions, $I(T|S, \mathcal{A})$ can be computed as $I(T|S, \mathcal{A}[T_1, \dots, T_{i_1}]) + \dots + I(T|S, \mathcal{A}[T_{i_k}, \dots, |T|])$. This immediately poses another inference question: Given an alignment \mathcal{A} of S and T , how many shifted or hinge rotated residues does it contain? Note that adding a shift or hinge rotation has an overhead for which must pay by achieving a better fit, if it is to be accepted. The overhead is a statement of the number of shifts or hinge rotations, k , which can be efficiently encoded in $I_{\text{integer}}(k)$ bits, plus the positions of the shifts/hinges, which can each be transmitted as integer offsets from the beginning of T also using the integer code (see Section 3.4). This

message length represents a further trade-off between complexity (in this case the number of shifts/hinges) and the fidelity of fit after rotating or shifting.

Inference of shifted/rotated residues: A dynamic programming algorithm (see Section 6.2.4) is used to optimally partition \mathcal{A} minimizing $I(T|S, \mathcal{A})$. The algorithm first constructs a matrix M of size $|T| \times |T|$ such that each cell $M(i, j)$ ($1 \leq i < j \leq |T|$) stores the value $I(T|S, \mathcal{A}(i, \dots, j))$ that is, the value of a partition when there are no hinges from i to j . The best partition of \mathcal{A} is then computed using the following one-dimensional dynamic programming recurrence relationship:

$$\mathcal{P}(1, \dots, j) = \min_{i=1}^{j-1} \begin{cases} M(1, j) + I_{\text{integer}}(1), \\ \mathcal{P}(1, \dots, i) + M(i, j) + I_{\text{integer}}(|\mathcal{P}(1, \dots, i)| + 1) \end{cases} \quad \forall 1 \leq j \leq |T|$$

where any $\mathcal{P}(1, \dots, i)$ gives the optimal partitioning (positions of the hinges/shifts) up to the i^{th} residue in T , $1 \leq i \leq |T|$ and $|\mathcal{P}(1, \dots, i)|$ gives the number of such partitions. This recurrence relationship simply states that the prefix of the alignment from $1 \dots j$, can be optimally stated with no hinges/shifts (first line of the formula), or with the optimal segmentation of the prefix $1 \dots i < j$ plus another segment from $i \dots j$. At the end of this procedure the value $\mathcal{P}(1, \dots, |T|)$ gives the component message length $I(T|S, \mathcal{A})$ of Equation 4.5, in a way that handles shift and hinge rotations. **This method of computing $I(T|S, \mathcal{A})$ is used for all experiments in the remainder of this chapter.**

4.8 Results and Discussion

In this Section, the information measure is compared with the nine popular scoring functions discussed in Section 4.2: DALI z-score (Holm and Sander, 1993), TM-Score (Zhang and Skolnick, 2004), MI and SI (Kleywegt and Jones, November 1994), STRUCTAL_score (Subbiah et al., 1993; Gerstein and Levitt, 1998; Levitt and Gerstein, 1998), GDT_TS and LGA_S3 (Zemla, 2003), SAS (Subbiah et al., 1993), and GSAS (Kolodny et al., 2005). Notably, there is no gold standard to compare structural alignment quality measures against. This has doubtless led to some of the proliferation of these measures (Slater et al., 2013). Rather than establishing the information measure as superior, the results in this section are a consistency check amongst these other quality measures and against the SCOP (Murzin et al. (1995); see Section 2.1.4) hierarchy as a baseline.

The section is divided into two experiments. Firstly, a large scale comparison is made among these scoring functions using the SCOPe (Fox et al., 2013) database to ensure that the information measure follows the expected profile of alignment quality between the levels of the SCOP hierarchy. Secondly, the level of disagreement between these scoring functions is evaluated. This evaluation relies on measures for disorder in ranked lists. Such measures are presented in Section 7.1 along with a new MML based measure based on similar principles to the MML based alignment quality measure described in this chapter.

Notched box-and-whisker plots are employed throughout this thesis because they can be very useful for encapsulating the distribution of the data. Any ‘box’ in a box-and-whisker plot represents the region between the first (25th percentile) and the third (75th percentile), where the height of the box gives the interquartile range (IQR). The second quartile (median) mark is shown as a red line within this box. The ‘whiskers’ in these plots subtracts/adds 1.5 times IQR to the first/third quartile on the lower/upper side, where the distribution between the two whiskers accounting for over 99% of the distribution (assuming the data follows a normal distribution). Further, the box is ‘notched’ on the sides, where the height of the notch

gives the 95% confidence interval around the median. Although this is not a formal statistical test, the non-overlap comparing two boxes (dealing with the same data) is often used as *strong evidence* that their medians are statistically different (Chambers, 1983).

4.8.1 Selection of Domains from the SCOP Database

Pairs of structural domains were randomly selected to vary along the hierarchical groups defined by SCOP (Murzin et al., 1995): Class, Fold, Superfamily and Family. This results in a collection of 500 domain pairs for each of the 5 levels of the SCOP hierarchy yielding 2500 SCOP domain pairs from 3000 unique domains. According to SCOP (see Section 2.1.4), domains sharing a family level relationship are most likely to have descended from a common ancestral domain and, hence, are very close in their structural distance. Domains related up to the superfamily level are also likely to be evolutionarily related, although their structures are often diverged moderately (or more). Domains sharing a common fold share a common structural core made up of major secondary structures that have a preserved geometry of interactions between them. At a class level, domains do not share any major structural relationship beyond the level of widely-prevalent standard supersecondary structures.

The method used to make this selection and a complete list of these 2500 domain pairs can be found in Appendix A. These data are used for both of the experiments below, and are also often used by experiments in later chapters.

4.8.2 Experiment 1: Benchmarking Against the SCOP Hierarchy

This first experiment tests the ability of the ten different alignment quality measures to differentiate between protein domains with a varying closeness of structural relationship. To do this, a data set of 2500 SCOP domain pairs were randomly selected (see Appendix A) according to the SCOP hierarchy: Family, Superfamily, Fold, Class, and Decoy (see Section 2.1.4). Where a decoy refers to a domain that is not in the same class as its associated, randomly selected pair. For each domain pair, an alignment is generated using the popular structural alignment methods DALI (Holm and Sander, 1993), TM-Align (Zhang and Skolnick, 2005b), LGA (Zemla, 2003), CE (Shindyalov and Bourne, 1998), and FatCat (Ye and Godzik, 2003). Each of these alignment programs produces sets of 2500 alignments each (*i.e.*, 500 for each level of the SCOP hierarchy). Each set is then scored with each of the nine scoring functions mentioned above plus the information measure. Note that this experiment seeks to compare scoring functions rather than alignment programs. The alignment programs simply serve as a method to supply various alignments to the scoring functions.

Note that for the information measure, the compression gained (in bits) over the null model message length is shown, that is, the $(I_{\text{null}}(\langle S, T \rangle) - I(\mathcal{A}, \langle S, T \rangle))$ message length. Thus, the greater the compression, the better the alignment. In contrast, when using raw message lengths $(I(\mathcal{A}, \langle S, T \rangle))$, the shorter the message length, the better the alignment.

The following two figures (Figure 4.8 and Figure 4.9) show notched box-and-whisker plots of the results from these comparisons. Each figure has 5 rows of box-and-whisker plots, one for each scoring function used to compute the alignment score. Each column corresponds to the alignment program used to generate the alignments. The plots can be visually compared by looking across a row. Each notched box-and-whisker plot displays the numerical scores (as quartile marks) produced by an alignment method and scoring function pair, over the 5 groups of approximately 500 alignments each. The size of the notch indicates the 95% confidence

interval for the median score. Note that each scoring function has a different range of possible values and therefore, looking down a column the y -axis scales will be different.

A cursory inspection of these box-whisker plots indicates that, *for the given alignments* which might not be the best/optimal ones,[§] all scoring functions consistently differentiate between the SCOP groups to some extent. However, none of the scoring functions can be said to separate the SCOP groups cleanly nor to be clearly better than the others. This probably reflects the fuzzy classification boundaries of SCOP, and partly the quality of the (potentially sub-optimal) alignments of domain pairs generated by these popular alignment methods. For example, the authors of **TM-Score**, Xu and Zhang (2010), claim that the numerical score of < 0.5 corresponds to alignments not being in the same fold. However, the box-and-whisker plot in Figure 4.8 corresponding to **TM-Align** alignments scored with **TM-Score** shows that the median score for domains that only share a Fold level relationship is 0.5 and, thus, only half the alignments of domain pairs in the same Fold classification have a **TM-Score** below 0.5. The same is true of the **DALI z-score**, which defines a significance threshold for domains in the same fold of $\text{z-score} > 2$ (Holm and Sander, 1993). In Figure 4.8, the plot corresponding to **DALI** alignments scored using the **DALI z-score**, the mean **DALI z-score** on SCOP domain pairs that share a Fold classification is 4.5, though more than 25% of these alignments generate a **DALI z-score** less than 0. On the other hand, the information measure finds that more than 75% of the alignments generated for SCOP domains with the same Fold classification by every alignment program have a compression greater than the significance cutoff of 0 bits. Furthermore, it finds the median compression for unrelated (Class and Decoy) pairs for every alignment program to be less than zero bits of compression: not significant. There are no clear significance criteria for the remaining scores.

Surprisingly, **TM-Align** generates the highest scoring Family, Superfamily, and Fold level alignments using **DALI z-score**. Unsurprisingly, **TM-Align** and **LGA** alignment programs perform best when evaluated against their own (native) scoring function (Recall that the native score for **LGA** is **LGA.S3**). Inspecting the plots for information measure compression, alignments generated by **DALI** and **LGA** achieve the greatest compression for domain pairs in the same Family and Superfamily, **TM-Align** finds the best alignments for domain pairs in the same Fold, and Class. Evaluating alignments using the **STRUCTAL_score**, **TM-Align** generates the best alignments for domains in the same Family, **TM-Align** **DALI** and **FatCat** perform similarly well on domains in the same SuperFamily, **TM-Align** performs best for domains in the same Fold, and **FatCat** performs best on domain pairs in the same Class. When inspecting the plots for the related **MI** and **SI** scoring functions, alignments produced by the **CE** program generate the best scores on domains on all classification levels when scored with **SI**, and **TM-Align** performs best at all levels using **MI**. Inspecting plots of the **SAS** score, alignments produced by the **CE** alignment program perform best on domains in the same Family, Superfamily, Fold (**FatCat** using **GSAS**), and Class. **TM-Align** generates alignments that achieve the smallest (best) **SAS** scores on unrelated decoy domains. Finally, the **GSAS** score does not clearly determine the best alignment program for domains in the Class or Decoy sets. Alignments produced by These results are summarised in Table 4.2 which shows the alignment programs that produce the highest score for each level of the SCOP hierarchy.

It is difficult to come to a conclusion regarding either the best or the worst scoring function based on these results as they do not show a consistent pattern of behaviour. However, they do indicate, on an aggregate level, that each scoring function, including the information measure described in this chapter are able to distinguish between alignments between domains at varying levels of the SCOP hierarchy. The results of this experiment do indicate a significant degree

[§]A method for optimising alignments for the information measure is discussed in Chapter 6

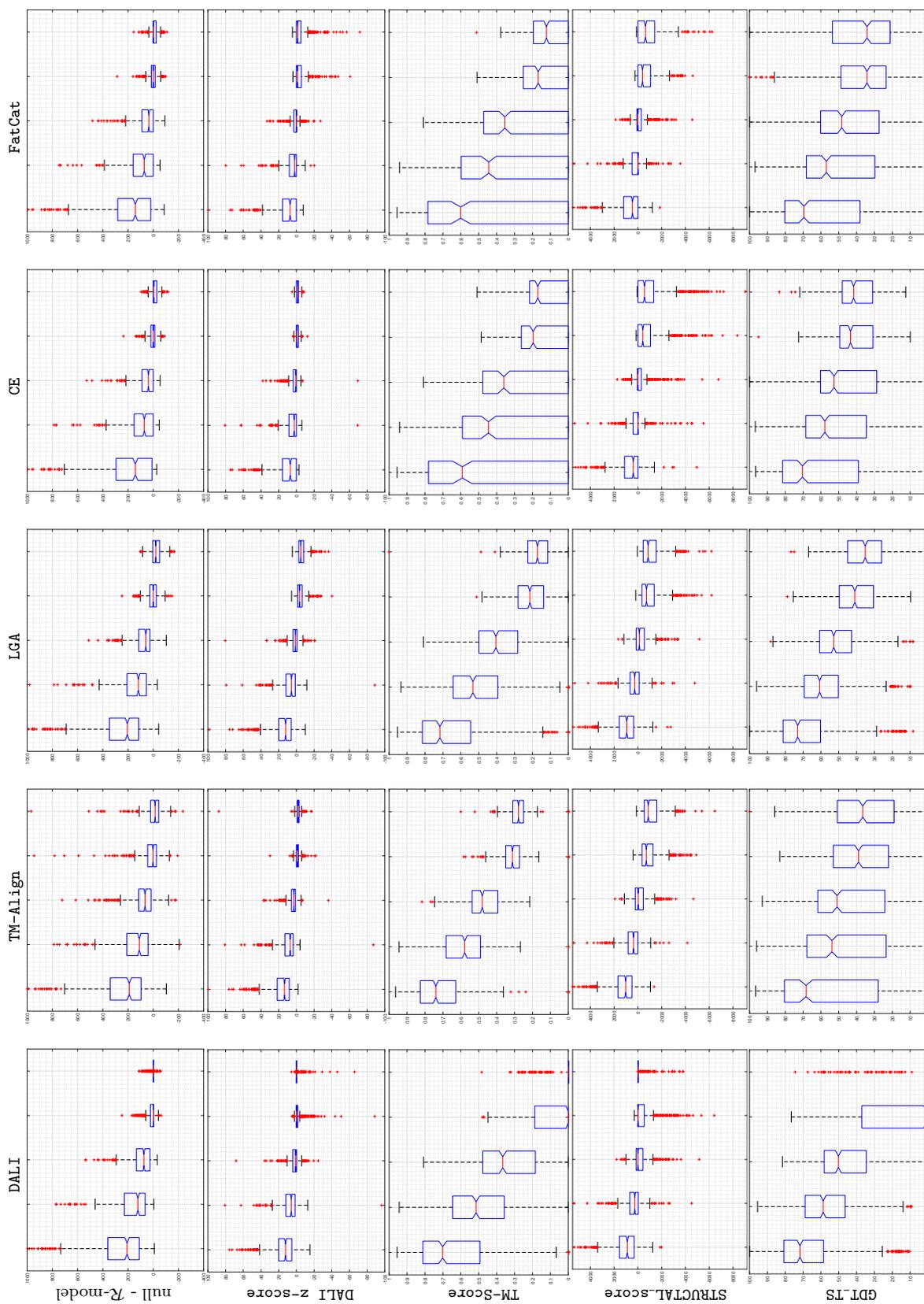


Figure 4.8: Notched box-and-whisker plots for the 2500 alignments. Columns indicate the alignment program. Rows indicate alignment quality criteria. Since the ordinate scale is equal throughout the row, the plots across various alignment programs are visually comparable. Grouped within each column (left-to-right): Family, Superfamily, Class, Decoy.

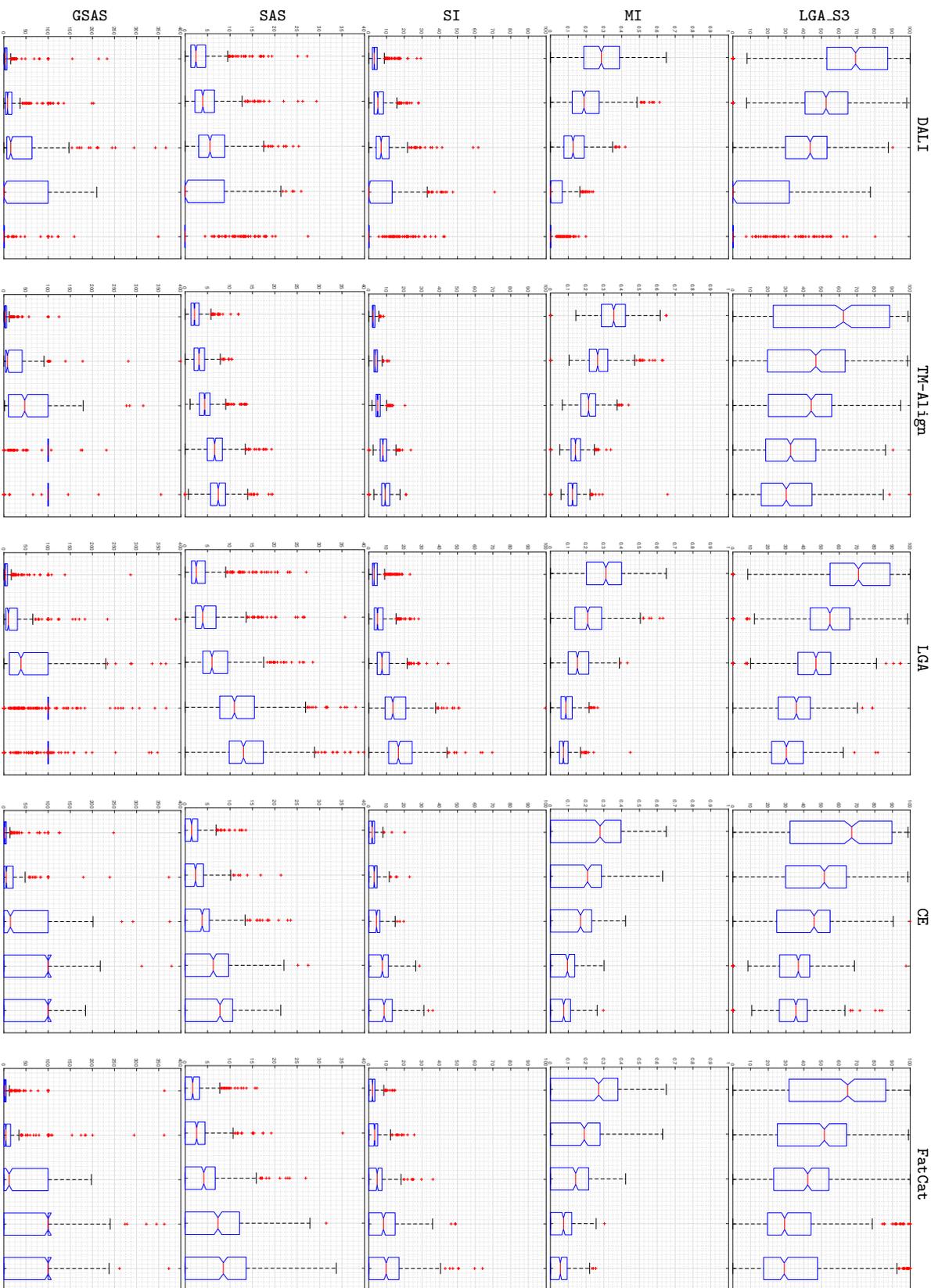


Figure 4.9: (Continued from Figure 4.8) Notched box-and-whisker plots for the 2500 alignments. Columns indicate the alignment program. Rows indicate alignment quality criteria. Since the ordinate scale is equal throughout the row, the plots across various alignment programs are visually comparable. Grouped within each column (left-to-right): Family, Superfamily, Fold, Class, Decoy.

Table 4.2: A summary of the (median) best performing alignment program (rows) according to the scoring functions (columns) for each level of the SCOP hierarchy. Detailed results are shown in Figures 4.8 and 4.9

	Information measure compression	STRUCTAL_score	DALI z-score	TM-Score	GDT_TS	LGA_S3	MI	SI	SAS	GSAS
Family	LGA	TM-Align	TM-Align	TM-Align	LGA	LGA	TM-Align	CE	CE	CE
Superfamily	DALI	TM-Align	TM-Align	TM-Align	LGA	LGA	TM-Align	CE	CE	CE
Fold	TM-Align	TM-Align	TM-Align	TM-Align	LGA	LGA	TM-Align	CE	CE	FatCat
Class	TM-Align	FatCat	DALI	TM-Align	CE	CE	TM-Align	CE	CE	–
Decoy	DALI	CE	DALI	TM-Align	CE	CE	TM-Align	CE	TM-Align	–

of disagreement between the respective scores. The degree of disagreement is quantified in the next experiment below.

4.8.3 Experiment 2: Level of Disagreement Between Measures of Alignment Quality

This experiment seeks to quantify the degree of disagreement between the structural alignment scoring functions. In experiment 1 (see Section 4.8.2) above, five structural alignment programs were used to generate 2500 alignments each between randomly selected SCOP domain pairs. Each of the 10 scoring functions were used to quantify the quality of these alignments. This allows the ranking of each list of 2500 alignment produced by a particular alignment program according to the opinion of each alignment quality scoring function. The ranking opinion of each scoring function can then be compared against the opinion of the other scoring functions. This facilitates a comparison of agreement on ranking of the best (or *top-k*) alignments by the various alignment quality scores used for benchmarking in this chapter.

Several measures exist to compare *top-k* ranked lists (Fagin et al., 2003). Some of these are introduced later in Section 7.1, which also proposes a new, information theoretic measure of the disorder between *top-k* lists based on similar concepts that underpin this chapter. Two other metrics are used to compare these ranked lists of alignments: one based on Spearman’s ρ rank correlation coefficient (Spearman, 1904) and another based on Kendall’s tau distance (Kendall, 1938). Both are slight modifications of the original measures which were proposed by Fagin et al. (2003).

A selection of alignment rank comparisons is shown in Figure 4.10.[¶] These plots show the correlation scores for the top ranked alignment in various combinations of alignment program and alignment scoring function. The size of the comparison is varied from $k = 1$ which compares only the top alignment ranked by each of the scoring functions, to $k = 25$ which compares the top 25 alignments ranked by each of the various scoring functions. The height (on the *y*-axis) of the lines within each plot can be interpreted as the level of disagreement for a given *top-k* ranking, while the slope can be interpreted as indicating the general level of agreement. Alignment scoring functions that generate a similar ranking of alignments produce flatter plots. Note that rankings that are in complete disagreement generate plots that increase quadratically using Spearman’s rho and Kendall’s tau, but linear using the information cost.

The selected rank comparisons in Figure 4.10 are divided into three sections. The left-hand column is devoted to showing scoring functions that disagree substantially on how alignments generated by a particular protein structural alignment program should be ranked. The right-hand column shows scoring functions that substantially agree on their ranking for a given

[¶]There are 225 combinations of ranked lists that can be compared. All plots can be found online at http://lcb.infotech.monash.edu.au/~jhc011/scop_ranking_plots/.

alignment program. The middle-column shows shoring functions with rankings between the above extremes.

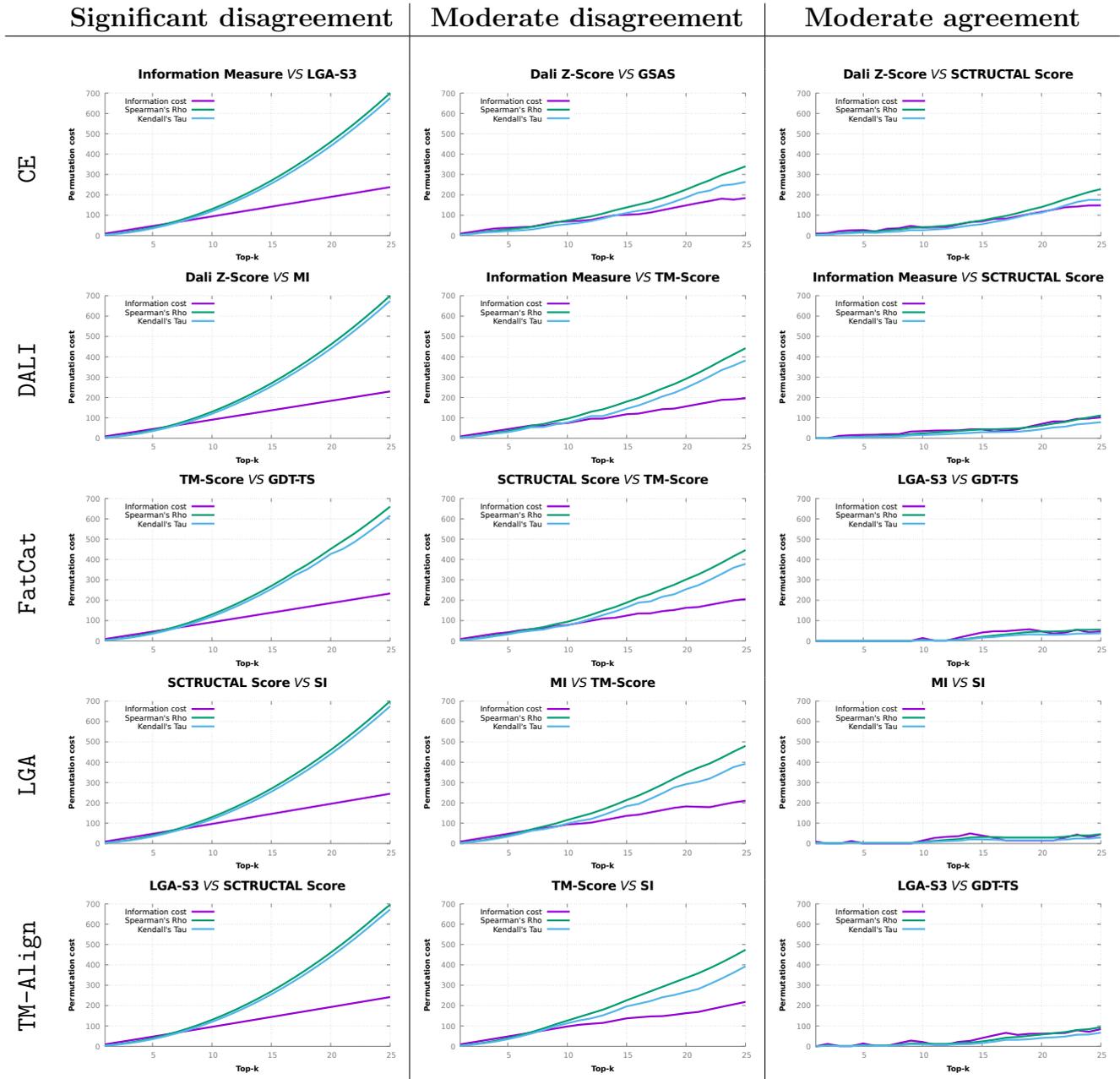


Figure 4.10: Some selected examples of plots showing quantifications of the agreement between structural alignment quality scores for the top 25 alignments between SCOP domain pairs. The ranking is performed by the alignment quality scoring functions: the information measure, DALI z-score, TM-Score, LGA_S3, GDT_TS, STRUCTAL_score, MI, SI, SAS, and GSAS, on alignments computed by the structural alignment programs DALI, TM-Align, LGA, CE, and FatCat. Disagreement in ranking is quantified by three measures: the Information Cost, Spearman’s ρ correlation coefficient, and Kendall’s τ correlation coefficient (see Section 7.1).

The results from this experiment further demonstrate the clear lack of consensus observed by Kolodny et al. (2005), Hasegawa and Holm (2009), and Slater et al. (2013). While there is some agreement between similarly formulated scoring functions, there is no consensus and scoring functions produce (often completely) contradictory rankings. The results of this experiment

reflect the standard dilemma human experts and scoring functions face in choosing between two conflicting objectives: coverage and fidelity of fit.

4.9 Conclusions

The importance of finding biologically meaningful structural alignments has led to the intensive development of methods for generating alignments and evaluating their quality. These structural alignment quality assessment methods largely define a function that mixes RMSD and the number of correspondences after least-squares superposition (see Table 2.2). However, these methods produce conflicting results and none has been generally accepted as clearly superior.

This Chapter explores the development of a framework for measuring alignment quality. The framework uses the information content of messages that losslessly compress the C_α atomic coordinates of a pair of protein structures, given a proposed alignment. A shorter message signifies a superior alignment. Furthermore, a method to infer the position of hinge rotated segments in a structure given an alignment is developed. This may be used in addition to a least-squares superposition to accurately assess alignment quality. The method contains *no* adjustable parameters, as it is built on a formal Bayesian principle of Minimum Message Length inference. It has three useful statistical properties for measuring alignment quality: 1) The difference between the lengths of the messages needed to transmit the pair of structures using any two alignments gives the log-odds posterior ratio. Thus, allowing a comparison of alignment quality based on message length; 2) It permits a natural test for statistical significance by comparison to the message length required to state the two structures independently (without an alignment hypothesis); and 3) It achieves an objective, formal trade-off between alignment complexity and the fidelity of fit between two structures through the alignment hypothesis based on a rigorous foundation of Bayesian statistics and MML.

The first experiment presented in this Chapter shows that the information measure is capable of distinguishing between alignments for pairs of domains that vary in their relationship with the SCOP hierarchy. In this regard it is competitive when compared against 9 other popular structural alignment scoring functions. Examination of competing alignments over many pairs of protein structures from the SCOP hierarchy is a sanity check. Since domain pairs at each level of the hierarchy vary in their structural relationship, a structural alignment quality measure should, on aggregate be able to distinguish between domain pairs at differing levels of the hierarchy. The results show that the information measure, using the encoding schemes described in this chapter, is able to consistently make this distinction. Secondly, a comparison of top- k alignment quality rankings for the SCOP domain pairs by various alignment quality scores reveals a significant degree of disagreement and contradiction. This confirms the findings of several important reviews of the field (Kolodny et al., 2005; Hasegawa and Holm, 2009; Sippl and Wiederstein, 2008; Slater et al., 2013; Ma and Wang, 2014).

There is no gold standard against which to measure alignment quality measures. This makes it difficult to conclusively determine which alignment quality score is best amongst the benchmark set of scores used in this chapter. Each scoring function performs its own ad hoc trade-off between the competing criteria of coverage versus the quality of fit to the structural data that manifests in the structural alignment problem. Conceptually however, the information measure stands out from the other scoring functions in that an objective trade-off is computed between detailed measures of coverage and fidelity. The descriptive complexity of the alignment hypothesis versus the compression of protein structural coordinate data enabled by the alignment hypothesis. These theoretical properties make the MML based framework for

measuring pairwise protein structural alignment quality a compelling choice for the development of an alignment program in Chapter 6.

A set of encoding schemes are selected (from amongst a host of alternatives) as concise estimates for each term in Equation 4.5. However, it is important to note that these may not be the most concise schemes possible. A more concise encoding method, when found, should replace the schemes suggested in this chapter. This replacement is demonstrated in the next chapter, which identifies aspects of the encoding schemes from this chapter which are sub-optimal and proposes methods to reduce their message length estimates.

Chapter 5

I-value: Assessing Alignment Quality Using Information Theory

“The best explanation of facts is the shortest.”

— C. S. Wallace (2005)

This chapter introduces systematic improvements to the set of coordinate encoding schemes used in Chapter 4. These improved encoding schemes build on the statistical models of protein directional data developed by Kasarapu and Allison (2015). The directional models form the basis of a sophisticated and efficient technique to define the relative position and local geometric context of corresponding residues, and use this to compress the coordinate data. These *improved* models for estimating the message length for terms in Equation 4.5 are more more efficient and concise than the *old* encoding schemes presented in Chapter 4. This is demonstrated on alignment benchmark datasets and on 2500 randomly selected domain pairs from the SCOP (Murzin et al., 1995) database.

This chapter culminates in the definition of a protein structural alignment scoring function, called *I*-value, based on the MML framework for alignment quality discussed in Chapter 4.

This chapter is based on the paper: Collier, J. H., Allison, L., Lesk, A. M., Stuckey, P. J., Garcia de la Banda, M., Konagurthu, A. S. (2016). Statistical Inference of Protein Structural Alignments, *In communication*. Preprint available at **URL:** <http://biorxiv.org/content/early/2016/06/02/056598>

5.1 Introduction

As discussed in Chapters 2-4, the traditional approach to formulating a protein structural alignment *scoring function* involves combining the contributions of a small number of important criteria, mainly *coverage* and *fidelity* (see Section 2.2.5). Table 2.2 showed some of the popular scoring functions that arose in the literature, highlighting their coverage and fidelity terms. A departure from this traditional approach is achieved by considering the structural alignment problem as an instance of the general class of statistical and inductive inference problems (see Section 4.3). In this light, Chapter 4 presented a statistical framework to assess protein structural alignment quality was developed based on the information theoretic criterion of Minimum Message Length (MML; see Section 3.3.3). In this framework, any structural alignment, \mathcal{A} , is a hypothesis that attempts to explain the observed C_α coordinate data of the protein structural pair in question, $\langle S, T \rangle$. The explanatory power of an alignment is estimated and quantified, using statistical models of encoding, to losslessly explain the C_α coordinate data. This chapter culminates in the definition of a protein structural alignment scoring function, called *I*-value,* based this framework, the related model (or \mathcal{R} -model) message length is defined as:

$$I\text{-value} = I(\mathcal{A}, \langle S, T \rangle) = \underbrace{I(\mathcal{A})}_{\text{First part}} + \underbrace{I_{\text{null}}(S) + I(T|S, \mathcal{A})}_{\text{Second part}} \quad \text{bits.} \quad (5.1)$$

And the null model message length is defined as:

$$I_{\text{null}}(\langle S, T \rangle) = I_{\text{null}}(S) + I_{\text{null}}(T)$$

For any proposed alignment, the computation of *I*-value and comparison with $I_{\text{null}}(\langle S, T \rangle)$ requires the estimation of component message length terms, $I(\mathcal{A})$, $I_{\text{null}}(S)$, and $I(T|S, \mathcal{A})$. In turn, the estimation of these terms requires statistical models of encoding. The development of an initial set of models was described, and the models evaluated in Chapter 4. In this chapter, new encoding models presented that substantially improve the estimation of these terms. To achieve this they use using sophisticated statistical models that, among other things, capture into a parametric form the observed distribution of protein coordinates. The improvements made in this chapter are summarised below and will be explained in detail in Section 5.2.

Improvement to the alignment encoding model, used to compute $I(\mathcal{A})$: Section 4.4 presented an adaptive encoding scheme to compute the alignment complexity term, $I(\mathcal{A})$, using a first-order Markov model. This model specifies a three-state edit machine that can execute (with associated probabilities based on the previous state) **match**, **insert**, and **delete** instructions. This three-state model is implicit in many alignment methods for biological sequences, most commonly seen in the implementation of affine gap costs (Gotoh, 1982; Allison et al., 1992). This model is extended here to ignore the effect of terminal gaps (insertions and deletions at the N-terminal and/or C-terminal ends of the proteins) in the alignment. Such alignments are commonly observed when aligning a monomeric protein with a longer dimeric protein.

Improvement to the null encoding, used to compute $I_{\text{null}}(\cdot)$: Section 4.5 presented a practical approach to compute the null model encoding length, $I_{\text{null}}(\cdot)$ of any given protein chain. However, this model of encoding assumed a simple *prior* distribution on protein

*This thesis uses the terms $I(\mathcal{A}, \langle S, T \rangle)$ and *I*-value interchangeably.

C_α coordinates. Specifically, this prior assumes that the directions[†] of C_α atoms is uniform (over a unit sphere). However, the empirical distribution of the C_α directions is non-uniform (see Figure 5.5(a)). To resolve this, this chapter modifies the null model encoding of protein chains to employ directional probability distributions (von Mises Fisher (Fisher et al., 1987) and Kent (Kent, 1982)) that capture the empirical distribution of C_α directional data into a parametric mixture model (Kasarapu and Allison, 2015; Kasarapu, 2015).

Improvement to the coordinate compression encoding, used to compute $I(T|S, \mathcal{A})$:

The message length term $I(T|S, \mathcal{A})$ (for transmitting the coordinates of T given the knowledge of the coordinates from S and the alignment \mathcal{A}) is a potential source of compression compared to the corresponding null model encoding term, $I_{\text{null}}(T)$. Therefore, the model used to compress the coordinates should build on and improve over the null model encoding. Since the null model encoding is modified in this chapter, the coordinate compression model also needs to be modified. Thus, this chapter presents a new coordinate compression model that builds on the parametric mixture model that is employed as the null model. This new coordinate compression model takes into account both the global and local structural similarity of C_α coordinates.

The remainder of this chapter is organised into two sections beginning with a description of the improvements made to the encoding schemes mentioned above. Benchmarking is performed between the alignment encoding schemes and between the null model encoding schemes. Finally, the chapter concludes with the performance benchmarking of I -value using the *improved* encoding schemes in comparison to the *old* encoding schemes. The results of this comparison show that the message length required to state coordinate data has significantly decreased.

5.2 Improved Encoding Schemes for I -value

This section describes, in detail the improvements to the encoding schemes necessary to estimate the message length terms $I(\mathcal{A})$, $I_{\text{null}}(\cdot)$, and $I(T|S, \mathcal{A})$. These improvements are compared against the corresponding encoding models described earlier in Chapter 4.[‡] These improved encoding schemes contribute to the final implementation of the I -value measure of protein structural alignment quality as presented in this thesis.

5.2.1 Improvement to the Alignment Encoding Model: $I(\mathcal{A})$

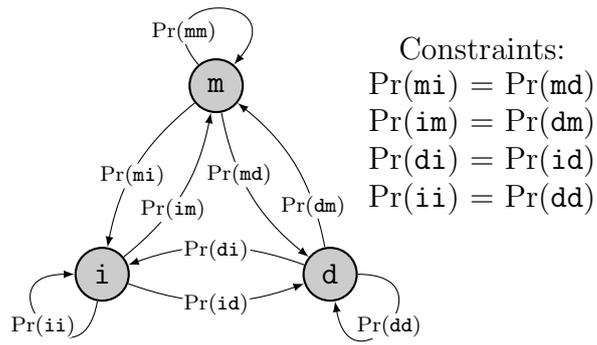
As described in Section 4.4, $I(\mathcal{A})$ was previously computed using a first-order Markov model based on an adaptive code. Briefly, this model treats any order-preserving alignment \mathcal{A} as a string produced by a three-state machine, which at each iteration produces one of the following state symbols: **match** (m), **insert** (i), and **delete** (d). See Figure 5.1.

This model has nine possible *transitions* between alignment states, each state transition has an associated probability. If the probabilities were known *a priori* (by the receiver), then the statement cost of each state transition is the negative logarithm of the probability of that transition (see Figure 5.1). However, in reality, the receiver does not know these probabilities

[†]Recall that any (x, y, z) coordinates of a C_α atom can be reparameterised as $(r, \langle \theta, \phi \rangle)$ using an internal (and canonical) spherical coordinate system, where $\langle \theta, \phi \rangle$ gives the direction of the C_α atom. See Section 4.5.

[‡]The rest of this chapter will use the term *improved* to refer to the encoding schemes described in this chapter, and *old* to refer to the best (final) encoding schemes presented in Chapter 4.

Figure 5.1: A three-state automata, with transition probabilities marked, used for adaptive first-order Markov encoding of an alignment. Symmetrical transition probabilities are enforced according to the constraints listed on the right. Repeated here (from Figure 4.2) for easy reference.



in advance and, therefore, the encoding method infers these probabilities adaptively (based on the information available at the receivers end).

Although this encoding scheme is highly practical and useful in estimating alignment complexity, it can lead to a large overestimation of the $I(\mathcal{A})$ term, in alignments that have long terminal gaps. Importantly, long terminal gaps are rather common when comparing and aligning proteins. As seen in Section 2.1.4, protein domains represent the fundamental evolutionary units in proteins. It has been well studied that the underlying genes coding for protein domains often undergo duplication in evolution (Björklund et al., 2006). This can lead to new evolutionary units (termed *super-domains*) that are larger and composed of single protein domains (Richardson, 1981). Aligning proteins where one or both contain a super-domain result in alignments containing long terminal gaps. Therefore, a small loss of efficiency in estimating the message length, $I(\mathcal{A})$, when no (or small) terminal gaps are present is acceptable when a large improvement can be found for the general case of alignments containing large terminal gaps.

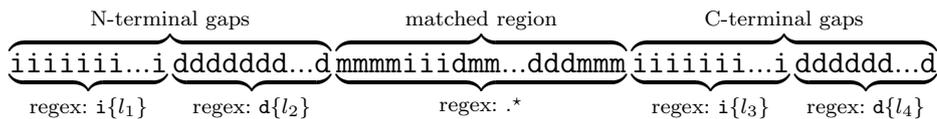


Figure 5.2: An example alignment finite-state automata string containing long terminal gaps. Such an alignment can be conceptually broken into three parts separated by the first and last match state in the whole alignment. Reading left to right, these parts are: (1) the N-terminal gaps with a regular expression of the form $i\{l_1\}d\{l_2\}$, preceding the first match, (2) the matched region between the first and last match states, and (3) C-terminal gaps with a regular expression of the form $i\{l_3\}d\{l_4\}$ succeeding the last match state. Note (l_1, l_2, l_3, l_4) represent the lengths of inserts and deletes in the N- and C-terminal parts of the alignments respectively, and they all take values ≥ 0 .

A minor improvement to the previously proposed encoding scheme makes it more expressive in the presence of large terminal gaps (leading to shorter $I(\mathcal{A})$ terms). Figure 5.2 provides an illustration that underpins the improvement described below. Any given alignment state string is split into three non-overlapping regions as follows:

1. The N-terminal gaps preceding the first match in the alignment, with $l_1 \geq 0$ inserts and $l_2 \geq 0$ deletes.

2. The matched region of the alignment between the first and the last `match` states.
3. The C-terminal gaps succeeding the last `match` state in the alignment, with $l_3 \geq 0$ `inserts` and $l_4 \geq 0$ `deletes`.

In the context of the above segmentation, consider a hypothetical communication where the transmitter wants to send the full alignment state string to the receiver. The transmitter begins by sending the set of four integers (l_1, l_2, l_3, l_4) , representing the lengths of `insert` and `delete` runs on either side the matched region. These integers are encoded using the `log*` integer code (see Section 3.4). This allows the receiver to reconstruct the alignment state string to the following extent (given as a regular expression): $i\{l_1\}d\{l_2\}.*i\{l_3\}d\{l_4\}$.

Subsequently, the transmitter can send the remaining matched region (between the first and the last `match`) using the same approach as previously described in Section 4.4. This involves sending the length $(|\mathcal{A}| - \sum_{i=1}^4 l_i)$ of the matched region using the `log*` code, followed by the state string of the matched region using the adaptive code described in Section 4.3.

To illustrate the effectiveness of this improved method over the adaptive code from Section 4.4, consider the hypothetical case of a comparison between a single domain protein (containing 100 residues), **A** versus a two domain protein (with 200 residues), **B**. Assume that the C-terminal domain of **B** `matches` exactly with **A**. The alignment state string, using a regular expression, would be of the form: $i\{100\}m\{100\}$. If only the C-terminal domain of **B** were considered, then the alignment state string would be $m\{100\}$. Now consider the $I(\mathcal{A})$ terms for these two alignments as computed by the old adaptive code. The former alignment with the long terminal gap gives $I(\mathcal{A}) = 38.2$ bits, while the latter alignment without the long terminal gap gives $I(\mathcal{A}) = 24.8$ bits. Now compare these values with the improved version, where the former gives $I(\mathcal{A}) = 34.2$ bits and the latter gives $I(\mathcal{A}) = 22.8$ bits, yielding a significant improvement.

The same applies to real world alignments. Consider, for example, an alignment produced by the DALI alignment program (Holm and Sander, 1993) between the structures of succinyl-CoA synthetase from wild boar (wwPDB 1EUD-A; 305 residues; 2 domains) and the glutamate mutase enzyme from *C. Cochlearium* bacteria (wwPDB 1CCW-A; 137 residues; 1 domain). DALI aligns the N-terminal domain of wwPDB 1EUD-A with wwPDB 1CCW-A, leaving 130 gaps before the first `match` state in the alignment. The matched region extends for the remainder of the alignment and there are no C-terminal gaps. In this case, the old adaptive code encodes $I(\mathcal{A})$ in 165.1 bits, the improved version encodes $I(\mathcal{A})$ in 143.5 bits. This is a significant improvement.

Time Complexity for Computing $I(\mathcal{A})$

Given an alignment \mathcal{A} , the lengths (l_1, l_2, l_3, l_4) can be determined in $O(|\mathcal{A}|)$ -time. As seen in the previous chapter (see Section 4.4) the encoding of a substring of \mathcal{A} defining the matched region between the first and the last `match` states, takes time linear in the substring length. This also takes $O(|\mathcal{A}|)$ time in the worst case. If the sizes of the pair of structures $\langle S, T \rangle$ are such that $|S| \leq |T| = n$, the maximum value $|\mathcal{A}|$ can take is $|S| + |T| \leq 2n$. Therefore, the computational complexity of the improved $I(\mathcal{A})$ grows linearly with the sizes of the two structures.

Quantitative Evaluation of the Improvement to $I(\mathcal{A})$ Estimation

The old and improved alignment encoding scheme to estimate $I(\mathcal{A})$ are quantitatively compared using a large set of benchmark alignments provided by the SABmark (Walle et al., 2005)

database which provides 29,759 pairwise structural alignments at the time this evaluation was conducted in May, 2016.

Each alignment was encoded using the old adaptive first-order Markov scheme from Section 4.4 and the improved version of this encoding model presented above. The results are displayed as notched box-and-whisker plots to show the distribution of encoding lengths in Figure 5.3.

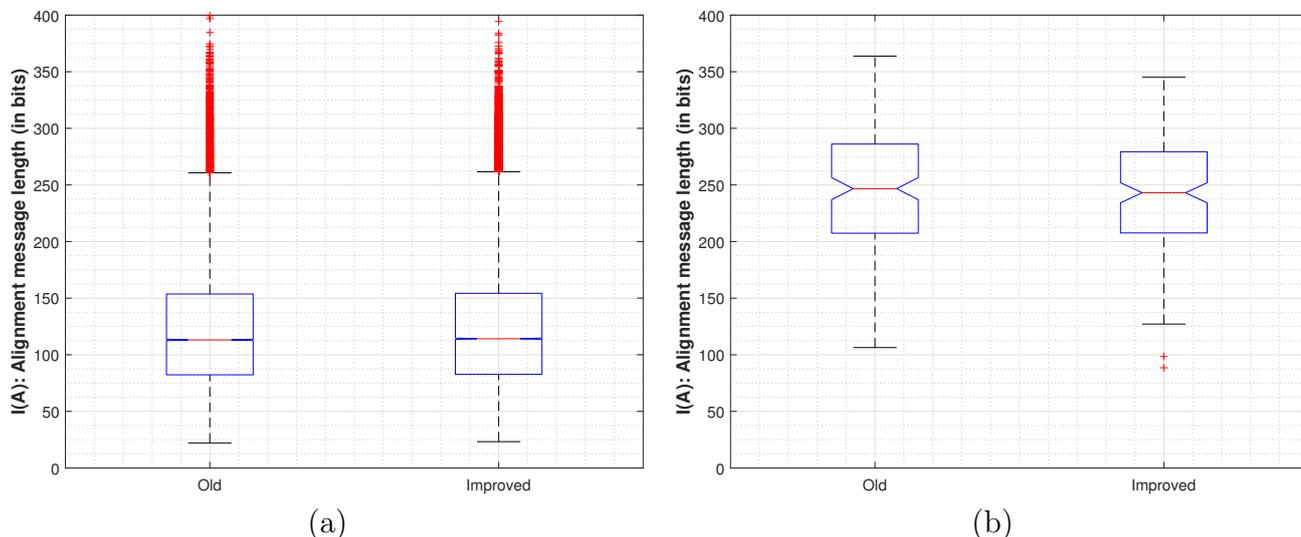


Figure 5.3: A comparison between the improved alignment encoding scheme (Improved) presented in Section 5.2.1, and the purely adaptive (Old) version of the first-order Markov alignment encoding scheme presented in Section 4.4. (a) Comparison between encoding schemes on the entire SABMark benchmark alignment set. (b) Comparison between the above alignment encoding schemes on SABMark filtered for large terminal gaps.

Based on these results it the Segmented Adaptive Markov encoding scheme provides very slightly worse compression on the entire SABMark benchmark alignment set (see Figure 5.3(a)). The Old scheme has a median encoding length for $I(\mathcal{A})$ of 111.3 bits with an inter-quartile range (IQR) of 68.09 bits. The Improved version achieves a median encoding length for $I(\mathcal{A})$ of 112.0 bits with an IQR of 68.08 bits. 0.7 bits worse. However, when filtering SABMark for long terminal gaps (Figure 5.3(b)), the difference is reversed. On this data, the Old scheme achieves a median estimate for $I(\mathcal{A})$ of 246.7 bits with an IQR of 78.3 bits in contrast with the Improved model which achieves a median estimate for $I(\mathcal{A})$ of 243.0 bits with an IQR of 71.4 bits. This is a small improvement of approximately 4 bits. The improved method overcomes the disadvantage of the old approach on alignments containing long runs terminal gaps. Therefore, **this improved encoding scheme is used by *I*-value and for the remainder of this thesis.**

5.2.2 Improved Estimation of the Null Model Message Length: $I_{\text{null}}(\cdot)$

This section describes a new null model scheme to encode protein C_α coordinate chains which modifies the corresponding encoding scheme defined in Section 4.5. Recall that, for an ordered set of C_α coordinates within a protein chain, the Cartesian coordinate of each C_α , $\vec{v} = (x, y, z)$ was transformed into $\vec{v} = (r, \langle \theta, \phi \rangle)$, an internal and canonical spherical coordinate system, where $\langle \theta, \phi \rangle \equiv \hat{v}$ represents the direction of the C_α vector on a unit-sphere.

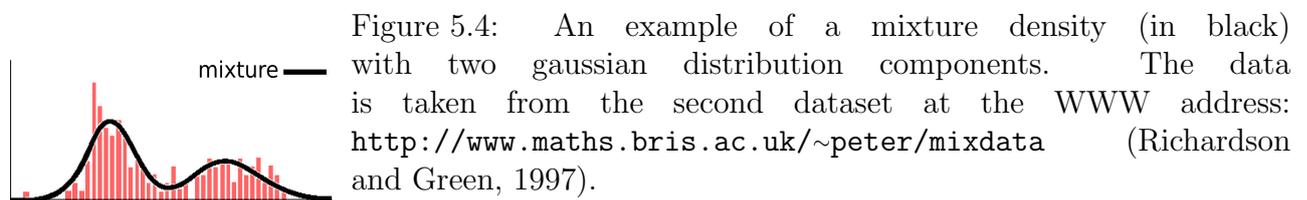
The encoding scheme for the null model description of coordinates in Section 4.5, transmitted (naïvely) the direction of each C_α over a uniform distribution. However, in reality the empirical distribution of directions between C_α coordinates is anything but uniform (see Figure 5.5(a)).

Recently, using an MML-based unsupervised learning approach, (Kasarapu and Allison, 2015) proposed a rigorous method to infer probabilistic mixture models on directional data that is distributed on Riemannian manifolds, mainly n -dimensional sphere and torus (Kasarapu, 2016). Their unsupervised learning technique was also applied to modelling protein C_α directional data using von Mises-Fisher and Kent probability distributions. This resulted in two separate mixture models, each elegantly encapsulating into a parametric form the underlying empirical distribution of C_α directions.

The proposed improvement to the null model results from employing these mixture models, since they allow for the assessment of (1) the likelihood of any direction of a C_α coordinate, and through this (2) the null model statement cost of each coordinate along the chain. These details are explained below. However, before the improved null model encoding is described, a brief background on probabilistic mixture models is provided, along with a description of the mixture models inferred by Kasarapu and Allison (2015).

Background on Mixture Models

Often, when dealing with real-world data, it is not possible to accurately describe the distribution of this data using a single probability density function. Instead, the data should be considered to have come from a *mixture* of sub-populations (see Figure 5.4). This is a concept known as mixture modelling (McLachlan and Basford, 1987).



Formally, a mixture model is a probability density function (PDF) resulting from a linear combination of the probability densities from component distributions of the form:

$$\mathcal{M}(\hat{v}; \vec{\Theta}) = \sum_{k=1}^{|\mathcal{M}|} w_k f_k(\hat{v}; \vec{\vartheta}_k) \quad (5.2)$$

In the above equation, \hat{v} represents the random variable, \mathcal{M} represents the mixture model containing $|\mathcal{M}|$ component distributions of the form $f_k(\cdot)$. Each component PDF, $f_k(\cdot)$ (with parameters $\vec{\vartheta}_k$) is weighted with a probability, w_k , according to the proportion of the data it explains, such that $\sum_{k=1}^{|\mathcal{M}|} w_k = 1$. Finally, $\vec{\Theta}$ represents the vector of mixture model parameters $\vec{\Theta} = \{|\mathcal{M}|, \{(w_k, \vec{\vartheta}_k)_{\forall 1 \leq k \leq |\mathcal{M}|}\}\}$.

Mixture Models of Kasarapu and Allison (2015)

Recent work by Kasarapu and Allison specifically deals with the modelling of multi-modal directional $\langle \theta, \phi \rangle$ data from protein C_α coordinates. In particular, the work considered the MML estimation of probabilistic mixture models containing component PDF terms composed of either entirely three-dimensional (3D) von Mises-Fisher (vMF) distributions or entirely Kent

distributions (Fisher et al., 1987; Kent, 1982). This resulted in two separate mixture models, each of which proposes a probability distribution for the directional data of protein C_α atoms along any protein chain. Specifically:

vMF mixture model (Kasarapu and Allison, 2015), represented as \mathcal{M}_{vMF} is a model with 35 component von Mises-Fisher probability density functions. Each vMF distribution defines contours of equal probability densities that are circular on the surface of a three-dimensional unit-sphere. Using the notations in Equation 5.2, each f_k in this mixture model is a probability density function with *three* free parameters of the form:

$$f(\hat{v}; \vec{\vartheta}) = \frac{1}{c(\kappa)} \exp(\kappa(\hat{\mu} \cdot \hat{v}))$$

where \hat{v} is the random variable, $\vec{\vartheta} = (\hat{\mu}, \kappa)$ is a vector of parameters containing the mean direction $\hat{\mu}$ and the concentration parameter κ , and $(\hat{\mu} \cdot \hat{v})$ defines the dot product between the two unit vectors. Finally, $c(\kappa)$ is the normalisation constant of the PDF defined as $c(\kappa) = \frac{4\pi \sinh(\kappa)}{\kappa}$, where $\sinh(\cdot)$ is the hyperbolic sine function.

Kent mixture model (Kasarapu, 2015), represented by $\mathcal{M}_{\text{Kent}}$ is a model with 23 component Kent probability density functions. A Kent distribution generalises vMF distribution and defines contours of equal probability densities that are elliptical (rather than circular) on the surface of the unit-sphere. Again, using the notations in Equation 5.2, each f_k in this mixture model is a probability density function with *five* free parameters of the form:

$$f(\hat{v}; \vec{\vartheta}) = \frac{1}{c(\kappa, \beta)} \exp(\kappa(\hat{\gamma}_1 \cdot \hat{v}) + \beta [(\hat{\gamma}_2 \cdot \hat{v})^2 - (\hat{\gamma}_3 \cdot \hat{v})^2])$$

where \hat{v} is the random variable, $\vec{\vartheta} = (\hat{\gamma}_1, \hat{\gamma}_2, \hat{\gamma}_3, \kappa, \beta)$ is a vector of parameters containing an orthogonal system of unit vectors, $(\hat{\gamma}_1, \hat{\gamma}_2, \hat{\gamma}_3)$, that require only three parameters to define, concentration parameter, κ , and a parameter that controls the eccentricity of the elliptical contours β . The various terms of the form, $(\hat{\gamma}_i \cdot \hat{v})$ (for $1 \leq i \leq 3$) define the various dot products. Finally, $c(\kappa, \beta)$ is the normalisation constant of the PDF defined as:

$$c(\kappa, \beta) = 2\pi \sum_{i=0}^{\infty} \frac{\Gamma(i + \frac{1}{2})}{\Gamma(i + 1)} \beta^{2i} \left(\frac{2}{\kappa}\right)^{2i + \frac{1}{2}} I_{2i + \frac{1}{2}}(\kappa)$$

where $\Gamma(\cdot)$ are Gamma functions, and $I_k(\cdot)$ is the modified Bessel function of the first kind of order k (Mardia and Jupp, 1999).

The fidelity of these two mixture models to accurately represent, in a parametric form, the entire empirical distribution of directional data of protein coordinates being modelled is shown in Figure 5.5. The individual components are represented in Figure 5.5(d-e), where contour lines encompassing 80% of the probability mass of each of the components are *projected* on a (θ, ϕ) -plane.

Note that there are two prominent peaks visible in Figure 5.5(a). The tallest at approximately $\theta = 60^\circ$, $\phi = 90^\circ$ corresponds to C_α coordinates belonging to helical secondary structures. The smaller peak at approximately $\theta = 120^\circ$, $\phi = 300^\circ$ corresponds to strand secondary structures (see Section 2.1.1).

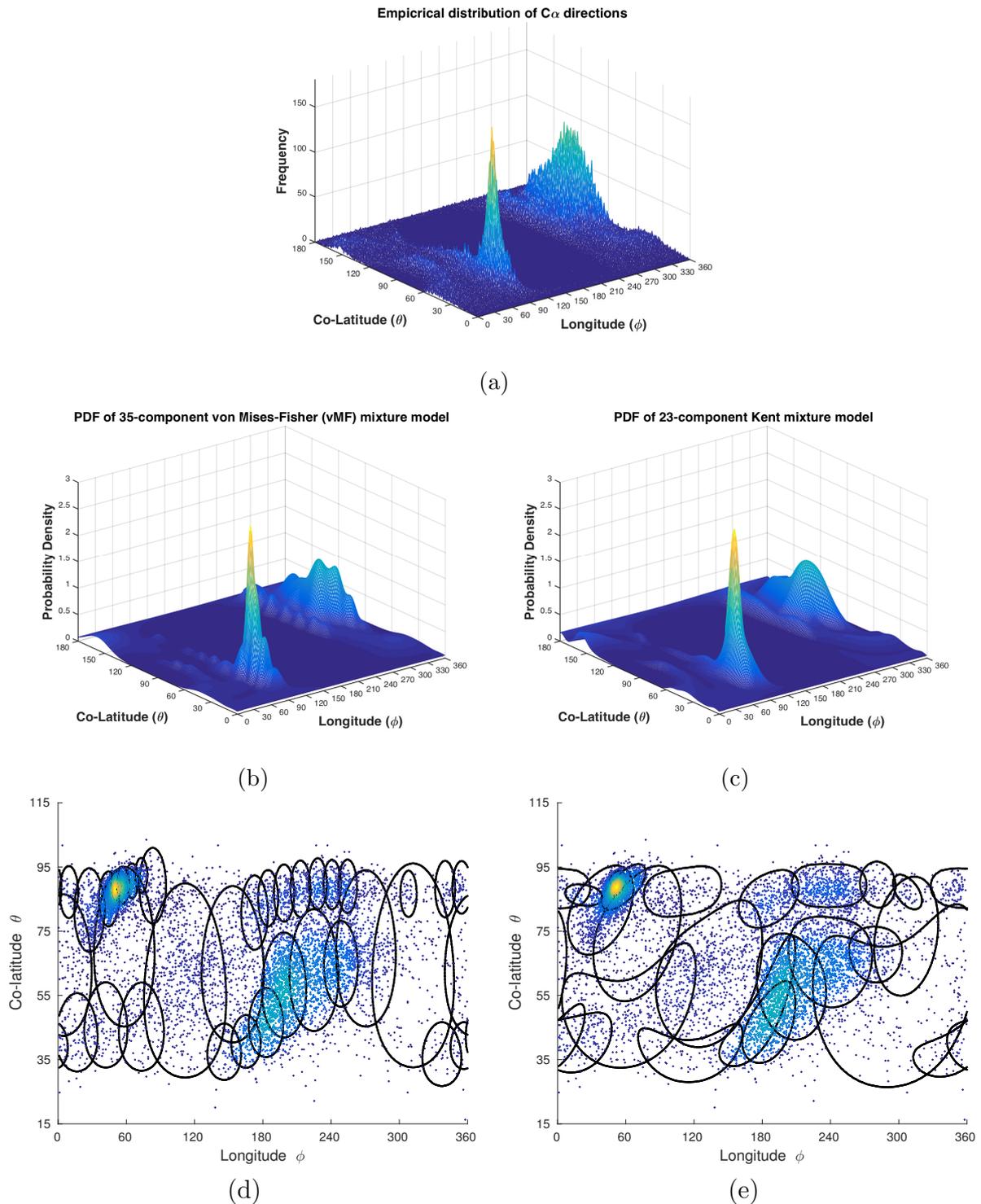


Figure 5.5: Fidelity of the 35-component vMF mixture model and the 23-component Kent mixture models. (a) The empirical distribution of directional data of C_α atoms. The 3D plot shows the frequencies of the observed directions of C_α atoms, $\langle \theta, \psi \rangle$ denoting \langle co-latitude, longitude \rangle . (b) Shows the probability density defined by the 35-component vMF mixture model (Kasarapu and Allison, 2015). (c) Shows the probability density defined by the 23-component Kent mixture model (Kasarapu, 2015). Contour lines on (d) vMF mixture components and (e) Kent mixture components overlaid on empirical protein directional data. One equi-probability-density contour is shown (in black) for each of the components in the respective mixture models. Each contour encloses 80% of the associated component's probability mass. Note that the black contours are distorted due to planar projections of circular (for vMF) and elliptical (for Kent) contours on a spherical surface.

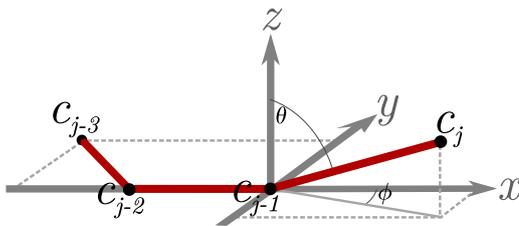


Figure 5.6: The direction $\langle \theta_j, \phi_j \rangle = \langle \text{co-latitude}, \text{longitude} \rangle$ of any C_α atom \vec{c}_j is computed as follows: the coordinates of the tetramer $\{\vec{c}_j, \vec{c}_{j-1}, \vec{c}_{j-2}, \vec{c}_{j-3}\}$ are translated and rotated such that c_{j-1} is at the origin, c_{j-2} lies on the negative x -axis, and c_{j-3} lies on the $-x, +y$ quadrant of the xy -plane. The co-latitude θ_j of \vec{c}_j is then defined by measuring the angle between \vec{c}_j and the z -axis. The longitude ϕ_j is measured as the angle between the projection of \vec{c}_j onto the xy -plane and the x -axis.

This chapter employs each of these mixture models (\mathcal{M}_{vmf} and \mathcal{M}_{kent}) to define an improved null model encoding scheme. They are compared against the old null model encoding scheme (see Section 4.5) later in this section.

Null Model Encoding of Protein Chains Using Directional Mixture Models

For any protein chain containing C_α coordinates represented as $C = \{\vec{c}_1, \vec{c}_2, \dots, \vec{c}_n\}$, the improved null model encoding uses the following procedure. The number of coordinates in C is first transmitted over a \log^* integer code, as in Equation 3.4, taking $I_{\log^*}(|C|)$ bits. This is followed by the transmission of coordinates, using either \mathcal{M}_{vMF} or \mathcal{M}_{Kent} mixture models. Before this, the first three coordinates $\{\vec{c}_1, \vec{c}_2, \vec{c}_3\}$ are transmitted exactly as previously carried out in Section 4.5.[§]

The transmission of each subsequent C_α atom, $\vec{c}_j \equiv (x_j, y_j, z_j)$, is performed by transforming it into the equivalent $(r_j, \langle \theta_j, \phi_j \rangle)$ (relatively-defined) spherical coordinates, where r_j is the distance (or radius) of \vec{c}_j defined from the previous C_α coordinate in the chain, \vec{c}_{j-1} , and the pair, $\langle \theta_j, \phi_j \rangle$ gives the direction (\hat{c}_j) of \vec{c}_j , measured relative to the canonical orientation defined by the preceding three transmitted coordinates, $\vec{c}_{j-1}, \vec{c}_{j-2}, \vec{c}_{j-3}$. The canonical orientation is illustrated in Figure 5.6.

The distance, r_j , is transmitted as before (see Section 4.5), based on the observation that the successive C_α - C_α distance is highly constrained (see Section 2.1.5). Therefore, each r_j can be efficiently encoded over a Normal distribution with parameters $\mu \pm \sigma = 3.8 \pm 0.2\text{\AA}$, to a precision of statement of $\epsilon = 0.001$ (see Section 3.3.3). Using this scheme, Equation 4.6 gave the length of the code required to transmit r_j using this method as:

$$I_{\text{radius}}(r_j) = -\log_2(\epsilon \cdot \mathcal{N}(r_j; \mu, \sigma)) \quad \text{bits.}$$

The receipt of this distance information by the receiver reduces the uncertainty regarding the position of \vec{c}_j by constraining it to be on the *surface* of the sphere with a radius of r_j , and centered at \vec{c}_{j-1} . For the receiver to fully determine the position of \vec{c}_j the direction, $\langle \theta_j, \phi_j \rangle$, of \vec{c}_j must still be encoded.

Previously in Section 4.5, each direction, $\langle \theta_j, \phi_j \rangle \equiv \hat{c}_j$, was encoded under the assumption that it is uniform on the surface a unit sphere. In this chapter, each direction \hat{c}_j can be encoded using one of the \mathcal{M}_{vMF} and \mathcal{M}_{Kent} mixture models. From Equation 5.2, the length of the code

[§]This is because the encoding scheme here requires a context from the preceding three coordinates.

required to specify the direction, \hat{c}_j , is computed as:

$$I_{\text{direction}}(\hat{c}_j \equiv \langle \theta_j, \phi_j \rangle | \mathcal{M}) = -\log_2(\epsilon^2 \cdot \mathcal{M}(\hat{c}_j; \vec{\Theta})) = -\log_2(\epsilon^2 \cdot \sum_{k=1}^{|\mathcal{M}|} w_k f_k(\hat{c}_j; \vec{\vartheta}_k)) \quad \text{bits.} \quad (5.3)$$

where \mathcal{M} is either \mathcal{M}_{vmf} or \mathcal{M}_{kent} , and $\epsilon = 0.001$ is the precision of statement of data.

Combining the statement cost to transmit the number of coordinates, and the code lengths described in Equations 4.6 and 5.3, the null model message length to transmit an entire chain of successive C_α coordinates, $C = \{\vec{c}_1, \dots, \vec{c}_n\}$, is given by:

$$I_{\text{null}}(C) = I_{\text{integer}}(n) + \sum_{j=1}^n I_{\text{radius}}(r_j) + I_{\text{direction}}(\langle \theta_j, \phi_j \rangle | \mathcal{M}) \quad \text{bits.} \quad (5.4)$$

Finally, using the above, the null model message length to transmit two structures $\langle S, T \rangle$ is simply:

$$I_{\text{null}}(\langle S, T \rangle) = I_{\text{null}}(S) + I_{\text{null}}(T) \quad \text{bits.}$$

Time Complexity of Computing $I_{\text{null}}(\cdot)$

The computation of the null model message length for any chain of C_α coordinates with n atoms requires $O(n)$ effort. This is because the computation of each $I_{\text{direction}}(\hat{c}_j | \mathcal{M})$ requires computing the likelihood of \hat{c}_j , given each component density function in the null mixture. The mixture models contain a constant ($|\mathcal{M}_{\text{Kent}}| = 23$; $|\mathcal{M}_{\text{vMF}}| = 35$) number of component probability density functions. Thus, the likelihood for each \hat{c}_j can be computed in constant time. Similarly, the computation of $I_{\text{radius}}(r_j)$ of each C_α atom is also a constant time operation. Therefore, over all n atoms, the computation of the null model message length of a given chain is $O(n)$.

Quantitative Evaluation of the Improvement to $I_{\text{null}}(\cdot)$

The null model message length derived from employing the \mathcal{M}_{vMF} and $\mathcal{M}_{\text{Kent}}$ mixture models are compared below against each other, and also against the corresponding (uniform-sphere) encoding method described previously in Section 4.5. This comparison is performed on the aggregate collection of 3000 randomly selected protein domains from the SCOP (Murzin et al., 1995) database (see Section 4.8.1). The method used to select these domains and a list of them is available in Appendix A.

For each domain in the collection, the null model message length terms ($I_{\text{null}}(\cdot)$) are computed using the three encoding schemes: (1) the old uniform-direction model (see Section 4.5); (2) the improved null model using the vMF mixture (\mathcal{M}_{vMF}); and (3) the improved null model using the Kent mixture ($\mathcal{M}_{\text{Kent}}$). Each of these message length terms are divided by the number of C_α atoms in the domain, yielding a useful descriptive statistic, average bits-per- C_α , to compare the three null encoding schemes.

The variance of this (bits-per- C_α) descriptive statistic across the three encoding schemes for all 3000 SCOP domains can be visually compared in Figure 5.7 using notched box-and-whisker plots. The most striking observation is that the uniform null model varies very little about the median value of 37.16 bits. This is because the radius (r_j) of successive C_α atoms is highly constrained around 3.8Å. Therefore, the radius terms take approximately 9.7 bits to state (see Equation 4.6). Subsequently, encoding each direction term ($\langle \theta_j, \phi_j \rangle$) uniformly on a sphere with

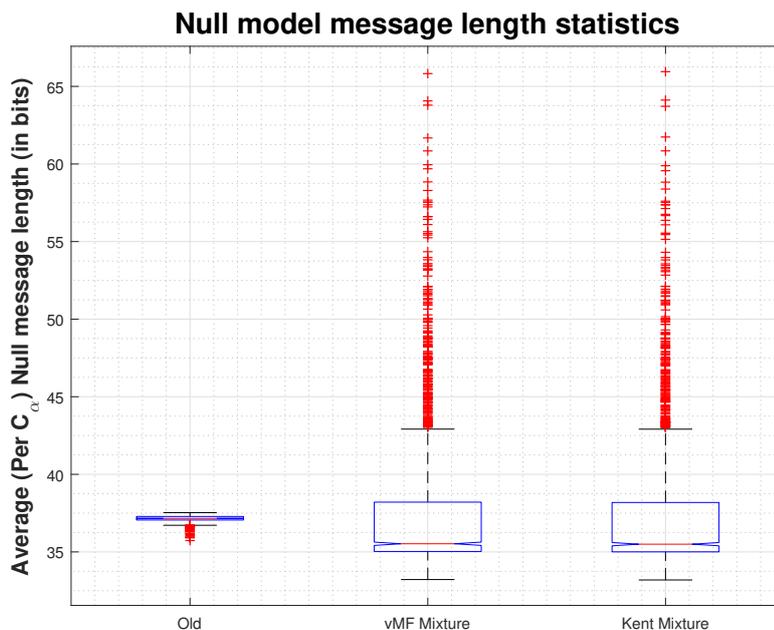


Figure 5.7: Comparison of the average $I_{\text{null}}(\cdot)$ on a per-residue basis between the improved null model encoding schemes (using either the mixture of vMF or Kent distributions) and the old (uniform-sphere) null model encoding scheme, using 3000 randomly selected SCOP domains. The y -axis shows the average message length to encode a C_α atom using the various null model encoding schemes.

radius of about 3.8\AA (see Equation 4.7) takes close to $\log_2\left(\frac{4\pi(3.8)^2}{(0.001)^2}\right) = 27.5$ bits to state. The sum of both the radius and direction statement costs has an average value of 37.2 bits-per- C_α atom using the uniform-direction null model. This result has very little variance based on the uniform direction assumption.

The improvement, as measured by the bits-per- C_α statistic, is categorical. The improved null encoding scheme, using either the \mathcal{M}_{vMF} or $\mathcal{M}_{\text{Kent}}$ mixture models, usually encodes C_α coordinate data using a shorter message than the uniform-direction null model. The mixtures are modelling the observed non-uniform and multi-modal directional distribution of consecutive C_α atomic coordinates better than the Old model. The median values for the improved null encoding scheme when using \mathcal{M}_{vMF} and $\mathcal{M}_{\text{Kent}}$ are nearly identical (35.39 and 35.37 bits-per- C_α respectively), saving about 1.8 bits over the uniform-direction null model, for every C_α atom encoded. This results in a large improvement, for example, a chain containing 100 residues could be stated, using the improved null model encoding scheme, with a message 180 bits shorter than the message length obtained by the Old uniform-direction scheme.

The interquartile range (IQR) for the mixture models is much greater (compared to the uniform-direction model) as would be expected with a non-uniform distributions of directions. This is because, over the collection of protein domains, the model is bound to encounter C_α coordinates that are distributed in high probability regions, for example, when they are a part of helices and strands, and hence require shorter message lengths. On the other hand, there will be C_α coordinates that fall within the lower probability regions (unstructured coils), yielding longer code lengths. However, as can be seen from Figure 5.7, the null encoding using mixture models is vastly more concise than the uniform-direction model in general.

Comparing \mathcal{M}_{vMF} and $\mathcal{M}_{\text{Kent}}$, the Kent mixture has the median message length, of 35.37 bits-per- C_α , with the IQR covering 2.07 bits. This is nearly identical to the message length values from the vMF mixture model with a median of 35.39 bits-per- C_α and an IQR of 2.09

bits. The almost identical performance of the two mixture models is expected, since they have been inferred by Kasarapu and Allison (2015) on the same empirical directional data. Since the Kent distribution generalises (and subsumes) the von Mises-Fisher distribution, it is able to explain the underlying data in fewer number of components (in comparison with vMF mixture), although each region of the $\theta\phi$ -plane between the two models has nearly identical probability densities (see Figure 5.5(b-c)).

5.2.3 Improved Estimation of the Coordinate Compression Model: $I(T|S, \mathcal{A})$

The coordinate compression model relies on efficiently encoding the C_α coordinates in T using the coordinate information from S (transmitted as a null model message in $I_{\text{null}}(S)$ bits; see Section 5.2.2) and the alignment information (transmitted as a three-state string in $I(\mathcal{A})$ bits; see Section 5.2.1).

As described earlier (see Section 4.6), this transmission is only concerned with the coordinates in T . This implies that the coordinate compression model (used when transmitting the coordinates in T) is only interested in the **insert** (i) and **match** (m) states in the alignment. Each C_α atom in T is either aligned with some C_α coordinate in S , or is left unaligned in the alignment. Therefore, T can be partitioned into successive blocks alternating between regions that are aligned with S and those that are left unaligned. In this case, a *block* means any run of successive C_α coordinates in T that are all either aligned or unaligned based is the \mathcal{A} .

Let each chain of successive coordinates in T corresponding to the **insert** (i) states in the alignment be denoted by \mathcal{I}_k , and let there be m such unaligned blocks of coordinates in T (of varying chain lengths). The coordinates in each (unaligned) block \mathcal{I}_k is transmitted using the null model method described in Section 5.2.2. The coordinates in the chain corresponding to each unaligned block takes $I_{\text{null}}(\mathcal{I}_k)$ bits to state. Observe that since these unaligned coordinates are stated over the null model, they offer no compression with respect to their encoding using the null model. The code length to transmit all of these runs of unaligned coordinates, \mathcal{I}_k , in T is:

$$I_{\text{unaligned}}(T|S, \mathcal{A}) = \sum_{k=1}^m I_{\text{null}}(\mathcal{I}_k)$$

Therefore, the potential source of compression when stating the coordinates in T efficiently, can only come from describing the matched C_α coordinate blocks. This builds on the information from the corresponding coordinates in S , with which they are aligned (according the information in \mathcal{A}). In the improved model, the code length to state these matched coordinates in T , using the corresponding coordinates in S , relies on a combination of two encoding schemes. The first takes into account the global structural similarity of the matched coordinates between S and T , and is called the *global* model of encoding. The second takes into account the local directional similarity of the corresponding C_α atoms in S and T , and is called the *local* model of encoding.

Encoding the Aligned Coordinates Using the Global Model

To compute the code length required to state the coordinates of C_α atoms using the global model, the following procedure is employed. The transmitter begins by performing a least-squares superposition of S and T based on the correspondences defined by the alignment \mathcal{A} . Let $\vec{t}_j \in T$ denote the coordinate being encoded using the global model, which is assigned

correspondence with $\vec{s}_i \in S$ according to the alignment \mathcal{A} . For convenience, assume \vec{s}_i and \vec{t}_j are the transformed coordinates after least-squares superposition.

The resultant set of residuals (norm of the error vectors) $\{\delta_{ij}\}$'s between the superposed pair s_i and t_j is transmitted over a chi distribution (with three degrees of freedom) using the MML method (see Section 3.3.3).[¶] The transmitter then encodes the set of distances $\{r_j\}$'s between successive t_{j-1} and t_j , as before, over a Normal distribution centered at $3.8 \pm 0.2 \text{ \AA}$ (see Equation 4.6).

The information from δ_{ij} and r_j permits the construction of two spheres, one centered at \vec{t}_{j-1} and the other at \vec{s}_i . These two spheres will intersect in a circle,^{||} thereby reducing the uncertainty of \vec{t}_j to lie on the circle of intersection. Given this, the coordinates of \vec{t}_j can be transmitted over a uniform distribution on the circle. Figure 5.8 gives an example of this encoding scheme, containing all of the elements described in this section.

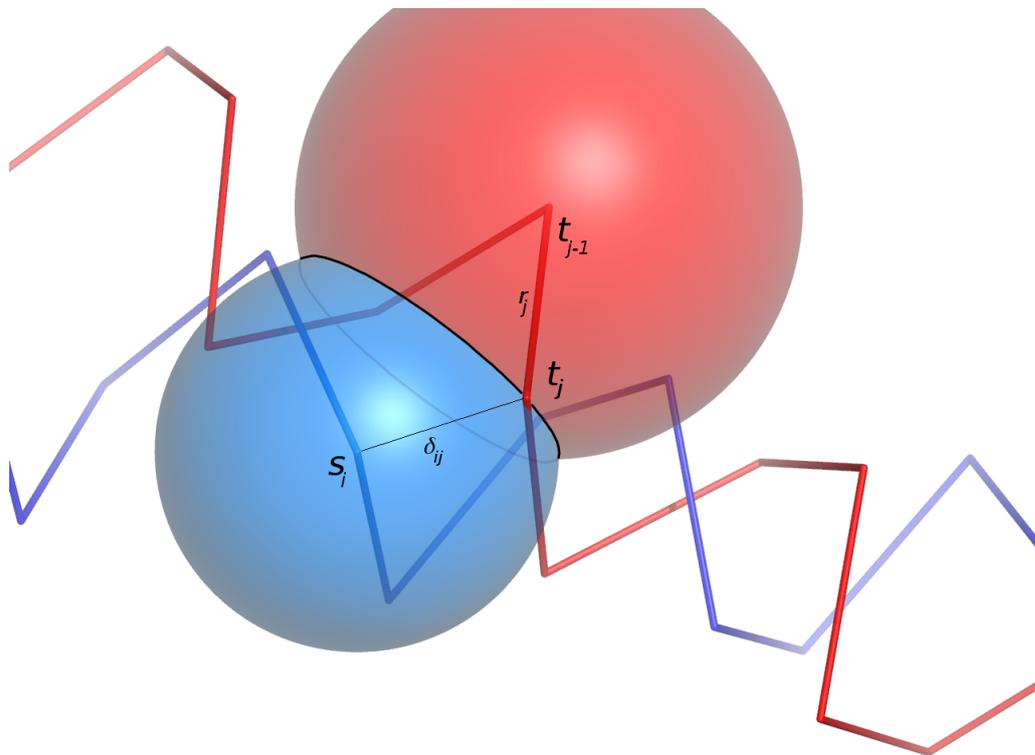


Figure 5.8: An illustration of the global model for encoding the coordinate \vec{t}_j as a deviation with respect to \vec{s}_i . The receiver is sent δ_{ij} and r_j which define a circle derived from intersecting spheres centered at \vec{s}_i and \vec{t}_{j-1} . The transmitter then states the position of \vec{t}_j on the circumference of the circle of intersection between the two spheres.

To summarise, stating each matched \vec{t}_j requires three pieces of information. First, the radius, r_j , which is stated over a normal distribution. Next, the residual δ_{ij} with respect to the corresponding \vec{s}_i stated using a chi distribution. Finally, the statement of the position of \vec{t}_j on the circumference of this circle of intersection between two spheres using a uniform distribution.

[¶]This is the most natural choice because the component-wise error terms ($\Delta x, \Delta y, \Delta z$) of each corresponding pair of coordinates, after least-square superposition, is best modeled over a symmetric 3D normal distribution $\mathcal{N}(t, \sigma)$, where σ is the RMSD after least-squares superposition. The distribution implies that the norm of the error vectors ($\sqrt{(\Delta x^2 + \Delta y^2 + \Delta z^2)}$) over all the residual terms is best modeled using the chi distribution.)

^{||}ignoring the pathological case when $\vec{t}_{j-1} = s_i$ or in the best case when $\vec{s}_i = \vec{t}_j$

The message length required to state each $\vec{t}_j \in T$, encoded using the coordinate information of its matched $\vec{s}_i \in S$, using the global model of encoding takes $I_{\text{global}}(\vec{t}_j|\vec{s}_i)$ bits.

Encoding Aligned Coordinates Using the Local Model

The local encoding model explores the compression of $\vec{t}_j \in T$ based on the similarity of the direction of its matched $\vec{s}_i \in S$. This model builds directly upon the concepts used by the null encoding model in that the position of t_j is encoded relative to t_{j-1} using a distance and a direction. This is achieved by redistributing the mixture model probability mass around the direction of \hat{s}_i . This ensures that directions near \hat{s}_i are more probable (see example in Figure 5.9). The distance, r_j , between t_{j-1} and t_j is encoded over a Normal distribution, as before (see Equation 4.6), centered at $3.8 \pm 0.2\text{\AA}$ in $I_{\text{radius}}(r_j)$ bits.

Recall from Equation 5.2 that the null mixture model is of the form $\mathcal{M} = \sum_{k=1}^{|\mathcal{M}|} w_k \cdot f_k(\vec{\vartheta}_k)$, where w_k is the weight (or probability) of the k^{th} component probability distribution. By the total probability theorem, $\sum_k w_k = 1$. In either the \mathcal{M}_{vMF} or $\mathcal{M}_{\text{Kent}}$ mixture model, each component is a directional (vMF or Kent respectively) probability density function.

If no (extra) information were available about \vec{t}_j , then the transmitter would have no choice but to encode it using the null model taking $I_{\text{null}}(\vec{t}_j)$ bits. However, extra information is available in the form of the correspondence of \vec{t}_j with \vec{s}_i . Therefore, the probability w_k of each component can be systematically updated using Bayes' theorem based on this relationship. That is, the probabilities, w_k , can be systematically updated given the expectation that \vec{t}_j is correlated with \vec{s}_i , as they are aligned.

Consider the computation of the posterior probability of the k^{th} component distribution given the direction \hat{s}_i , $\Pr(\text{Component}_k|\hat{s}_i)$:

$$\underbrace{w'_k}_{\text{unnormalised}} \Pr(\text{Component}_k|\hat{s}_i) = \frac{\overbrace{\Pr(\text{Component}_k)}^{w_k} \overbrace{\Pr(\hat{s}_i|\text{Component}_k)}^{\text{likelihood}}}{\underbrace{\Pr(\hat{s}_i)}_{\text{constant over all mixture model components}}}$$

Note that $\Pr(\text{Component}_k)$ is equal to w_k in the null model and $\Pr(\hat{s}_i|\text{Component}_k)$ is the likelihood** of the direction \hat{s}_i given the component probability density function, $f_k(\hat{s}_i|\vec{\vartheta}_k)$ using the null mixture model $\mathcal{M} \in \{\mathcal{M}_{\text{vMF}}, \mathcal{M}_{\text{Kent}}\}$. Though $\Pr(\hat{s}_i)$ (prior on the data) is unknown, it gets algebraically eliminated from the equation when the posteriors (or unnormalised w'_k) are normalised so that for the normalised posteriors $\sum_k w'_k = 1$. Therefore, each $\Pr(\text{Component}_k|\hat{s}_i)$ is normalised by dividing with $\sum_{k=1}^{|\mathcal{M}|} \Pr(\text{Component}_k|\hat{s}_i)$.

The null weights, w_k are now replaced with the new normalised posterior weights, w'_k , in the null mixture model \mathcal{M} . Thus, the coordinate \vec{t}_j can now be encoded using the posterior-reweighted mixture model component probabilities, w'_k , in the context of the matched \vec{s}_i coordinate information.

This results in a new posterior reweighted mixture model, \mathcal{M}' , to encode the direction of \vec{t}_j using the direction of \vec{s}_i . This new mixture model is as follows:

$$\mathcal{M}' = \sum_{k=1}^{|\mathcal{M}|} w'_k \cdot f_k(\vec{\vartheta}_k)$$

**Likelihood can be computed in constant time.

Using this reweighted mixture model, the direction \hat{t}_j can be encoded in: $I_{\text{direction}}(\hat{t}_j|\mathcal{M}')$ bits. Therefore, from the above and Equation 4.6, the message length required to encode $\vec{t}_j \in T$ (aligned with $\vec{s}_i \in S$) using the local encoding model is:

$$I_{\text{local}}(\vec{t}_j|\vec{s}_i, \mathcal{M}') = I_{\text{radius}}(r_j) + I_{\text{direction}}(\hat{t}_j|\mathcal{M}') \quad \text{bits.} \quad (5.5)$$

The effect of posterior-reweighting of component probabilities in the mixture model is illustrated in Figure 5.9. The left column (Figure 5.9(a)), shows 10,000 random directions sampled from the 35-component vMF mixture model inferred by Kasarapu and Allison (2015). The top row shows the planar view with contour lines from the vMF mixture model components projected onto the plot of sampled data (already shown in Figure 5.5(d)), and the bottom row shows the directional data distributed onto a sphere. Posterior reweighting causes a redistribution of probability mass around the direction, \hat{s}_i . Figure 5.9(b) shows the effect of posterior reweighting around an \hat{s}_i in part of a helical secondary structure, and Figure 5.9(c) shows posterior reweighting around an \hat{s}_i in part of a strand secondary structure.

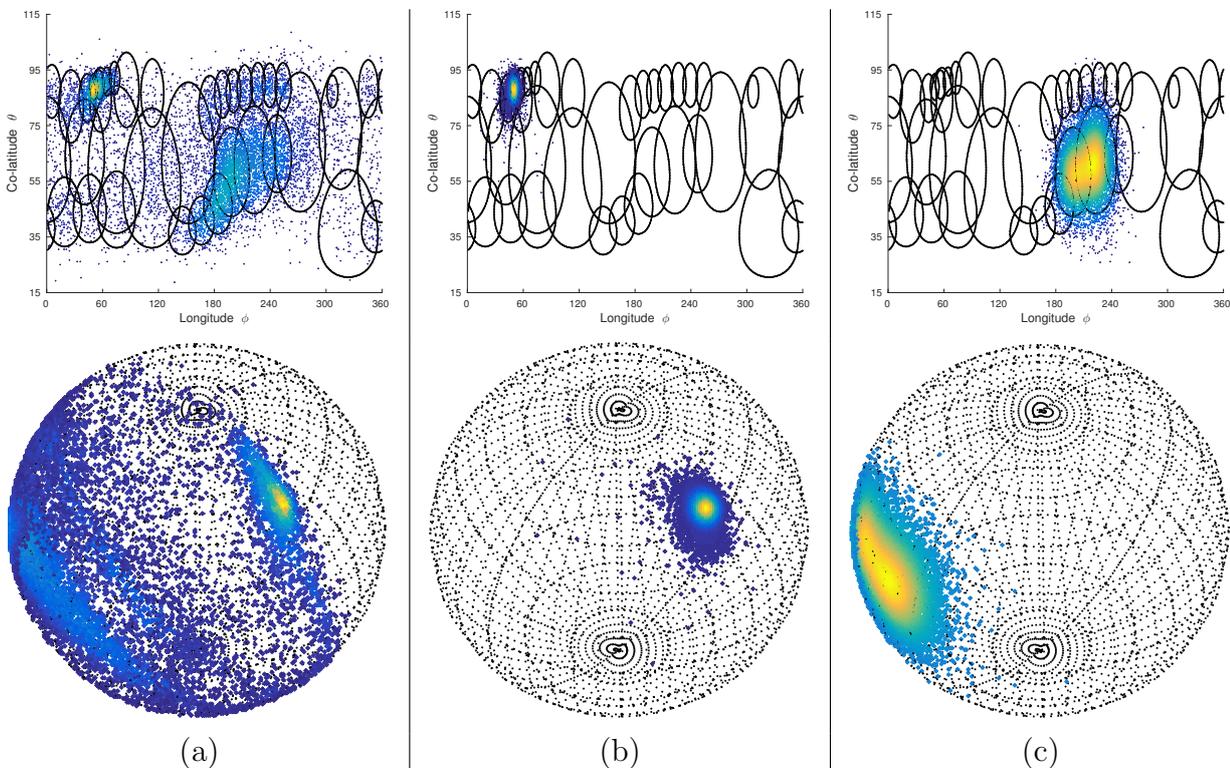


Figure 5.9: The effect of posterior reweighting on a null directional mixture distribution. The top shows sampled data (in color) from the mixture distribution with overlaid mixture component contours (in black). The bottom row shows the same sampled data projected onto a sphere. (a) Data sample from \mathcal{M}_{vMF} . (b) Data sampled from $\mathcal{M}'_{\text{vMF}}$ after posterior reweighting into a helical secondary structure. (c) Data sampled from $\mathcal{M}'_{\text{vMF}}$ after posterior reweighting into a strand secondary structure. Note the grouping of the probability mass around the expected direction.

For example, assume that the matched coordinate, $\vec{s}_i \in S$, had the direction, $\hat{s}_i \equiv \langle \theta = 90^\circ, \phi = 50^\circ \rangle$. This will result in the reweighting according to the local encoding scheme, which is shown in Figure 5.9(b). In this case, the majority of the component probability is distributed at the components near \vec{s}_i , as indicated by the tight clustering of the sampled directions from the posterior-reweighted mixture model. Another example is shown in Figure 5.9(c) where

$\hat{s}_i \equiv \langle \theta = 60^\circ, \phi = 210^\circ \rangle$. In this case, the vMF components in this area are not so tightly concentrated and thus, the sampled data from the reweighted mixture distribution are relatively more diffused compared to the previous example.

In summary, the general essence of posterior-reweighting of the component probabilities is to ensure that the probability density of the mixture model is systematically reshaped to be more focused around the observed direction, \hat{s}_i . Given that \vec{t}_j is aligned with \vec{s}_i , this will ensure that the directions near \hat{s}_i become more probable, as seen in Figure 5.9. This leads to shorter code lengths to encode \vec{t}_j . On the other hand, if \vec{t}_j is misaligned with \vec{s}_i , the directions will drift away from each other, yielding longer code lengths.

Combing the Global and Local Models to Encode Matched $\vec{t}_j \in T$

A combination of both the local and global encoding methods as described above is used to encode any aligned $\vec{t}_j \in T$. A 50%-50% two-component mixture model is defined using the global and local encoding as follows. The probability of \vec{t}_j under the global model of encoding can be computed as $\Pr_{\text{global}}(\vec{t}_j) = 2^{I_{\text{global}}(\vec{t}_j)}$. Its probability under the local model can be computed as $\Pr_{\text{local}}(\vec{t}_j | \vec{s}_i, \mathcal{M}') = 2^{I_{\text{local}}(\vec{t}_j | \vec{s}_i, \mathcal{M}')}$. This allows for the definition of a two component mixture model with equal (50%-50%) weights by:

$$\Pr_{\text{aligned}}(\vec{t}_j | \vec{s}_i) = 0.5 \Pr_{\text{global}}(\vec{t}_j | \vec{s}_i) + 0.5 \Pr_{\text{local}}(\vec{t}_j | \vec{s}_i, \mathcal{M}').$$

The code length to state each aligned \vec{t}_j can be computed from above as $\log_2(\Pr_{\text{aligned}}(\vec{t}_j | \vec{s}_i))$. For all the aligned coordinates, the code length to state these coordinates in T takes:

$$I_{\text{aligned}}(T | S, \mathcal{A}) = \sum_{\forall (\vec{s}_i \leftrightarrow \vec{t}_j) \in \mathcal{A}} \log_2(\Pr_{\text{aligned}}(\vec{t}_j | \vec{s}_i))$$

Thus, the message length term $I(T | S, \mathcal{A})$ under this improved model of encoding takes:

$$I(T | S, \mathcal{A}) = I_{\text{unaligned}}(T | S, \mathcal{A}) + I_{\text{aligned}}(T | S, \mathcal{A}) \quad \text{bits.}$$

Time Complexity of Computing $I(T | S, \mathcal{A})$

The global model of encoding requires the least-squares superposition of the matched coordinates between S and T . The maximum number of matched coordinates in any \mathcal{A} is bounded by $N = \min(|S|, |T|)$. The computational effort to find the best superposition under the least-squares measure is worst-case $O(N)$ (see Section 2.2.3).

For the global model, the computation of the code length terms associated with δ_{ij} and r_j , as well as the encoding over the circle of intersection (between spheres) can be performed in $O(1)$ (constant) time. Over all aligned coordinates, the time complexity to estimate the message length over the global model takes $O(N)$ time, as the number of matches are bounded by N . This is the same for the local encoding, as each aligned $\vec{t}_j \in T$ requires the computation of the re-weighted probabilities (w'_k 's) that take a constant $O(|\mathcal{M}|)$ effort (since the number of components in the two mixtures is a constant). The encoding of unaligned coordinates of T (encoded using the null model) has linear $O(|T|)$ complexity. Since $|T| \geq N$, the estimation of $I(T | S, \mathcal{A})$ given the coordinates of S and T and any alignment between them, \mathcal{A} , has linear, $O(|T|)$, time complexity.

5.3 Results and Discussion

The previous section introduced new encoding models for alignments and coordinate data. The new encoding schemes for alignments and the null model have been described and evaluated above in Sections 5.2.1 and 5.2.2 respectively. The compression performance between the improved models as part of *I*-value, and the old models described previously in Section 4.6 is evaluated below.

Chapter 4 compared the performance of the information measure, using the old encoding schemes (see Section 4.8) against the following nine popular scoring functions: DALI *z*-score (Holm and Sander, 1993), TM-Score (Zhang and Skolnick, 2004), MI and SI (Kolodny et al., 2005), STRUCTAL_score (Subbiah et al., 1993; Gerstein and Levitt, 1998; Levitt and Gerstein, 1998), GDT_TS and LGA_S3 (Zemla, 2003), SAS (Subbiah et al., 1993), and GSAS (Kolodny et al., 2005). These scores were computed using a large data set of alignments produced by the popular structural alignment methods DALI (Holm and Sander, 1993), TM-Align (Zhang and Skolnick, 2005b), LGA (Zemla, 2003), CE (Shindyalov and Bourne, 1998), and FatCat (Ye and Godzik, 2003). The results in this chapter are computed on the same data set. The old encoding models are compared with the improved ones using the vMF mixture model and the Kent mixture model.

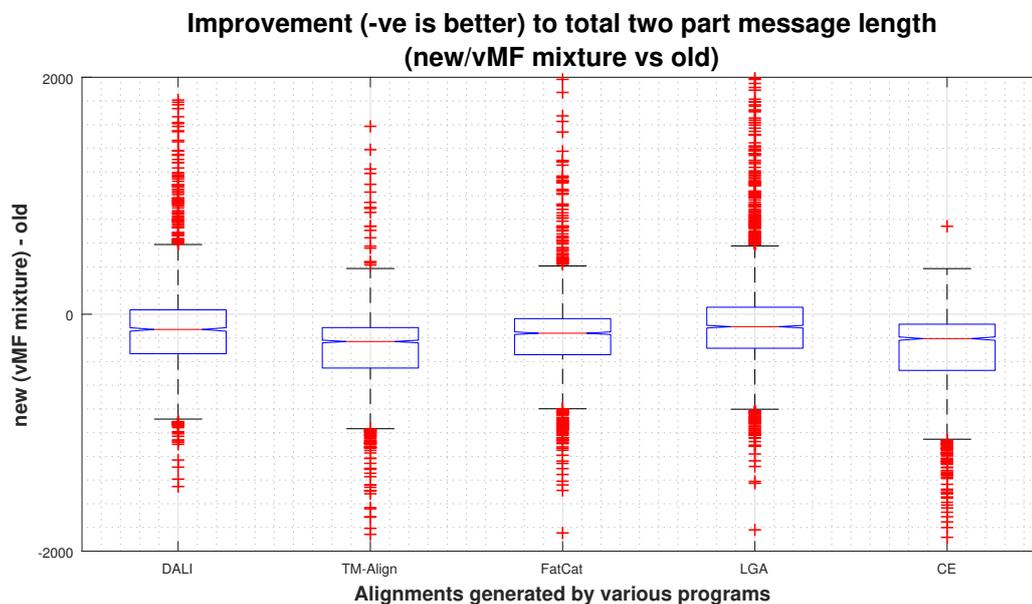
The data used in these results are the same as those used in the evaluation of encoding models in Section 4.8 (see Section 4.8.1). The procedure for extracting these domain pairs is outlined in Appendix A. Further, a table of the specific domains used is also given in Appendix A.

Each of the alignment programs listed above was used to produce a set of approximately 2500 alignments each. Each alignment is then evaluated by the old information measure from Chapter 4, and compared against *I*-value in two different configurations: *I*-value using the vMF mixture model from Kasarapu and Allison (2015) and the encoding schemes as described in this chapter; and *I*-value using the Kent mixture model from Kasarapu (2015) and the encoding schemes as described in this chapter.

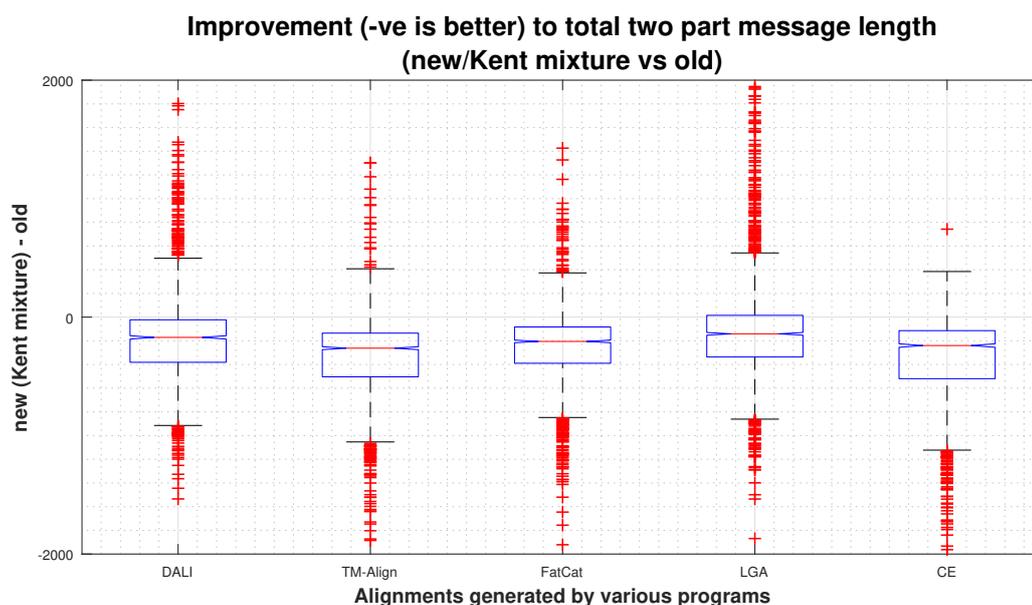
The results of this comparison are shown in Figure 5.10, which shows notched box-whisker plots for these comparisons over alignments produced by the various structural alignment programs. Figure 5.10(a) compares the message length using the old encoding schemes from Chapter 4 against the encoding schemes from this chapter, as defined in Section 5.2 using the 35-component vMF mixture. Figure 5.10(b) makes the same comparison using the 23-component Kent mixture model. The plots show the difference in total message lengths (see Equation 4.5) between the combinations of encoding schemes used. A negative value means that the improved encoding scheme has performed better by encoding a shorter total two-part message, and vice versa for positive values.

The first thing to note regarding the results in Figure 5.10 is that the new encoding schemes significantly decrease the two-part message length, $I(\mathcal{A}, \langle S, T \rangle)$, across all alignment programs, as the median reduction in total two-part message length is at least 100 bits. Therefore, the new encoding models have succeeded in more efficiently estimating the message length terms from Equation 4.5. Note however, that approximately 25% of alignments generate a total message length that is longer when using the new encoding methods. These alignments are between domains that are unrelated: Class and Decoy. In these cases the message length can be much greater, and also often longer than the null model message length.

Specifically, the median reduction of total message length using the vMF mixture model ranges from -119.9 bits for LGA alignments, up to -225.8 bits for TM-Align alignments. Except for DALI and LGA alignments, all achieved a reduction in total message length for at least



(a)



(b)

Figure 5.10: A comparison of message length estimates of the total two-part message lengths between the old encoding schemes and the improved versions using either (a) the mixture of vMF distributions, or (b) the mixture of Kent distributions. The y -axis shows the difference in total message length: negative values mean the improved encoding models produce a shorter (better) total message length; positive values mean the old encoding models produce a shorter message length.

75% of the alignments generated. The difference is even greater when examining the total message length reduction using the Kent mixture model. The reduction now ranges from -170.9 bits for LGA alignments up to -322.9 bits for TM-Align alignments. This is a significant improvement over both: the old models for encoding, and the vMF mixture using the new models for encoding.

There are two important points to take from this experiment. Firstly, the \mathcal{R} -model encoding in Section 4.6 can easily gain compression over the uniform null model defined in Section 4.5. In the worst case the old coordinate compression model is equal to the old null model encoding. This suggests that the uniform null model is too naïve a representation of protein directional data and the null directional models proposed by Kasarapu and Allison (2015) and Kasarapu (2015) are far better models of protein directional data. Secondly, the improved, \mathcal{R} -model using the mixture of Kent distributions is able to achieve greater compression over the null model than when using the mixture of vMF distributions, at the same time more heavily penalising poor correspondences. This is a desirable property because the more efficient the encoding schemes allows for a more accurate bound on the true information content.

5.4 Conclusions

At its core, the structural alignment problem depends on the trade-off between two conflicting objectives: maximising the coverage, and maximising the fidelity of fit (Irving et al., 2001). A method to objectively resolve this tension was proposed in Chapter 4 which outlined a framework for assessing structural alignment quality, an information measure (see Equation 4.5) based on MML. Chapter 4 also set out a number encoding mechanisms to estimate the code length for the three terms in Equation 4.5: $I(\mathcal{A})$, $I(S)$, and $I(T|S, \mathcal{A})$.

This chapter develops the ideas presented in Chapter 4 further by proposing and evaluating improved methods for estimating the code lengths for these terms. This includes an evolution of the best encoding methods discussed in Chapter 4. Those old encoding methods were naïve but useful in paving the way for the development of more sophisticated methods in this chapter. The main contributions of this chapter are the improved null and compression model encoding schemes based on inferred mixtures of vMF and Kent distributions from representative empirical protein data based on work by Kasarapu and Allison (2015). An improvement was also made to the code length estimate for $I(\mathcal{A})$ in the case where the alignment contains long terminal gaps. The improved encoding scheme for $I(\mathcal{A})$ introduced in this chapter is a modification to the adaptive first-order Markov scheme (see Section 4.4). It retains all of the advantages on this old scheme while solving a problem that arises when dealing with long terminal gaps which bias the alignment state transition probabilities.

These improvements are benchmarked against the set of 2500 SCOPe (Fox et al., 2013) domain pairs. The same domain pairs as used in Chapter 4 for benchmarking. From these domain pairs, a set of popular alignment programs were used to produce 12,500 alignments. The results from this benchmarking show that the mixture of Kent distributions is more efficient as part of the improved null model and \mathcal{R} -model encoding and is therefore used by all later references in this thesis to the *I*-value and its $I_{\text{null}}(\cdot)$ and $I(T|S, \mathcal{A})$ terms. Furthermore, this chapter demonstrated that the improved models for encoding are the more efficient and discriminative when discerning between closely competing alignments. In these cases also, the mixture of Kent distributions produces the shortest code lengths. Therefore ***I*-value is implemented using the Kent mixture based improved encoding models for the remainder of this thesis.**

This chapter provides a solid foundation upon which to build an alignment search algorithm. Various approaches to the search problem are explored in the next chapter, culminating in an alignment program called `MMLigner`.

Chapter 6

MMLigner: Searching for Pairwise Protein Structural Alignments

“Don’t ask for guarantees. And don’t look to be saved in any one thing”
— Ray Bradbury, Fahrenheit 451

This chapter deals with methods to search for meaningful structural alignments using the information measure of alignment quality, *I*-value, formulated in Chapter 4 and improved upon in Chapter 5. It culminates in the development of a search program, **MMLigner**, that identifies high quality and statistically significant structural alignments using the *I*-value quality measure. The program is free-software and available for download at <http://lcb.infotech.monash.edu.au/mmligner>. The consistency of **MMLigner** is demonstrated by benchmarking its alignment results against state-of-the-art alignment programs. The results produced by **MMLigner** are highly competitive, and often able to identify alignments where current programs do not. Importantly, **MMLigner** is able to consistently identify a range of significant alternative structural alignments, avoids pairing up spurious correspondences, and prefers simple alignments over complex ones. Furthermore, structural alignments generated by **MMLigner** are considered at least competitive according to popular measures of structural alignment quality and often succeeds where other alignment programs do not identify any alternative alignments.

This chapter is based on the paper: Collier, J. H., Allison, L., Lesk, A. M., Stuckey, P. J., Garcia de la Banda, M., Konagurthu, A. S. (2016). Statistical Inference of Protein Structural Alignments, *In communication*. Preprint available at **URL:** <http://biorxiv.org/content/early/2016/06/02/056598>

6.1 Introduction

The pairwise protein structural alignment problem (see Section 2.2) involves the assignment of ordered one-to-one correspondences between a subset of residues of two protein structures based on their coordinate information. Solving this alignment problem involves three major considerations:

Representation: Protein structural data are represented internally by alignment programs in multiple ways. A common representation of a protein is as an ordered set of its C_α coordinates. As seen in earlier chapters, the I -value measure was defined using this representation as it is more suitable for accurate atomic coordinate representation and superposition. Other representations of protein structures were discussed previously in Section 2.2.4.

Objective function: The structural alignment problem can be seen as a combinatorial optimisation problem, where an objective function needs to be defined to evaluate the fitness of any possible alignment. This chapter uses the I -value definition presented previously in Chapter 5 as the objective function. Recall that I -value measures the explanatory power (in bits) of any alignment to describe the C_α coordinate data of the proteins being aligned. Several other state-of-the-art objective functions have been discussed in Sections 2.2.5 and 4.2.

Search method: Given an objective function, the goal of an alignment program is to find the best alignment(s) under the stated measure of fitness that the objective function defines. Using I -value, this chapter will develop an effective search procedure that explores the alignment search space and attempts to identify alignment(s) that minimise I -value. Developing and evaluating this search algorithm forms the main focus of this chapter.

The chapter begins with a general review of common search algorithms used by popular structural alignment programs. This is followed by a description of some of the promising early directions this research took in attempting to find a reliable search method that optimises I -value. The chapter then presents an effective heuristic search method that supports the implementation of a pairwise structural alignment program called `MMLigner`. Finally, the performance of `MMLigner` is benchmarked against a host of popular protein structural alignment programs.

6.2 Structural Alignment Search Methods

The structural alignment problem is an inherently difficult computational problem (Wang and Jiang, 1994; Hasegawa and Holm, 2009). Even with the most straightforward characterisation of protein structures as contact maps (see Section 2.2.4), this problem can be reduced to the subgraph isomorphism problem (Grindley et al., 1993; Raymond and Willett, 2002; Krissinel and Henrick, 2004), a well-studied intractable (NP-hard) problem from graph theory (Garey and Johnson, 1979). This is to say, there is no known complete algorithm that can deterministically find an optimal solution in polynomial time, using most formulations of objective functions for this problem.* Therefore, heuristic search strategies are commonly employed to produce near optimal alignments in a reasonable amount of time.

*Under certain circumstances an objective function can be approximated, in a way that makes polynomial time capable of optimising for the approximation (Kolodny and Linial, 2004).

This section will provide a brief review of common search techniques used by popular structural alignment programs. For more thorough discussion, readers are pointed to the following works : Knuth (1999a); Eidhammer et al. (2000); Vesterstrøm (2006); Russell and Norvig (2009a); and Ma and Wang (2014).

6.2.1 Assembly of Well-Fitting Fragment-Pairs

Fragment-pair assembly is a widely used method to find structural alignments and it is among the most effective ones. To describe this method of finding structural alignments, consider the following definitions. A *fragment* of a protein is any contiguous region within the protein chain. That is, if a protein structure S contains the following ordered set of C_α atomic coordinates $\{\vec{s}_1, \vec{s}_2, \dots, \vec{s}_n\}$, Then a fragment defines the regions of coordinates $S_{i\dots j, \forall 1 \leq i < j \leq n} = \{\vec{s}_i, \vec{s}_{i+1}, \dots, \vec{s}_j\}$, of *length* $j - i + 1$. A *fragment-pair* refers to two fragments of equal lengths, one from each of the two proteins, S and T , being aligned. Thus, a fragment-pair identifies two fragments of the form $\langle S_{i_1\dots j_1}, T_{i_2\dots j_2} \rangle$ such that $j_1 - i_1 = j_2 - i_2$. A *well-fitting* fragment-pair implies that the two fragments are superposable within a specified threshold of RMSD; that is, $\text{RMSD}(S_{i_1\dots j_1}, T_{i_2\dots j_2}) \leq \text{specified_threshold}$.

To find a structural alignment between two protein structures S and T , the fragment-pair assembly method first requires the construction of a *library* of well-fitting fragment-pairs associated with those two structures.[†] This library is exhaustively constructed by iterating over both structures and superimposing all possible fragment-pairs of a nominated length, L , checking if they fit within the specified threshold of RMSD. Each superposition of fragments takes $O(L)$ effort (see Section 2.2.3). Therefore, constructing such a library takes $O(L \cdot |S| \cdot |T|)$ effort. Since L is often a small number (< 10), and the sizes of the structures ($|S|$ and $|T|$) are < 500 , this step can be computed in seconds on standard desktop computers.

Finally, an attempt is made to assemble this library of well-fitting fragment-pairs in such a way that they jointly superimpose with each other consistently (that is, within some specified RMSD threshold). However, exhaustively assembling fragment pairs has a combinatorial effect on the computation time required for assembly. To overcome this bottleneck, a number of dynamic programming-based heuristics (see Section 6.2.4) are used to construct an approximate assembly of fragment-pairs, while ensuring that the assembly does not violate ordering constraints. Depending on the specific technique, this is sufficient to construct an approximate alignment that is then subject to further refinements.

Using fragment pairs to construct an alignment has several advantages over dealing with individual residue pairs. Firstly, there are often fewer consistent combinations of well-fitting fragment pairs than individual residue pairs. Therefore, algorithms can quickly operate on the library of fragment pairs to rapidly produce (approximate) alignments that are good starting points for local search (see below). Also, fragment pairs contain the local context of the structures being aligned, that context is largely invisible when only considering residue pairs. Due to these advantages, this technique is used by several popular protein structure alignment programs such as WHATIF (Vriend and Sander, 1991), CE (Shindyalov and Bourne, 1998), MaxSub (Siew et al., 2000), MAMMOTH (Ortiz et al., 2002), MUSTANG (Konagurthu et al., 2006) and Fr-TM-Align (Pandit and Skolnick, 2008).

[†]Often all fragment-pairs in this library contain fragments with identical lengths. However, some methods allow variable length fragment-pairs.

6.2.2 Local Search Methods

Local search is a class of search heuristics commonly used in computer science for solving hard combinatorial optimisation problems. The search strategy involves exploring the candidate solution space by applying local changes to the current (best) solution found. These local changes are made iteratively until the search converges to a solution that cannot be improved (or some maximum allowable number of changes to the solution is reached). Depending on the strength of the objective function, local search heuristics can find good solutions to the combinatorial problem, even if they are not guaranteed to be optimal. Some of the commonly used local search techniques applied to the structural alignment problem are discussed below.

Hill Climbing Search

A commonly employed local search heuristic applied to the structural alignment problem is the *Hill climbing* technique (Russell and Norvig, 2009b). Hill climbing is an iterative refinement procedure that starts from an initial (or *seed*) alignment. The improvements are made by applying a set of *perturbation operations* to the existing state of an alignment, and accepting only the perturbation that most improves the alignment in terms of the objective function. Since such heuristics always make a locally optimal choice, they are said to be *greedy*.

Note that hill climbing heuristics find optimal solutions only if the search space over the candidate solutions (using the defined objective function) is convex. In practice, the search space of most combinatorial problems (as with the structural alignment problem) is a complex ('rugged') function. Therefore, hill climbing techniques often get trapped within a local optima, making them very sensitive to the starting solution. One approach that is often used to overcome this sensitivity is to run several searches from different starting points. A more reliable way for hill climbing to avoid getting trapped in local optima is to use *stochastic* approaches that rely on randomised acceptance of improved alignments (less greedy). This addition of randomness is fundamental to the next two local search methods described below. However, there are some programs that usefully apply a type of hill climbing approach to at least part of their search algorithm. These include MINAREA (Falicov and Cohen, 1996), LOCK (Singh and Brutlag, 1997), FASE (Vesterstrøm and Taylor, 2006), and SPalignNS (Brown et al., 2016).

Monte Carlo Search

Monte Carlo methods also rely on the iterative stochastic sampling of candidate solutions from the search space. However, unlike the greedy hill climbing heuristics described above, the acceptance criteria of the perturbed solutions in Monte Carlo methods is probabilistic. It is this probabilistic acceptance that potentially prevents the solution from prematurely converging to (certain kinds of) local optima, thus obtaining for a more extensive search of the solution space.

Many Monte Carlo search methods are adaptations of the original Metropolis-Hastings algorithm as applied to statistical mechanics (Metropolis et al., 1953; Hastings, 1970). Simulated Annealing (Kirkpatrick et al., 1983) is a practical implementation of the Metropolis-Hastings algorithm, and a general purpose stochastic optimisation technique used to find approximate solutions with intractably large and complex problem solution spaces. This method is analogous to the controlled cooling of solids to crystalline states (to minimise defects). The intuition behind actual annealing is based on the observation that particles of matter are randomly arranged at high temperatures (in gaseous and liquid states) and the gradual cooling to the

ground energy state allows those particles to arrange into a regular lattice structure. In condensed matter physics, the distribution of the random variable \mathbf{E} associated with the kinetic energy of particles at each value of temperature \mathcal{T} is characterised by the *Boltzmann distribution* of the form

$$\Pr(\mathbf{E} = E, \mathcal{T}) \propto \exp\left(-\frac{E}{\kappa_B \mathcal{T}}\right)$$

where $\exp\left(-\frac{E}{\kappa_B \mathcal{T}}\right)$ is the *Boltzmann factor*. This distribution has the effect of concentrating the probability on lower kinetic energy states as the temperature decreases. In the limit as $\mathcal{T} \rightarrow 0$, only the lowest energy states become probable (van Laarhoven and Aarts, 1987).

In order to simulate this annealing process for combinatorial optimisation problems, specifically the alignment problem, each alignment \mathcal{A} is treated to be analogous to some configuration of particles. The evaluation of that alignment using a given objective function, denoted by $Q(\mathcal{A})$, can be seen as analogous to the energy state E of the random variable in the Boltzmann distribution, and a control parameter c mimics the temperature parameter $\kappa_B T$.

Thus, the simulated annealing algorithm is run iteratively, like the original Metropolis-Hastings algorithm, starting with a high value of the control parameter c . Each alignment \mathcal{A}_i is perturbed to choose another alignment \mathcal{A}_j in its neighbourhood. If $Q(\mathcal{A}_i)$ is the evaluation of the objective function Q for the alignment \mathcal{A}_i (similarly, for $Q(\mathcal{A}_j)$), let the Metropolis criterion be defined as $\Delta Q_{ij} = Q(\mathcal{A}_j) - Q(\mathcal{A}_i)$. Then, if $\Delta Q_{ij} \leq 0$, the perturbed alignment is accepted with a probability of 1. On the other hand, when $\Delta Q_{ij} > 0$, the odds ratio of the Boltzmann-like distribution gives the probability of accepting the perturbed alignment \mathcal{A}_j with the current state of the alignment \mathcal{A}_i :

$$\Pr(\mathcal{A}_i \xrightarrow{\text{perturbed}} \mathcal{A}_j, c) = \frac{\Pr(Q(\mathcal{A}_i), c)}{\Pr(Q(\mathcal{A}_j), c)} = \exp - \left(\frac{\Delta Q_{ij}}{c}\right)$$

This process of perturbing $\mathcal{A}_i \xrightarrow{\text{perturbed}} \mathcal{A}_j$ is iterated, holding the control parameter c fixed, until the state of the alignment approaches the mode of the Boltzmann-like distribution. The control parameter is then gradually lowered (for example, c (lowered) = $0.88c$) until c approaches a small value (van Laarhoven and Aarts, 1987).

In general the simulated annealing algorithm for optimising an alignment, \mathcal{A} , using an alignment quality scoring function, Q , follows the steps set out in Algorithm 1 below.

Algorithm 1: Simulated Annealing

input : A seed alignment \mathcal{A} and a starting temperature, c .

output: An alignment optimised for the quality measure, Q .

```

1  $\mathcal{A}_i \leftarrow \mathcal{A}$ 
2 while  $c > \text{small value}$  do
3   while some number of iterations do
4      $\mathcal{A}_j \leftarrow \text{perturb}(\mathcal{A}_i)$ 
5     if  $Q(\mathcal{A}_j) \leq Q(\mathcal{A}_i)$  then
6        $\mathcal{A}_i \leftarrow \mathcal{A}_j$ 
7     else if accept with probability:  $\Pr(\mathcal{A}_i \xrightarrow{\text{perturbed}} \mathcal{A}_j, c)$  then
8        $\mathcal{A}_i \leftarrow \mathcal{A}_j$ 
9      $c \leftarrow \text{lower}(c)$ 
10 return  $\mathcal{A}_i$ 

```

The COMPARER (Sali and Blundell, 1990), DALI (Holm and Sander, 1993), and CE (Shindyalov and Bourne, 1998) programs are some example of popular structural alignment programs that use Monte Carlo methods to generate protein structural alignments.

Population Based Search

population based search methods are a class of general purpose stochastic optimisation techniques which maintain a population of solutions at each iteration (instead of just one, as seen in the search methods described above). Of the commonly used population based heuristics used for the structural alignment problem, the Genetic algorithm (Fraser, 1957; Mitchell, 1996) is arguably most popular. This search technique is inspired by the Darwinian theory of natural selection, and borrows terms such *crossover*, *mutation* and *fitness* from it (Darwin, 1859).

Broadly, starting from a population of candidate solutions, this search method involves selecting better solutions judged by a fitness (objective) function Q . The selected solutions undergo a *crossover* step to combine the best parts of the better solutions in the population. This is followed by a *mutation* step which could involve similar perturbation operations to those used for simulated annealing. After the selection, crossover and mutation steps, the existing population of solutions is updated into a new population and the process is repeated until a set of conditions are met.

The success of genetic algorithms depends on many factors, including the ability of the perturbation operations to sample from the local solution space and generate better solutions. This requires the definition of perturbation operations that are flexible and computationally efficient. The structural alignment programs KENOBI (Szustakowski and Weng, 2000), K2 (Szustakowski and Weng, 2003) and GANGSTA (Kolbeck et al., 2006) employ a genetic algorithm using their own alignment quality measures as the fitness function.

6.2.3 Search by Alternating Superposition and Alignment

This approach to find structural alignments is inspired by the Expectation Maximisation (EM) technique (Dempster et al., 1977). EM is well established in statistical learning where it is often used to optimise the parameters of a statistical model relying on hidden variables (Dempster et al., 1977).

Let D be some data and Θ be some (vector of) parameter(s) being estimated based on a statistical model with hidden variables, $\vec{\mathcal{H}}$. There are many classes of problems that require an estimated Θ and $\vec{\mathcal{H}}$ that optimise the given statistical model (often as a maximum-likelihood estimate or maximum a posteriori estimate; see Section 3.3). This is near impossible with both Θ and $\vec{\mathcal{H}}$ being unknown. To overcome this, the EM approach uses the following iterative strategy. Starting with some initial guess (or arbitrary choice) of parameters Θ_0 , and holding these parameters fixed, the method estimates the hidden variables $\vec{\mathcal{H}}_1$. This is called the *Expectation* step (or E-step). Then, holding the estimated hidden variable $\vec{\mathcal{H}}_1$ fixed, the method re-estimates the parameters Θ_1 . This step is called the *Maximisation* step (or M-step). These Expectation and Maximisation steps are iterated as follows:

$$\Theta_0 \xrightarrow{\text{E-step}} \vec{\mathcal{H}}_1 \xrightarrow{\text{M-step}} \Theta_1 \xrightarrow{\text{E-step}} \vec{\mathcal{H}}_2 \xrightarrow{\text{M-step}} \Theta_2 \dots \Theta_i \xrightarrow{\text{E-step}} \vec{\mathcal{H}}_{i+1} \xrightarrow{\text{M-step}} \dots$$

until either $\Theta_i \equiv \Theta_{i+1}$, or the maximum limit of iterations has been reached.

For the structural alignment problem, the C_α coordinates of the structural pair $\langle S, T \rangle$ are analogous to the data, D , in the general EM approach, while the parameters of superposition (rotation matrix, translation vector, and RMSD, associated with the optimal alignment \mathcal{A})

are analogous to Θ , and the hidden variables associated with that alignment \mathcal{A} are analogous to $\vec{\mathcal{H}}$. Therefore, applying the EM approach to this problem involves starting with an initial seed alignment from which the superposition parameters (Θ_0) are computed. Based on the all-versus-all C_α - C_α distances after superposition, a new alignment is generated (giving the updated hidden variables $\vec{\mathcal{H}}_1$ associated with this alignment). From this stage, the process of generating superposition parameters (Θ_i) and, through that superposition, new alignment parameters ($\vec{\mathcal{H}}_{i+1}$) is iterated until convergence.

Clearly, this approach of alternating between superposition and alignment is sensitive to the initial seed alignment. To overcome this issue either a reliable starting alignment is needed or the method has to be run with multiple distinct seeds to produce more reliable results (Eidhammer et al., 2004). Examples of protein structure alignment programs using this technique include SAL (Kihara and Skolnick, 2003), FRAGALIGN (Akutsu, 1996) and YASCA (May, 1996).

6.2.4 Dynamic Programming Based Methods for Structural Alignment

Dynamic Programming (Bellman, 1952) is an important class of algorithmic problem solving techniques which are extremely useful for addressing combinatorial optimisation problems that allow for linear (or sequential) ordering constraints. In order to be applicable, this problem solving technique requires the following two key properties to hold in the given optimisation problem.

Bellman principle of optimality: The principle relies on the ability of the given problem to be partitioned into smaller subproblems. If the optima of the global problem can be constructed from the optima of the smaller subproblems through a strictly additive function, then the problem exhibits the principle of optimality. That is, the problem possess an optimal substructure such that solving the subproblems optimally and summing their objective functions will result in the optimal evaluation of the global problem.

Overlapping Subproblems: If the global problem can be divided into subproblems, and their subproblems overlap between various branches of the subdivisions, then the global problem is set to have an overlapping substructure. This implies that each problem that overlaps can be optimally solved exactly once, and its optimal solution stored (*memoised*) in a history (or *lookup*) data structure. When a problem possesses both overlapping subproblems, and an optimal substructure, the optimal solution can be solved efficiently starting from a (lowest-order) subproblems that are trivially solvable, and building up to the higher-order subproblems through the memoisation technique.

Intuitively, the dynamic programming method of solving an optimisation problem involves breaking a large, complex problem into smaller and simpler component problems and storing their solutions in such a way that they need only be computed once (Bellman, 1952, 1957; Cormen et al., 2009). If the problem meets these required conditions, dynamic programming is guaranteed to produce an optimal solution.

Dynamic programming has been extensively applied to the field of sequence alignment of biomolecules, especially for protein sequences (Gusfield, 1997). An excellent overview of the key contributions that led to the application of dynamic programming to sequence comparison problems is given by Kruskal (1983). Needleman and Wunsch (1970) were amongst the first to apply dynamic programming to compare protein sequences and their technique, summarised below, is still very common for sequence and structure comparison.

Let \aleph represent the alphabet of amino acids, and $S = (s_1, s_2, s_3, \dots)$ and $T = (t_1, t_2, t_3, \dots)$ denote the ordered set of amino acid symbols (or sequences) along the chain of the two proteins. An objective function for a *sequence* alignment is defined using:

a scoring matrix $M : \aleph \times \aleph \rightarrow \mathbb{R}$. This scores all match states in the (sequence) alignment, and is defined as a matrix of substitution scores associated with every pair of amino acid symbols in the alphabet.

a general *subadditive*[‡] gap penalty function $\Gamma : \mathbb{Z} \rightarrow \mathbb{R}$. This function takes an integer argument corresponding to the length, l , of a run of insert or delete states in the alignment, and returns a real number signifying the penalty for those gap states.

Using this objective function, Needleman and Wunsch (1970) proposed a general-purpose dynamic programming recurrence relation of the form:

$$\begin{aligned}
 DP(0, 0) &:= 0 \\
 DP(i, 0) &:= \Gamma(i) && \forall 1 \leq i \leq |S| \\
 DP(0, j) &:= \Gamma(j) && \forall 1 \leq j \leq |T| \\
 DP(i, j) &:= \max \begin{cases} DP(i-1, j-1) + M(s_i, t_j) \\ \max_{1 \leq l \leq i} \{DP(i-l, j) + \Gamma(l)\} \\ \max_{1 \leq l \leq j} \{DP(i, j-l) + \Gamma(l)\} \end{cases} && \forall 1 \leq i \leq |S|, \forall 1 \leq j \leq |T|
 \end{aligned} \tag{6.1}$$

where DP is a history (memoisation) matrix of size $|S|+1 \times |T|+1$ such that each cell, $DP(i, j)$, in the history matrix stores the optimal score of the subproblem associated with an alignment of the prefixes of two sequences, $(s_1, s_2, s_3, \dots, s_i)$ with $(t_1, t_2, t_3, \dots, t_j)$.

The above dynamic programming recurrence is defined using a general subadditive gap penalty function. As it is defined, the time complexity for solving this recurrence is $O(|S||T|^2 + |S|^2|T|)$, that is, it requires a cubic computational effort, $O(N^3)$, where $|S| = |T| = N$. However, when the general subadditive gap penalty function, Γ , takes a special form, the dynamic programming recurrence can be further simplified. For instance when Γ is a linear function of the gap length of the form $\Gamma(l) = \delta l$, where δ is a constant, Needleman and Wunsch (1970) simplified the above recurrence into a form that takes only quadratic, $O(N^2)$, computational effort. Similarly, Gotoh (1982) proposed yet another $O(N^3)$ time dynamic programming recurrence using an *affine-linear* gap penalty function of the form $\Gamma(l) = \delta_1 l + \delta_2$, where δ_1, δ_2 are constants, although the recurrence involved three (rather than one) history matrices (Eidhammer et al., 2004).

There have been several attempts to extend the dynamic programming apparatus of sequence alignment to structural alignments, to be used as fast heuristics rather than as complete optimal techniques. One approach has been to transform the 3D structural alignment problem into a one-dimensional characterisation, where the symbols in such characterisations encode the structural and physico-chemical information of successive amino acid residues within a specified window (Levine et al., 1984; Karpen et al., 1989; Friedberg et al., 2007). Although very fast to run, such one-dimensional characterisations of three dimensional structures do not yield accurate results (Kolodny et al., 2005; Hasegawa and Holm, 2009). They are particularly vulnerable when the structures being compared contain elements of secondary structures that are deleted with respect to each other, or even when there is a major difference in the lengths of secondary structural elements (Konagurthu et al., 2006).

[‡]A function $f : \mathbb{R} \rightarrow \mathbb{R}$ is subadditive if $\forall p, q \in \mathbb{R}, f(p+q) \leq f(p) + f(q)$

Another approach is to define, using a reasonable superposition of the two structures,[§] an all-versus-all Euclidean distance matrix between each possible C_α - C_α pair. This distance matrix is converted into a scoring matrix (analogous to the substitution matrix in the sequence alignment case) and, using an *ad hoc* gap penalty function, the structures are aligned (iteratively as in the aforementioned EM approach) using minor modifications to the standard form of the dynamic programming recurrence. While all of these attempts are heuristics, they have shown promise for generating seed alignments that can be later refined (Konagurthu et al., 2006). Other programs that use dynamic programming in some form or other include DALI (Holm and Sander, 1993) which performs dynamic programming over a distance matrix, STRUCTAL (Levitt and Gerstein, 1998), SAL (Kihara and Skolnick, 2003) which uses standard Needleman Wunsch dynamic programming to find structural alignments from specially constructed scoring matrices, and TM-Align (Zhang and Skolnick, 2005b), which uses Needleman-Wunsch dynamic programming over secondary structures to obtain a seed alignment which is improved upon using other methods.

Double Dynamic Programming

Double dynamic programming is a dynamic programming algorithm that performs superposition and alignment simultaneously, rather than alternating as described above. This is achieved by performing dynamic programming in duplicate when constructing an alignment: once at a global level (using a high-level scoring matrix, HL), and once at the local level (using low-level scoring matrices, ${}^{ij}LL$), for every pair of residues. This is illustrated in Algorithm 2.

Let $S = (s_1, s_2, s_3, \dots, s_m)$ and $T = (t_1, t_2, t_3, \dots, t_n)$ denote the ordered set of amino acids in the pair of structures, S and T . For each pair of residues, s_i and t_j , a low-level scoring matrix, ${}^{ij}LL$, is constructed under the constraint that this pair forms part of the optimal path. Under this constraint, a superposition (see Section 2.2.3) is calculated, then the alignment objective function is used to fill-in the scoring matrix. Note that, a superposition of a single residue pair does not orient S with respect to T . One way of establishing an orientation is to define a unique reference coordinate system using the residues around s_i and t_j . For example, the triplets $\{s_{i-1}, s_i, s_{i+1}\}$ and $\{t_{j-1}, t_j, t_{j+1}\}$ can be superimposed in order to orient S with T . Then, depending on the objective function being optimised, (s_i, t_j) can be forced into the optimal path within ${}^{ij}LL$ by either setting the score in ${}^{ij}LL_{ij}$ to be so high that the optimal path must pass through it, or by computing the optimal path first through ${}^{ij}LL_{kl}$, ($1 \leq k \leq i, 1 \leq l \leq j$), then again through ${}^{ij}LL_{kl}$, ($i \leq k \leq m, j \leq l \leq n$) and joining the paths.

The high level scoring matrix, HL , is constructed by accumulating (normalised) scores from all of the low-level scoring matrices. This reinforces higher scoring alignment paths through HL and eliminates lower scoring paths. Finally, a Needleman-Wunsch dynamic program is run, using HL as the scoring matrix, to find the optimal path. The time complexity of this double dynamic programming approach is quadratic in the size of the structures being aligned. Assuming that $|S| \leq |T|$ and $N = |T|$, each low-level matrix can be computed in $O(N^2)$ time. Since there are N^2 low-level matrices that go into building the high-level matrix, construction of HL requires $O(N^4)$ time. Finally, dynamic programming on HL requires $O(N^2)$ time. Therefore, double dynamic programming requires $O(N^4)$ time to find an alignment.

Double dynamic programming was used by the landmark SSAP (Taylor and Orengo, 1989; Orengo and Taylor, 1996) protein structure alignment program, which is also used as the automatic basis for classification by the CATH domain classification database (see Section 2.1.4;

[§]This can be computed from an initial seed alignment, as seen in the local search methods.

Algorithm 2: Double Dynamic Programming

```

input : A pair of protein structures:  $S$  and  $T$ 
output: An alignment

1  $HL[] \leftarrow [0]_{|S| \times |T|}$ 
2 for  $i \leftarrow 1$  to  $\|S\|$  do
3   for  $j \leftarrow 1$  to  $\|T\|$  do
4      $^{ij}LL[] \leftarrow \text{create\_low\_level}(i, j)$ 
5      $HL[] \leftarrow HL[] + \text{normalise}(^{ij}LL)$ 
6 return  $\text{run\_dynamic\_program}(HL[])$  /* Return the optimal path through  $HL$  */

```

Orengo et al. (1997)). The SAP (Taylor, 2000) algorithm applies double dynamic programming several times in an iterative manner.

6.3 Searching for Structural Alignments Using I -value

This section presents a structural alignment program, **MMLigner**, that uses the MML based I -value measure developed in Chapter 4 and improved in Chapter 5. **MMLigner** implements an efficient search heuristic that initially finds a reliable set of seed alignments for a given pair of protein structures using methods similar to those described in Section 6.2.1 earlier. These seed alignments are then refined over an EM-like approach (see Section 6.2.3). In this section, the alignment search heuristic will be referred to as the ‘**MMLigner** algorithm’.

The full details of the design principles behind the **MMLigner** algorithm are described in Section 6.3.2. However, before settling upon the **MMLigner** algorithm as the method of choice to search for structural alignments using I -value measure, many different approaches were trialled during the long gestation of this thesis. The most promising of these is a heuristic based on dynamic programming, and discussed below in Section 6.3.1. Although this search method lacks the computational efficiency and consistency that was finally achieved using the **MMLigner** algorithm, the dynamic programming heuristic is nevertheless noteworthy for potentially facilitating a visual description of the optimal alignment solution space (see below and Chapter 8). Thus its inclusion in this thesis.

6.3.1 Dynamic Programming Using the I -value Measure

Recall that (from Equation 5.1), for a pair of protein structures $\langle S, T \rangle$, defined as ordered sets of C_α coordinates $S = \{\vec{s}_1, \vec{s}_2, \vec{s}_3, \dots\}$ and $T = \{\vec{t}_1, \vec{t}_2, \vec{t}_3, \dots\}$, and any alignment \mathcal{A} between them, the I -value measure of alignment quality is defined as:

$$\underbrace{I(\mathcal{A}, \langle S, T \rangle)}_{I\text{-value}} = \underbrace{I(\mathcal{A})}_{\text{First part}} + \underbrace{I_{\text{null}}(S) + I(T|S, \mathcal{A})}_{\text{Second part}} \quad \text{bits.}$$

As seen earlier (see Section 4.4), any (order-preserving) alignment \mathcal{A} between the two structures can be represented as a three-state string over the **match**, **insert**, and **delete** alignment states. Thus, any *prefix*[¶] of that alignment string describes the alignment relationship between the C_α coordinates: $S_{1\dots i} = (\vec{s}_1, \vec{s}_2, \dots, \vec{s}_i)$ with $T_{1\dots j} = (\vec{t}_1, \vec{t}_2, \dots, \vec{t}_j)$. Let $\mathcal{A}_{i,j}$ represent the alignment prefix that suggests an alignment between the C_α coordinates $\langle S_{1\dots i}, T_{1\dots j} \rangle$ and let

[¶]Given a string $s[1 \dots n]$, the prefix of the string is given by $s[1 \dots i], \forall 1 \leq i \leq n$.

$I\text{-value}(i, j)$ give the I -value for describing the structural coordinates $\langle S_{1\dots i}, T_{1\dots j} \rangle$ using the alignment $\mathcal{A}_{i,j}$:

$$I\text{-value}(i, j) = I(\mathcal{A}_{i,j}, \langle S_{1\dots i}, T_{1\dots j} \rangle) = I(\mathcal{A}_{i,j}) + I_{\text{null}}(S_{1\dots i}) + I(T_{1\dots j} | S_{1\dots i}, \mathcal{A}_{i,j}) \quad \text{bits.}$$

Using these notations, a dynamic programming heuristic is defined using I -value as the objective function. What is referred to as optimal is any value that is optimal according to the heuristic, however, since the method is a heuristic, the value stored cannot be guaranteed to be optimal. This approach is similar in spirit to the algorithm used for pairwise protein sequence alignment in Equation 6.1, due to Gotoh (1982), using *affine-linear* gap penalty function mentioned previously (see Section 6.2.4).

The method proceeds as follows. Let DP_{match} , DP_{delete} , and DP_{insert} represent three dynamic programming history matrices, each of size $|S| + 1 \times |T| + 1$. Let any cell $DP_{\text{match}}(i, j)$, store the *optimal* (or shortest) $I\text{-value}(i, j)$ for the coordinates $\langle S_{1\dots i}, T_{1\dots j} \rangle$, such that the optimal alignment $\mathcal{A}_{i,j}^{\text{match}}$ ends in a match state. (In other words, the pair of C_α coordinates \vec{s}_i and \vec{t}_j are aligned in that optimal alignment $\mathcal{A}_{i,j}^{\text{match}}$). Similarly, the cells $DP_{\text{delete}}(i, j)$ and $DP_{\text{insert}}(i, j)$ store the *optimal* (or shortest) $I\text{-value}(i, j)$ such that their optimal alignments $\mathcal{A}_{i,j}^{\text{delete}}$ and $\mathcal{A}_{i,j}^{\text{insert}}$ end in delete or insert states respectively, where either \vec{s}_i or \vec{t}_j are unaligned in the optimal alignments corresponding to those cells.

Based on this, a heuristic dynamic programming recurrence is defined, where the cells within the three history matrices, $\{DP_{\text{match}}, DP_{\text{delete}}, DP_{\text{insert}}\}$, storing I -values for the subproblems (that is, the alignment of $\langle S_{1\dots i}, T_{1\dots j} \rangle$), can be updated from the neighboring subproblems (involving $(i-1, j-1)$ -cells, $(i-1, j)$ -cells and $(i, j-1)$ -cells). For instance, the value stored at the cell $DP_{\text{match}}(i, j)$, where the alignment $\mathcal{A}_{i,j}^{\text{match}}$ ends in a match state, can be derived from the alignment solution associated with one of the following cells: $DP_{\text{match}}(i-1, j-1)$, $DP_{\text{delete}}(i-1, j-1)$, or $DP_{\text{insert}}(i-1, j-1)$.

Let $\Delta I_{m \rightarrow m}$ represent the update to the I -value term required for expanding the alignment $\mathcal{A}_{i-1, j-1}^{\text{match}}$ (associated with $DP_{\text{match}}(i-1, j-1)$) by a further match state. This new alignment $\mathcal{A}_{i,j}^{\text{match}}$ is associated with the cell $DP_{\text{match}}(i, j)$. Therefore, the term $\Delta I_{m \rightarrow m}$ involves the following updates:

Update to $I(S)$: The coordinate $\vec{s}_i \in S$ is stated using the null model (see Section 4.5) taking $I_{\text{null}}(\vec{s}_i)$ bits.

Update to $I(T|S, \mathcal{A})$: The coordinate $\vec{t}_j \in T$ is stated using the coordinate compression model encoding (see Section 4.6) using the information from \vec{s}_i taking $I_{\text{aligned}}(\vec{t}_j | \vec{s}_i)$ bits.

Update to $I(\mathcal{A})$: $I(\mathcal{A}_{i,j}^{\text{match}})$ (see Section 4.4) can be updated from $I(\mathcal{A}_{i-1, j-1}^{\text{match}})$ by:

- removing the code length term accounting for its length, $I_{\text{integer}}(|\mathcal{A}_{i-1, j-1}^{\text{match}}|)$
- adding the code length term accounting for the length increase, $I_{\text{integer}}(|\mathcal{A}_{i-1, j-1}^{\text{match}}| + 1)$
- adding the code length to state the $m \rightarrow m$ alignment state transition, $-\log_2(\text{Pr}(\text{mm}))$

The general set of recurrences driving this dynamic programming heuristic and the update terms are listed below:

$$DP_{\text{match}}(i, j) = \min \begin{cases} DP_{\text{match}}(i-1, j-1) + \Delta I_{m \rightarrow m} \\ DP_{\text{insert}}(i-1, j-1) + \Delta I_{i \rightarrow m} \\ DP_{\text{delete}}(i-1, j-1) + \Delta I_{d \rightarrow m} \end{cases}$$

where

$$\begin{aligned}
\Delta I_{m \rightarrow m} &= I_{\text{null}}(\vec{s}_i) + I_{\text{aligned}}(\vec{t}_j | \vec{s}_i) - \log_2(\text{Pr}(\text{mm})) - I_{\text{integer}}(|\mathcal{A}_{i-1,j-1}^{\text{match}}|) + I_{\text{integer}}(|\mathcal{A}_{i-1,j-1}^{\text{match}}| + 1) \\
\Delta I_{i \rightarrow m} &= I_{\text{null}}(\vec{s}_i) + I_{\text{aligned}}(\vec{t}_j | \vec{s}_i) - \log_2(\text{Pr}(\text{im})) - I_{\text{integer}}(|\mathcal{A}_{i-1,j-1}^{\text{insert}}|) + I_{\text{integer}}(|\mathcal{A}_{i-1,j-1}^{\text{insert}}| + 1) \\
\Delta I_{d \rightarrow m} &= I_{\text{null}}(\vec{s}_i) + I_{\text{aligned}}(\vec{t}_j | \vec{s}_i) - \log_2(\text{Pr}(\text{dm})) - I_{\text{integer}}(|\mathcal{A}_{i-1,j-1}^{\text{delete}}|) + I_{\text{integer}}(|\mathcal{A}_{i-1,j-1}^{\text{delete}}| + 1)
\end{aligned}$$

$$DP_{\text{delete}}(i, j) = \min \begin{cases} DP_{\text{match}}(i-1, j) + \Delta I_{m \rightarrow d} \\ DP_{\text{insert}}(i-1, j) + \Delta I_{i \rightarrow d} \\ DP_{\text{delete}}(i-1, j) + \Delta I_{d \rightarrow d} \end{cases}$$

where

$$\begin{aligned}
\Delta I_{m \rightarrow d} &= I_{\text{null}}(\vec{t}_i) - \log_2(\text{Pr}(\text{md})) - I_{\text{integer}}(|\mathcal{A}_{i,j-1}^{\text{match}}|) + I_{\text{integer}}(|\mathcal{A}_{i,j-1}^{\text{match}}| + 1) \\
\Delta I_{i \rightarrow d} &= I_{\text{null}}(\vec{t}_i) - \log_2(\text{Pr}(\text{id})) - I_{\text{integer}}(|\mathcal{A}_{i,j-1}^{\text{insert}}|) + I_{\text{integer}}(|\mathcal{A}_{i,j-1}^{\text{insert}}| + 1) \\
\Delta I_{d \rightarrow d} &= I_{\text{null}}(\vec{t}_i) - \log_2(\text{Pr}(\text{dd})) - I_{\text{integer}}(|\mathcal{A}_{i,j-1}^{\text{delete}}|) + I_{\text{integer}}(|\mathcal{A}_{i,j-1}^{\text{delete}}| + 1)
\end{aligned}$$

$$DP_{\text{insert}}(i, j) = \min \begin{cases} DP_{\text{match}}(i, j-1) + \Delta I_{m \rightarrow i} \\ DP_{\text{insert}}(i, j-1) + \Delta I_{i \rightarrow i} \\ DP_{\text{delete}}(i, j-1) + \Delta I_{d \rightarrow i} \end{cases}$$

where

$$\begin{aligned}
\Delta I_{m \rightarrow i} &= I_{\text{null}}(\vec{s}_i) - \log_2(\text{Pr}(\text{mi})) - I_{\text{integer}}(|\mathcal{A}_{i-1,j}^{\text{match}}|) + I_{\text{integer}}(|\mathcal{A}_{i-1,j}^{\text{match}}| + 1) \\
\Delta I_{i \rightarrow i} &= I_{\text{null}}(\vec{s}_i) - \log_2(\text{Pr}(\text{ii})) - I_{\text{integer}}(|\mathcal{A}_{i-1,j}^{\text{insert}}|) + I_{\text{integer}}(|\mathcal{A}_{i-1,j}^{\text{insert}}| + 1) \\
\Delta I_{d \rightarrow i} &= I_{\text{null}}(\vec{s}_i) - \log_2(\text{Pr}(\text{di})) - I_{\text{integer}}(|\mathcal{A}_{i-1,j}^{\text{delete}}|) + I_{\text{integer}}(|\mathcal{A}_{i-1,j}^{\text{delete}}| + 1)
\end{aligned}$$

The above recurrence involves latent variables in terms of nine transition probabilities, $\text{Pr}(\text{mm}), \text{Pr}(\text{im}), \text{Pr}(\text{dm}), \dots$ and so on, that dictate the encoding of each state transition in the alignment encoding. These terms can be inferred using an EM approach (see Section 6.2.3). Starting with some initial values for these probabilities, the E-step involves running the above dynamic program to find an alignment. The M-step involves updating the transition probabilities derived from the resultant alignment and iterating the dynamic program until convergence.

The above set of recurrences do not guarantee finding the optimal alignment using the I -value measure. This is because the I -value measure does not obey the Bellman *principle of optimality* (Bellman, 1952): the global structural alignment problem cannot be broken down into *strictly additive* subproblems (as being shown in the aforementioned recurrence relationships). This is mainly due to the fact that the estimation of $I(T|S, \mathcal{A})$ terms is not additive as it relies on superposition and the superposition depends on the correspondences assigned by the alignment (see Section 2.2.3). For various (i, j) -cells in the dynamic program, the superposition changes depending on the alignments and, hence, the additivity breaks down.

Generating Landscapes of Structural Alignment Quality

Despite the above dynamic program being a heuristic, it often results in very good structural alignments, particularly when the dynamic program is not distracted by early greedy choices during the recurrence. This is useful, as it can be used to produce very interesting and insightful visualisations of the structural alignment landscape. A landscape \mathcal{L} , of alignment quality between structures S and T can be defined by a matrix of order $(|S| + 1 \times |T| + 1)$. Each cell

$\mathcal{L}(i, j)$ of this matrix stores the I -value of the best alignment *passing through* (or involving) the pair of coordinates \vec{s}_i and \vec{t}_j .

To compute $\mathcal{L}(i, j)$, the method begins by computing the dynamic programming history matrices $\{DP_{\text{match}}, DP_{\text{delete}}, DP_{\text{insert}}\}$, as shown in the aforementioned recurrence. After this, the same dynamic programming is carried out, but in the backwards direction by reversing the order of coordinates in S and T . The resulting history matrices are denoted by $\{\text{bwd}DP_{\text{match}}, \text{bwd}DP_{\text{delete}}, \text{bwd}DP_{\text{insert}}\}$. Using these history matrices computed in both the forward and reversed directions, $\mathcal{L}(i, j)$ can be computed as:

$$\begin{aligned} \mathcal{L}(i, j) = & \min(DP_{\text{match}}(i, j), DP_{\text{insert}}(i, j), DP_{\text{delete}}(i, j)) \\ & + \min(\text{bwd}DP_{\text{match}}(i, j), \text{bwd}DP_{\text{insert}}(i, j), \text{bwd}DP_{\text{delete}}(i, j)) \end{aligned}$$

Plotting the matrix \mathcal{L} leads to a visual appreciation of the optimal alignment landscape using the I -value measure. An example of this landscape for the pair of protein structures, Succinyl-CoA synthetase from *Sus scrofa* (wwPDB 1EUD-A; a two domain protein containing 305 residues) and Glutamate mutase from *Clostridium cochlearium* (wwPDB 1CCW-A; a single domain protein containing 137 residues) is given in Figure 6.1. Notice that there are two “valleys” through which an alignment path is traced (in red and yellow). These paths correspond to the two alternative domain alignments identified by the algorithm. The first is an alignment of the N-terminal domain from wwPDB 1EUD-A with wwPDB 1CCW-A. The second is the alignment of the C-terminal domain of wwPDB 1EUD-A with wwPDB 1CCW-A. A more detailed analysis of the alignments between this pair of structures is given later, in Section 6.3.2 and Figure 6.2. More examples of these landscapes can be found in Section 8.1.4.

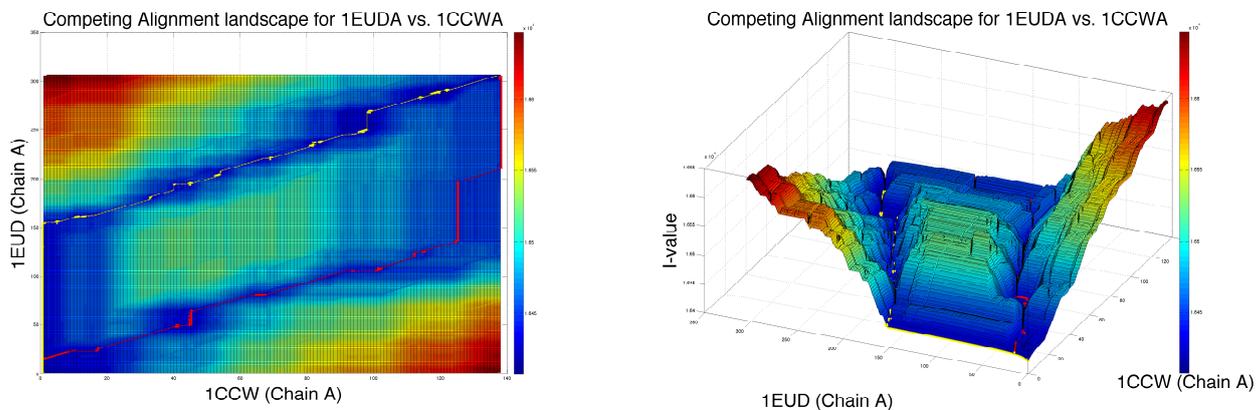


Figure 6.1: A visualisation of the landscape of competing alignment quality for the pair of structures, 1EUD-A and 1CCW-A. The two best alternative alignments are highlighted in red and yellow.

Summary

This section has described a dynamic programming based heuristic to search for structural alignments using the I -value measure of alignment quality. The evaluations of the performance of this heuristic shows it does not consistently generate high quality alignments.[‡] The inconsistencies arise, in many cases, due to premature greedy choices that are globally suboptimal and

[‡]Results are not included because a completely new search method is proposed (see Section 6.3.2), which supersedes this heuristic.

distract the dynamic programming algorithm from finding a reasonable structural alignment. However, when the algorithm does find a good structural alignment, it can be used to visualise of the landscape of optimal alignments, from which alternative alignments can be interactively explored. Since it is critical to have a consistent and reliable alignment search method, a different search strategy was explored, and the resulting method is explained in Section 6.3.2 below.

6.3.2 The MMLigner Algorithm

This section describes the final version of the heuristic search procedure used by MMLigner to identify statistically significant and biologically meaningful pairwise alignments using I -value (see Chapter 5) as the objective function. An important advantage of the search method detailed in this section is that it is able to explore alternative structural alignments for the same pair of structures and report those that pass the null hypothesis test as described in Section 4.3.

The MMLigner search method is carried out in two phases. The first phase, called seeding, combines fast fragment assembly using sufficient statistics for superposition with dynamic programming (see Section 6.2.1 and Section 7.2). The aim of this phase is to quickly and deterministically generate alternative structural alignments by efficiently clustering consistent assemblies of well-superposable fragment pairs for the two given structures S and T . The second phase, called refinement, optimises these seed alignments using the I -value criterion with an expectation maximisation optimisation algorithm (see Section 6.2). This search procedure is illustrated by example using the pair of structures wwPDB 1EUD-A and wwPDB 1CCW-A already used in the previous section. This pair of structures was earmarked as a difficult case for structural alignment by Sippl and Wiederstein (2008), as they possess two plausible alternative structural alignments and, hence, they are useful in demonstrating the effectiveness of the MMLigner search method for identifying and reporting alternative alignments.

Phase 1: Generating Alignment Seeds

This phase is intended to extract a set of good seed alignments. Briefly, phase one consists of the following steps:

1. Find all equi-sized contiguous fragments from each structure that fit within an RMSD threshold. This identifies locally fitting fragment pairs.
2. Filter the fragments found in step 1 by eliminating all fragment pairs that do not jointly fit below an RMSD threshold. This eliminates locally fitting fragment pairs that are not globally useful.
3. Cluster the remaining fragment pairs into groups that fit consistently.
4. For each cluster, build a scoring matrix based on the superimposed fragment distances within the cluster. Then use dynamic programming to build a seed alignment from the cluster.

The remainder of this section details each of the above steps.

Phase 1, Step 1: Identification of a library of maximal fragment pairs (MFPs).

Given a pair of structures S and T , the algorithm identifies all well-superposable, maximally-extended fragment pairs that fit within a threshold of RMSD of 2 Å. A Maximal Fragment Pair

(MFP) is the central unit of operation for phase 1 and it is defined as a contiguous well fitting pair of fragments with at least 6 residues. An MFP can be considered as a contiguous diagonal series of cells in a matrix with $|S|$ rows and $|T|$ columns. An MFP of length l , starting at S_i in S and starting at T_j in T , is represented by the diagonal starting at $\bar{D}_{i,j}$ and occupying the diagonal for l cells. The results of this step, defined in Algorithm 3, is a library of MFPs for the given pair of structures being aligned. This library can be visualised in matrix form as in Figure 6.2(a).

Algorithm 3: Identify library of MFPs

input : A pair of protein structures: a reference structure, S , and a target structure, T
output: All maximally sized fragment pairs that fit within a threshold of RMSD

```

1  $mfp\_library[] \leftarrow []$ 
2 for  $i \leftarrow 1$  to  $\|S\| - 6$  do
3   for  $j \leftarrow 1$  to  $\|T\| - 6$  do
4      $ss \leftarrow \text{superimpose}(S_{i,i+6}, T_{j,j+6})$ 
5     for  $l \leftarrow 6$  to  $\min(\|S\| - i, \|T\| - j)$  do
6       if not  $\text{superseded}(ss, mfp\_library[])$  and  $\text{rmsd}(ss) < 2.0$  then
7          $mfp\_library[] \leftarrow \text{append}(ss)$ 
8       else
9         break
10       $ss \leftarrow \text{update}(S_{i+l+1}, T_{j+l+1})$ 
11 return  $mfp\_library[]$ 

```

Algorithm 3 systematically searches over S and T , growing every possible MFP from the minimum size of 6 until the RMSD threshold is broken. This is performed rapidly by using the addition `update` function for superposition sufficient statistics defined Section 7.2.4. An MFP is not added to the library if it is *superseded* by an MFP already there. A superseding MFP is one that contains all the same correspondences and more. If a valid MFP is superseded by an MFP already in the library, it is not added. Note that *superseding* is distinct from *overlapping*. Overlapping occurs when MFPs share some, but not all, of their residue-residue correspondences.

This procedure results in a large search space as shown for example, in Figure 6.2(a), where the blue highlighted areas represent MFPs found using this method for the pair of structures wwPDB 1EUD-A and wwPDB 1CCW-A. The next step is used to reduce the library of MFPs by eliminating globally poor fitting areas.

Phase 1, Step 2: Filtering the library of MFPs. The MFP library from step 2 is filtered in this step to contain only MFPs that can be jointly superposed with at least two other MFPs in the set. To begin, every pair of non-overlapping MFPs that can be jointly superposed under the threshold of RMSD of 3Å is stored. Any MFP that does not superpose within this threshold is discarded since it shows no evidence of being spatially consistent with any other MFPs in the set. This eliminates MFPs that are locally but not globally meaningful. For each pair of MFPs that jointly superpose within the RMSD threshold, the algorithm extends the superposition further using yet another (non-overlapping) MFP. Thus, a superposition using a triplet of MFPs is formed. Any triplet of MFPs that does not jointly superimpose below the threshold of RMSD of 4Å is discarded. This further prunes the original library of MFPs (see Figure 6.2(b)).

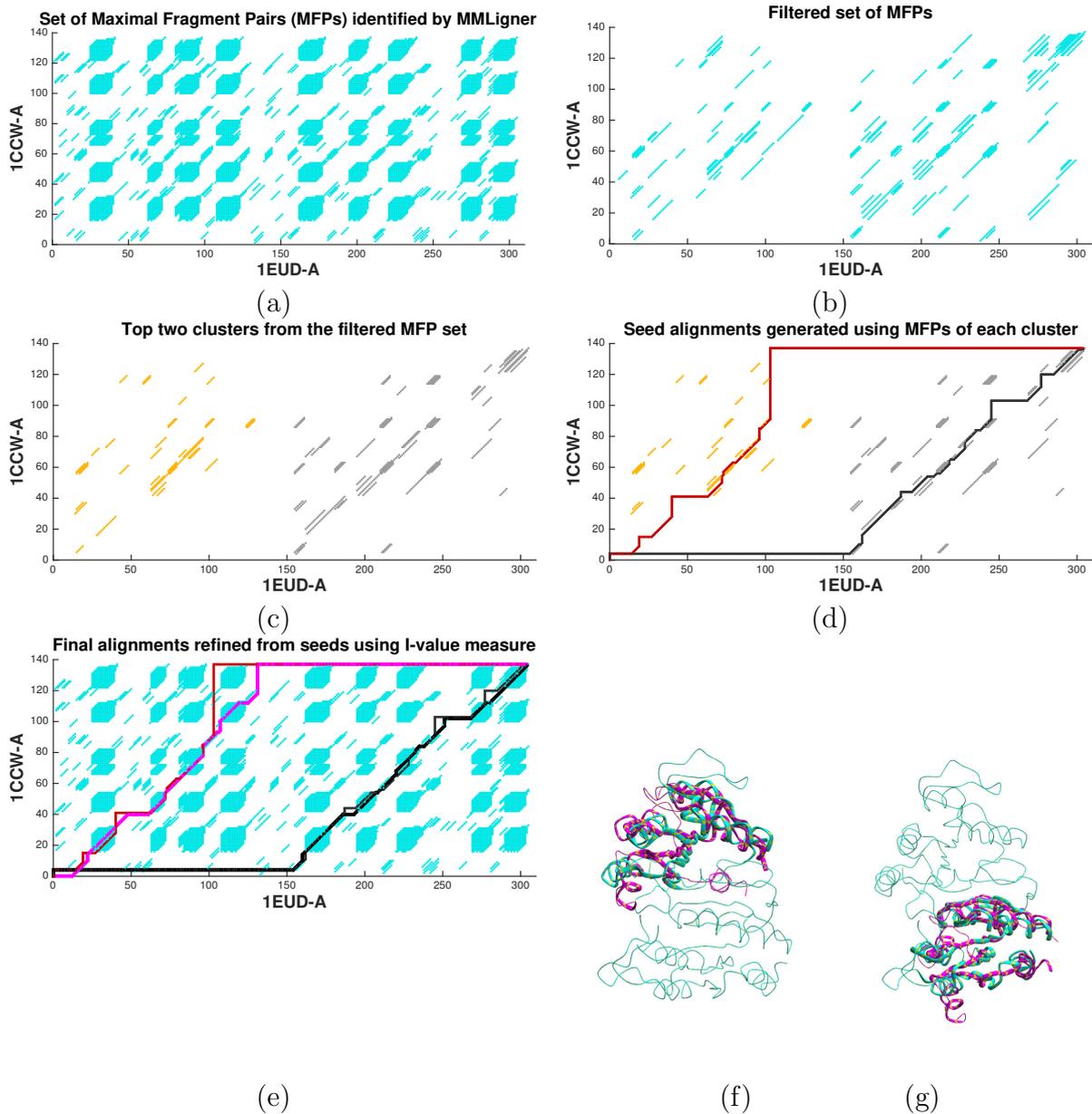


Figure 6.2: Illustration of different steps used in *MMLigner*'s heuristic search using the example of wwPDB 1EUD-A and wwPDB 1CCW-A. See Section 6.4.1 for more details about this structural pair. (a) A 2D matrix showing the full library of maximal (well-superposable) fragment pairs (MFPs) identified by *MMLigner*. Each pair is represented as a contiguous series of diagonal blue dots. (b) Filtered set of MFPs were each listed MFP jointly superposes with two other (non-overlapping) MFPs in the set. (c) Separation of the filtered set of MFPs into two clusters of consistently fitting MFPs. (d) Determination of a tentative seed alignments for each cluster shown as a source (bottom left) to sink (top right) path. (e) Refinement of the tentative seed alignments using *I*-value measure. (f-g) Superposition of the two structures using the final two alternate alignments found post refinement. Note, the first and second rows in Table 6.2 corresponds to the alignment paths shown in black and magenta in (e).

If carried out naïvely, this filtering procedure can be computationally expensive. However, by using sufficient statistics of superposition as described in Section 7.2, the filtering procedure can exhaustively and very efficiently compute joint superpositions over all MFP pairs, and their further extensions to MFP triples, by benefiting from the constant time update feature from Section 7.2.4. To do this, when the original library of MFPs is computed, the sufficient statistics of the superposition in each MFP is also stored. The sufficient statistics and, hence, the RMSD of joint superposition of pairs of MFPs can be computed as a constant time update using the sufficient statistics of individual MFPs. Similarly, extensions of pairs to triples can also be updated in constant time. As a result, the identification of MFPs and the pruning can be carried out exhaustively in the matter of a few seconds. Figure 6.2(b) shows the result of filtering the set of MFPs from Figure 7.5(a).

Phase 1, Step 3: Clustering the filtered set of MFPs. The aim of this step is to partition the filtered set of MFPs into groups (or clusters) of consistently fitting MFPs. A seed alignment can then be generated within each cluster.

The clustering heuristic proceeds iteratively as in Algorithm 4. First, the filtered set of MFPs is sorted in decreasing order of length (in terms of number of residue pairs in each MFP) and the longest MFP in the filtered set is assigned as the initial singleton cluster. The iterative process of clustering then starts by traversing the remaining sorted list of filtered MFPs, starting from the longest. For each MFP in the list, the algorithm checks whether it can be added to any of the already created clusters. If it can, it is added into that cluster. Otherwise, a new cluster is created with this MFP as its singleton member. An MFP can be added to a cluster if the MFP jointly superposes with at least 40% of the MFPs already assigned to that cluster. This is repeated until all MFPs in the filtered list has been assigned to a cluster.

At the end of this procedure, there may be clusters that contain only a small number of short MFPs, and/or a set of MFPs that largely *overlap* in a small section. Such clusters should be discarded. To achieve that, if the combined, non-overlapping length of the MFPs in the cluster is less than three** times the minimum MFP length, that is if the cluster defines less than 18 consistent correspondences, the cluster is eliminated. Note that this is not applied simply to clusters containing only a small number of MFPs since, for closely related structures, an MFP can contain a large proportion of one or both structures and is thus meaningful in defining an alignment.

The remaining clusters are used in the next steps to identify seed alignments. Figure 6.2(c) shows the top two clusters of MFPs (based on the combined length of MFPs in them) for the filtered set from Figure 6.2(b).

Phase 1, Step 4: Finding a seed alignment using the clustered MFPs. In this final step for phase 1, the set of MFPs in each cluster are converted into a weight matrix, which is used to construct a rough seed alignment using a dynamic programming technique. Dynamic programming is introduced in Section 6.2.4 above.

A scoring matrix, \bar{D} , with $|S|$ rows and $|T|$ columns is computed for each cluster. Every MFP over $3 \times \text{MIN_MFP_LEN}$ is assigned a simple (RMSD, N_e) type score based on the scoring function from Ilyin et al. (2004) as in Equation 6.2.

$$V = 0.25 \cdot N_e \cdot e^{-0.39 \cdot \text{RMSD}^2} \quad (6.2)$$

**Three due to the filtering step accepting a triplet of MFPs

Algorithm 4: Clustering filtered MFPs

```

input : A filtered set of MFPs
output: A set of MFP clusters

1 MFP[] ← sort_by_length(MFP[])
2 clusters[] ← []
3 for mfp in MFP[] do
4   for cluster in clusters[] do
5     if mfp superimpose with ≥ 40% of cluster ≤ THRESHOLD then
6       cluster ← add_to_cluster(cluster)
7       continue
8   if not member_of_cluster(mfp) or empty(clusters[]) then
9     clusters[] ← append_new_cluster(mfp)
10 return clusters[]

```

Each cell occupied by these MFPs in \bar{D} are assigned the value $\frac{V}{3 \times \text{MIN_MFP_LEN}}$. This process of filling cells in the matrix is repeated for non-overlapping MFP pairs (and triples) by multiplying the number of ways a MIN_MFP_LEN MFP can be tiled over the total extent of matches described by the MFPs in the pair (or triple). This process reinforces closely matching residue pairs and eliminates poorly matching residue pairs.

Once the scoring matrix \bar{D} has been constructed using the above process, an alignment is produced using the constant gap-penalty dynamic programming technique described in Section 6.2.4. The gap-penalty is set to zero to ensure that the generated alignment is constructed purely based on the geometric scores from the clustered and filtered MFPs. The recurrence relation is as follows:

$$\begin{aligned}
 M_{0,0} &= 0 \\
 M_{i,0} &= M_{0,j} = 0 \\
 M_{i,j} &= \max \begin{cases} M_{i-1,j-1} + \bar{D}_{i,j} \\ M_{i-1,j} \\ M_{i,j-1} \end{cases} \quad (6.3)
 \end{aligned}$$

This relation fills a memoisation matrix, M , from which the optimal path is extracted. This procedure is repeated for every cluster identified step 3 to generate geometrically plausible seed alignments as input to phase 2, which is described below. Figure 6.2(d) shows the alignment paths found for the top two clusters identified in Figure 6.2(c).

Phase 2: Refinement of the Seed Alignment Using I -value Criterion

Phase 1 of the procedure produces geometrically plausible seed alignments that are optimised in phase 2 through a series of small perturbations. Using each seed alignment as the starting point, a series of perturbations are carried out to identify the final alignment that the **MMLigner** search method reports. The fitness of each (perturbed) alignment is evaluated using the I -value measure (see Chapter 5). In particular, the fitness is maximised as the amount of compression

the perturbed alignment achieves over the null model message length:

$$\begin{aligned} \text{compression}(\mathcal{A}, \langle S, T \rangle) &= I_{\text{null}}(\langle S, T \rangle) - I\text{-value} \\ &= I_{\text{null}}(\langle S, T \rangle) - I(\mathcal{A}, \langle S, T \rangle) \end{aligned}$$

The procedure is described by Algorithm 5 below. Broadly, the approach is similar to the EM method introduced in Section 6.2.3. It treats alignments as a list of matched blocks in the same way as the corresponding block alignment encoding method represented alignments in Section 4.4. The algorithm iterates over each match block in the alignment and applies a series of primitive perturbation operations to it and to any leading and trailing gaps. This is repeated, operating upon the current *best* alignment at each iteration, until the method converges or reaches the maximum of 25 iterations. Note that the perturbation functions: *ExtendMatchBlock*, *ShrinkMatchBlock*, *SwapMatch*, *SlideMatchBlock*, and *RealignClosest* are defined below.

Algorithm 5: Heuristic Expectation-Maximisation search

input : An initial (seed) alignment, \mathcal{A} , and a pair of structures, $\langle S, T \rangle$
output: An I -value (locally) optimal alignment, \mathcal{A}^*

```

1  $n\text{Iters} \leftarrow 0$ 
2  $\mathcal{A}^* \leftarrow \mathcal{A}$ 
3 while  $n\text{Iters} \leq 25$  do
4    $best\_ival \leftarrow \text{compression}(\mathcal{A}, \langle S, T \rangle)$ 
5    $ptrbd[] \leftarrow []$ 
6    $ptrbd\_ival[] \leftarrow []$ 
7   for  $i \leftarrow 1$  to  $\|matched\_blocks\|$  do
8     for  $n \leftarrow 1$  to 6 do
9       for  $perturbType$  in  $\{ExtendMatchBlock, ShrinkMatchBlock, SwapMatch,$ 
10         $SlideMatchBlock\}$  do
11          $\mathcal{A}_{tmp} \leftarrow perturbType(i, n, left)$ 
12          $ptrbd\_ival[] \leftarrow \text{append}(\text{compression}(\mathcal{A}_{tmp}, \langle S, T \rangle))$ 
13          $ptrbd[] \leftarrow \text{append}(\mathcal{A}_{tmp})$ 
14          $\mathcal{A}_{tmp} \leftarrow perturbType(i, n, right)$ 
15          $ptrbd\_ival[] \leftarrow \text{append}(\text{compression}(\mathcal{A}_{tmp}, \langle S, T \rangle))$ 
16          $ptrbd[] \leftarrow \text{append}(\mathcal{A}_{tmp})$ 
17          $\mathcal{A}_{tmp} \leftarrow \text{RealignClosest}(i)$ 
18          $ptrbd\_ival[] \leftarrow \text{append}(\text{compression}(\mathcal{A}_{tmp}, \langle S, T \rangle))$ 
19          $ptrbd[] \leftarrow \text{append}(\mathcal{A}_{tmp})$ 
20        $\mathcal{A}_{tmp} \leftarrow \text{best}(ptrbd\_ival[], ptrbd[])$ 
21       if  $ptrbd\_ival[\mathcal{A}_{tmp}] < best\_ival$  then
22          $\mathcal{A}^* \leftarrow \mathcal{A}_{tmp}$ 
23          $best\_ival \leftarrow ptrbd\_ival[\mathcal{A}_{tmp}]$ 
24      $n\text{Iters} \leftarrow n\text{Iters} + 1$ 
25 return  $\mathcal{A}^*$ 

```

There are five primitive perturbation operations defined for this algorithm. Four of these primitive operations take 3 arguments: 1) the index^{††} of the matched block to operate upon,

^{††}In these examples the first index is 1.

2) the size of the operation to perform, and 3) the direction to perform the operation. The meaning of the direction argument is dependent on the operation being performed, however it is generally related to the use of gaps either *left* or *right* of the matched block. The fifth primitive perturbation operation, *RealignClosest*, takes only one argument, the matched block index. All of these operations are described below alongside minimal working examples.

ExtendMatchBlock($i, n, direction$): This primitive tries to extend the i^{th} matched block by n residues either on the left or right of the matched block depending on the *direction* argument. That is, it tries to create new matches out of the deletes and inserts flanking the matched block in the specified direction. The number of new matches is limited by $\min(|\text{inserts}|, |\text{deletes}|)$ on either flank of the i^{th} matched block. For example, there are 3 deletions and 3 insertions on the *left* of the matched block marked by ‘*’ characters over residues 4,5,6,7 in *T* below. Therefore, the size of the matched block can be increased by a maximum of three matches to the left. In the example below, it is extended at the left by adding 2 matches for residues 2,3:

<i>Before</i>	<i>Operation</i>	<i>After</i>
***** 123 4567 S XXX---XXXX T ---XXXXXXXX 1234567	<i>ExtendMatchBlock</i> (2, 2, left)	***** 1 234567 X-XXXXXX -XXXXXXX 1234567

ShrinkMatchBlock($i, n, direction$): This primitive tries to shrink the i^{th} match block by n residues either on the left or right of the matched block depending on the *direction* argument. That is, it tries to remove matches by converting matched pairs of residues into insertions and deletions flanking the current matched block in the specified direction. The number of removed matches is limited by the size of the matched block being operated upon. In the example below, the matched block index 1 marked by ‘*’ characters over residues 4,5,6,7 in *T* below is shrunk from the right by converting the three matches between residues 5,6,7 into inserts and deletes, thus shrinking the match block to a size of 1 between residues 4 in *S* and *T*:

<i>Before</i>	<i>Operation</i>	<i>After</i>
***** 123 4567 S XXX---XXXX T ---XXXXXXXX 1234567	<i>ShrinkMatchBlock</i> (2, 3, right)	* 123 4 567 XXX---X---XXX ---XXXXXXXX--- 1234567

SwapMatch($i, n, direction$): This primitive tries to swap n aligned residues across a monotonous (only inserts, or only deletes) gapped region from an adjacent matched block to the i^{th} matched block (or vice versa) in the specified direction. The number of gaps remains constant but the size of the block increases by the size of the swap. The size of the swap is limited by the number of matched residues in the adjacent matched block. In the example below, the second matched block, marked by ‘*’ characters, containing residues 7,8,9 from *T* is increased in size by swapping the match of residue 3 in the adjacent matched block to the right. Thus increasing the size of the second matched block by 1, now containing residues 6,7,8,9 from *T*:

<i>Before</i>	<i>Operation</i>	<i>After</i>
*** 123 456 S XXX--XXX T XXXXXXXX 123456789	<i>SwapMatch</i> (2, 1, right)	**** 12 3456 XX--XXXX XXXXXXXXXX 123456789

SlideMatchBlock($i, n, direction$): This primitive tries to change the residue-residue correspondences of the i^{th} matched block by moving (or *sliding*) n residues in S left or right relative to T according to the *direction*. Note that the number of correspondences in a block remains constant or reduces (no new correspondences are created), and that a shift left in S is equivalent to a shift right in T and vice-versa. The size of the shift is limited by the size of the matched block being operated upon, plus the number of gaps available in the direction of the shift. Two examples are shown below. The first involves shifting a matched block, the residues 1,2,3 in S to the left by two positions reducing the match block for residues 1,2,3 in T to just 1. The second example shows the same shift but this time to the right. In this case the shift is not over gaps but over unaligned residues, resulting in no decrease in match block size for but a change in which residues are aligned:

<i>Before</i>	<i>Operation</i>	<i>After</i>
*** 123 456 S XXX--XXX T XXXXX-XX 12345 67	<i>SlideMatchBlock</i> (1, 2, left)	* 123 456 XXX----XXX --XXXXX-XX 12345 67
*** 123 456 S XXX--XXX T XXXXX-XX 12345 67	<i>SlideMatchBlock</i> (1, 2, right)	*** 123456 --XXXXXX XXXXX-XX 12345 67

RealignClosest(i): This primitive is a special case in that it operates over an area and, thus, no size or direction is given as an argument. This primitive is a limited version of the dynamic programming algorithm used in step 4 of the phase 1 seeding procedure. It realigns the entire region around the specified match block (including its flanking gaps), based on C_{α} - C_{α} distances after superposition of the structures based using the current alignment. For example, assume that in the alignment below all residue pairs in the first matched block marked by ‘*’ fit well. Assume also, that the 2 inserted residues fit well with the 2 deleted residues. Performing the *RealignClosest*(0) operation on this matched block results in an alignment of all 5 residues in S : 1, 2, 3, 4, 5. Note however, that the original aligned block need not be preserved, and could be replaced by a more well fitting set of matches for the given alignment.

<i>Before</i>	<i>Operation</i>	<i>After</i>
*** 123 456 S XXX--XXX T XXXXX--X 12345 6	<i>RealignClosest</i> (1)	***** 123456 XXXXXX XXXXXX 123456

The procedure begins by building a scoring matrix, \bar{D} , from the residues in S and T , restricted to those contained in the matched blocks and surrounding adjacent inserted and

deleted residues. Firstly, the entire structures are superimposed according to the unmodified alignment, then, in the range of residues above, an inter-residue distance matrix is constructed containing the distances between all pairs of residues from S and T in the restricted range. Each value in \bar{D} is then computed as the inter-residue distance subtracted from the largest inter-residue distance in the distance matrix.

Once \bar{D} has been constructed, a new alignment for the restricted range in S and T is produced using a constant gap-penalty dynamic program. The recurrence relation is given in Equation 6.3 with the gap penalty set to zero. Extracting the optimal path produces a re-alignment of the restricted range around the indexed matched block. This re-aligned area replaces the restricted range in the alignment.

All seed alignments produced from phase 1 are refined using the above primitives over an EM algorithm, as defined in Algorithm 5. At each iteration, the algorithm attempts to exhaustively perturb each match block using the above perturbation primitives, and greedily chooses the best perturbation by assessing the level of compression using the I -value. This continues until either the alignment converges (it always will but might take a long time) or reaches the maximum number of iterations. This behaviour is intended to ensure the algorithm explores the local space thoroughly around the starting point provided by the first, *seeding* phase of **MMLigner**.

6.4 Results and Discussion

In this section, the performance of **MMLigner** is evaluated and compared with the following popular structural alignment programs: **CE** (Shindyalov and Bourne, 1998), **DALI** (Holm and Sander, 1993), **LGA** (Zemla, 2003), **FatCat** (Ye and Godzik, 2003), and **TM-Align** (Zhang and Skolnick, 2005b). The evaluation takes three forms. First, the utility of **MMLigner** in reporting alternate (competing) structural alignments is explored using case studies. Secondly, three structural pairs from Sippl and Wiederstein (2008) on difficult structural alignments are used to qualitatively compare the above alignment methods with **MMLigner**. And third, a quantitative evaluation is undertaken on how well all of these alignment methods perform in discriminating closely-related, moderately-diverged, highly-diverged and unrelated proteins as defined by the SCOP (Murzin et al., 1995; Lo Conte et al., 2000) hierarchical domain classification database. Here, the performance of these alignment programs is measured by the set of alignment quality measures discussed in Chapter 4 and using the same set of 2500 randomly selected domain pairs from Section 4.8 (also listed in Appendix A).

6.4.1 Identification of Alternate Structural Alignments

First case study: wwPDB 1EUD-A versus wwPDB 1CCW-A. Consider again the alpha chains (chain A) from the pair of proteins: Succinyl-CoA synthetase from *Sus scrofa* (wwPDB 1EUD-A) and Glutamate mutase from *Clostridium cochlearium* (wwPDB 1CCW-A). As indicated before, wwPDB 1EUD-A contains 306 amino acid residues, while wwPDB 1CCW-A contains 137 residues. The SCOP database dissects wwPDB 1EUD-A into two domains, **d1euda1** (region A:1-130) and **d1euda2** (region A:131-306), and classifies them under different folds. The N-terminal (CoA-binding) domain, **d1euda1**, falls under NAD(P)-binding Rossmann fold, while the C-terminal domain, **d1euda2**, is classified under Flavodoxin-like fold. In contrast, SCOP classifies wwPDB 1CCW-A as a small subunit domain, **d1ccwa_**, also under the Flavodoxin-like fold.

Interestingly, the two domains of the Succinyl-CoA synthetase, `d1euda1` and `d1euda2`, share self-similarity and, hence, it is possible to align wwPDB `1CCW-A` to either the N-terminal region (A:1-130) or C-terminal region (A:131-306) of wwPDB `1EUD` resulting in two alternate alignments. This structural pair is among a set of hard structural alignments identified by Sippl and Wiederstein (2008).

`MMLigner` successfully identifies both these alignments (see Figure 6.2(e-g)). Of the other programs, only `DALI` identifies both these alignments as being significant, while `CE`, `FatCat`, `TM-Align`, and `LGA` yield just one significant alignment of wwPDB `1CCW-A` involving the C-terminal domain of wwPDB `1EUD-A`. See Section 6.4.2 for further discussion about the quality of structural alignments produced by these alignment methods.

Second case study: wwPDB `2SAS-A` versus wwPDB `1JFJ-A`. Consider the alpha chains (chain A) from the pair of calcium-binding proteins from *Branchiostoma lanceolatum* (wwPDB `2SAS-A`: 134 residues) and *Entamoeba histolytica* (wwPDB `1JFJ-A`: 185 residues). The corresponding domains in SCOP, `d2sasa_` and `d1jffa_`, are classified within the Calmodulin-like family (suggesting close evolutionary relationship). Both the domains contain a duplication of two EF-hand (helix-loop-helix) motifs. That is, there are four EF-hand motifs in each domain, with two EF-hands forming the N-terminus subunit and two EF-hands forming the C-terminus subunit. However, the duplicated subunits have markedly different geometries in each of the proteins: while the subunits are flexed compactly in `d2sasa_`, they are found to be relaxed in `d1jffa_`.

When aligning these two structures, there are four possible alignments matching each possible pair of EF-hands between the two proteins. Let these four possible alignments be denoted by NN, NC, CN and CC, corresponding to the matching up of the pair of FH-hands at the N-terminus or C-terminus part of the two proteins. `MMLigner` identifies all four alignments and flags them as significant (since the *I*-value message lengths of all these alignments are shorter than the null model message length). These alignments are shown as four distinct source-to-sink colored paths on a 2D plot in Figure 6.3(a) and the corresponding least-squares superpositions of wwPDB `2SAS-A` on wwPDB `1JFJ-A` are shown (with colors consistent with the 2D plot) in Figure 6.3(b-c). See figure caption for more details.

Passing the same structural pair through other structural aligners for a comparison, it can be seen that, excluding `DALI`, all other aligners return just one structural alignment of the four possible alignments. Table 6.1 contains the coverage and RMSD values returned by the various aligners. Notice that although `DALI` reports all four alignments, only two of them are reported as competitive. `MMLigner` consistently produces good quality alignments on all the four cases.

To summarise, `MMLigner` is able to successfully identify a range of significant alternative alignments, which are at least consistent with equivalent alignments produced by other alignment programs. These alternatives reveal structural relationships that are invisible when only a single alignment is produced.

6.4.2 Performance of `MMLigner` on Hard Structural Alignment Cases

Three pairwise structural comparisons that are classified as ‘hard structural alignment problems’ by Sippl and Wiederstein (2008) are reported here. This dataset is used to assess the quality of alignments reported by the alignment programs, mentioned earlier, on difficult to align structure pairs, where multiple reasonable alternative alignments exist. The quality of the alignments generated by these programs is shown in Table 6.2. This table gives the traditional coverage and RMSD measures. In addition, information-theoretic estimates of alignment

Table 6.1: Assessment of structural alignments produced by **MMLigner**, **DALI**, **FatCat**, **CE**, **TM-Align**, and **LGA** on the pair of calcium-binding domains, **d2sasa_** and **d1jfja_**. Alignment **NN** denotes the alignment of N-terminal subunit of **d2sasa_** (residues 1 to 99) with the corresponding N-terminal subunit of **d1jfja_** (residues 1 to 70). Alignment **CC** denotes the alignment of C-terminal subunit of **d2sasa_** (residues 100 to 184) with the corresponding C-terminal subunit of **d1jfja_** (residues 71 to 134). (Similar definitions for the Alignments **CN** and **CC**.) The Coverage column gives the number of residue-residue correspondences reported by the respective alignment programs. The RMSD column gives the root-mean-squared-deviation after best superposition in Åunits. The $I(\mathcal{A})$ column gives the measure of alignment (descriptive) complexity in bits. The compression column gives the difference between the null model message length and the I -value for each alignment. The symbol ‘-’ is used when no alignment is reported by the respective alignment program.

	Coverage	RMSD	$I(\mathcal{A})$ (bits)	Compression (bits)		Coverage	RMSD	$I(\mathcal{A})$ (bits)	Compression (bits)		Coverage	RMSD	$I(\mathcal{A})$ (bits)	Compression (bits)
MMLigner						FatCat					TM-Align			
alignment NN	53	2.75	93.2	88.9	-	-	-	-	-	-	-	-	-	-
alignment NC	65	3.42	81.2	95.6	-	-	-	-	-	-	-	-	-	-
alignment CN	62	2.90	61.5	127.1	-	-	-	-	-	67	3.13	73.7	114.2	-
alignment CC	65	3.37	71.1	116.8	39	5.50	77.9	-58.3	-	-	-	-	-	-
DALI						CE					LGA			
alignment NN	65	13.35	46.9	-306.3	-	-	-	-	-	-	-	-	-	-
alignment NC	66	3.73	92.5	59.8	-	-	-	-	-	-	-	-	-	-
alignment CN	80	12.53	65.2	-134.3	-	-	-	-	-	72	5.57	94.2	28.3	-
alignment CC	64	3.51	76.3	88.6	73	4.55	72.2	88.1	-	-	-	-	-	-

(descriptive) complexity and of the compression they achieve based on the I -value measure are provided. Note that a complex alignment, \mathcal{A} in this context corresponds to one that takes a long message, $I(\mathcal{A})$, to encode. For the estimation of $I(\mathcal{A})$ given in Section 5.2.1, the longer the **match/delete/insert** blocks the simpler the alignment is measured to be.

The first two rows of Table 6.2 deal with the two alternate alignments for the pair **wwPDB 1EUD-A** versus **wwPDB 1CCW-A** discussed in Section 6.4.1. All the alignment programs considered report one of the two alignments, while only **MMLigner** and **DALI** report both. In the first case, **LGA**’s alignment was the worst overall in terms of coverage and RMSD, and was flagged statistically insignificant by the I -value statistical significance test, since its I -value message length is 5.2 bits longer the null model message length. **FatCat** and **TM-Align** produce alignments with similar coverage but **TM-Align**’s RMSD is better. However, the alignment complexity ($I(\mathcal{A})$) is significantly worse than that of **FatCat**’s alignment. On manual inspection, it can be seen that **TM-Align** aligns singleton or pairs of residues in regions of dissimilarity purely on the grounds that they happen to drift close by in space. Manual inspection of the other cases indicates that **TM-Align** often results in alignments with higher coverage than the other alignment programs and with acceptable RMSD values but contains several spurious correspondences and are, thus, more complex. **DALI** yields an alignment with similar coverage and RMSD statistics as **TM-Align**, but with slightly less complex alignments. Both **MMLigner** and

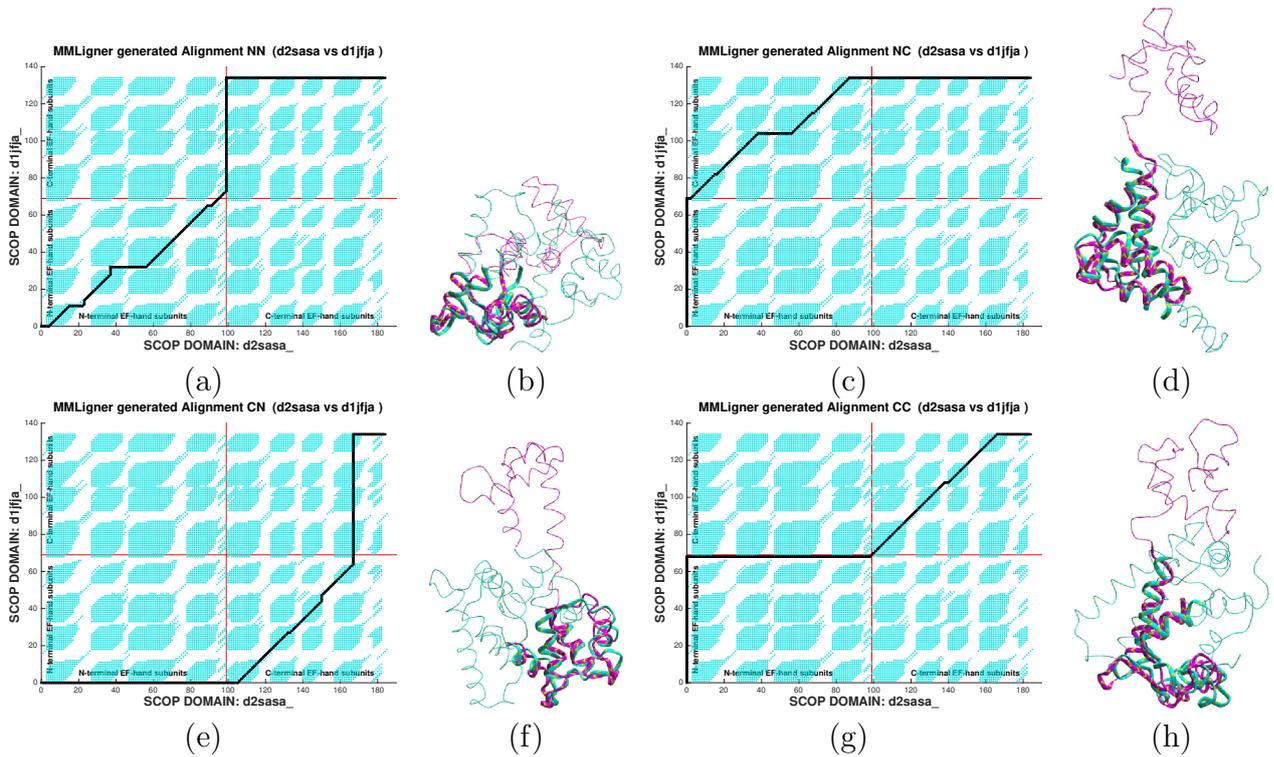


Figure 6.3: Four alignments identified by **MMLigner** when aligning the two calcium-binding protein domains `d2sasa_` and `d1jfja_`. These alignments are denoted here as NN, NC, CN and CC. (a,c,e,g) Identified alignments (NN, NC, CN, CC respectively) shown as paths from source (bottom-left) to sink (top-right) on a 2D dot plot of MFPs found during the search in the backdrop. (b,d,f,g) Superpositions of `d1jfja_` on `d2sas_` using the identified NN, NC, CN and CC alignments, respectively.

CE produce similar alignments with the difference accounted for by the different levels of conservatism of their respective objective functions. In the second case, only **MMLigner** and **DALI** produce an alternate alignment for the above structural pair. **MMLigner** produces the more conservative and agreeable alignment of the two: **DALI** appears to be overly eager to extend existing match blocks beyond where the structures show local similarity.

The pair `d1euda1` versus `d1euda2` (third row of Table 6.2) examines the structural self-similarity of the two domains of the wwPDB 1EUD-A. **MMLigner**, **CE**, and **TM-Align** produce agreeable alignments with minor differences that can be put down to differences in the objective functions being optimised. Qualitatively, these alignments are similar. Other aligners, **DALI**, **FatCat**, and **LGA** produce poor-quality alignments.

To summarise, **MMLigner** consistently produces reliable structural alignments for each of these hard structural alignment cases from Sippl and Wiederstein (2008). **CE** in most cases performs equally well, however it is unable to identify alternate alignments when they exist. **DALI** and **TM-Align** produce good alignments in a good number of cases, but are sprinkled with spurious correspondences upon closer inspection. **LGA** consistently produces the worst alignments of the programs used in this comparison. Furthermore, **MMLigner** can identify alternative alignments for the same structural pair when they exist, and it does so consistently compared to other programs. Most programs are designed to report only one alignment.

Table 6.2: Performance of several popular structural alignment programs on the hard structural alignment cases reported by Sippl and Wiederstein (2008). The meaning of table headers is the same as in Table 6.1. The alignment of wwPDB 1EUD-A versus wwPDB 1CCW-A should generate two alternate structural alignments (see Section 6.4.1). The symbol ‘-’ is used when an alignment program did not report an alignment.

	Coverage	RMSD	$I(\mathcal{A})$	Compression
MMLigner				
1EUD-A vs. 1CCW-A	118	3.15	120.0	146.7
1EUD-A vs. 1CCW-A	98	3.17	111.4	108.2
d1euda1 vs. d1euda2	94	3.36	111.2	109.6
DALI				
1EUD-A vs. 1CCW-A	125	3.77	139.9	91.4
1EUD-A vs. 1CCW-A	111	3.87	116.1	69.1
d1euda1 vs. d1euda2	122	11.4	93.3	-137.7
FatCat				
1EUD-A vs. 1CCW-A	125	4.44	114.1	61.0
1EUD-A vs. 1CCW-A	-	-	-	-
d1euda1 vs. d1euda2	105	5.79	102.3	-14.3
CE				
	119	3.26	123.6	89.6
	-	-	-	-
	97	3.39	112.8	97.3
TM-Align				
	125	3.45	146.8	68.6
	-	-	-	-
	99	3.43	131.8	69.0
LGA				
	122	6.19	187.4	-5.2
	-	-	-	-
	112	11.11	108.4	-88.4

6.4.3 Large Scale Comparison on SCOP Domains with Varying Structural Distance

Finally, a quantitative analysis on a large data set of protein structures is undertaken to assess whether these programs can discriminate between protein structural pairs across the entire spectrum of structural and evolutionary distance. A statistically large data set was selected from SCOP. This is the same dataset used for experiments in Section 4.8 and in Section 5.3. Domains were selected from SCOP using the procedure outlined in Appendix A, which also gives a complete listing of the domains used.

As used previously, performance of the alignments generated by various programs is evaluated using the descriptive statistics for observed coverage, RMSD alignment descriptive complexity ($I(\mathcal{A})$) and compression gained (both in bits) using I -value. Analysing this large amount of quantitative data across these four criterion requires efficient graphical representations to glean clear insights.

Figure 6.4 presents a matrix of box-and-whisker plots, where the four rows correspond to each of the four criteria: coverage, RMSD, alignment complexity and compression; and the six columns correspond to the six alignment programs: **MMLigner**, **CE**, **DALI**, **TM-Align**, **FatCat**, and **LGA**, respectively. Each plot in the matrix summarises, as notched box-and-whisker plots, the distribution of data for a specific alignment program (in columns) measuring a specific alignment quality criterion (in rows) across each of the 5 levels of the SCOP heirarchy: Family, Superfamily, Fold, Class and Decoy.

Examining Figure 6.4, it can be seen that the *median* (red line within the box plots) statistics for alignment coverage criterion corresponding to **DALI**, **TM-Align**, and **LGA** are similar and dominate those corresponding to other programs, including **MMLigner**. The median alignment coverage for **CE** and **FatCat** remains the most conservative over all the programs. However, analysing coverage in isolation is not useful without putting it in the context of how well the structures fit. Comparing the median alignment RMSD statistics after the best rigid-body superposition of respective structures based on the generated alignments, it is clear that **MMLigner** yields the most favourable profile across the SCOP groups, followed by **CE** and **FatCat**. **DALI** and **LGA** yield alignments with poorest fits, especially for Family, Superfamily and Fold groups.

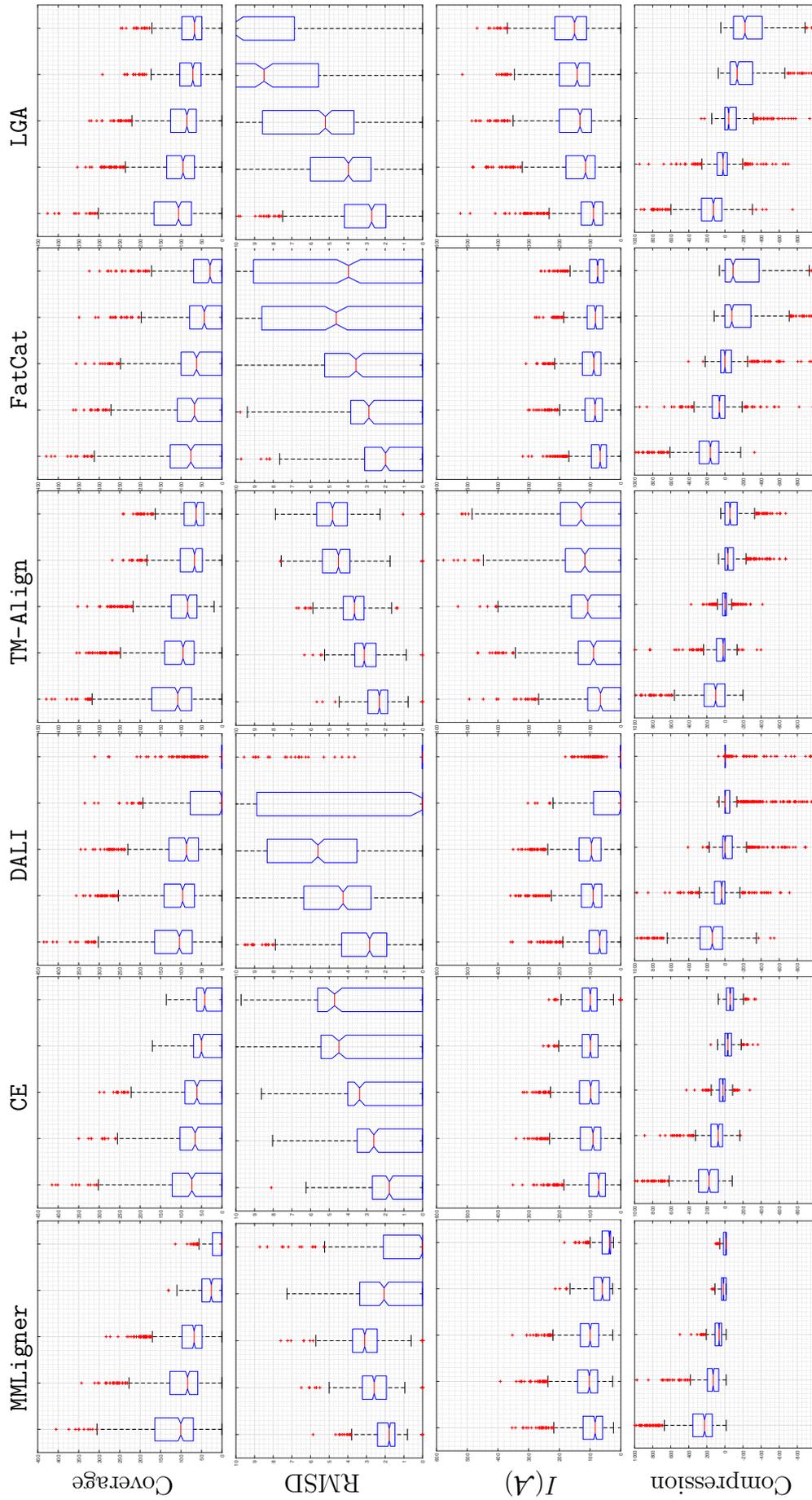


Figure 6.4: Notched box-and-whisker plots displaying the distribution of various criteria across $500 \times 5 = 2500$ alignments. Columns indicate the alignment program used to align the entire data set: MMLigner, CE, DALI, TM-Align, FatCat, and LGA. Rows indicate various alignment criteria: Coverage, RMSD, $I(\mathcal{A})$ and Compression. Note that the ordinate scale for each of the four criteria has been forced to be **fixed** to their respective ranges so as to make the comparisons between plots across alignment programs more accessible. Withing each column, left to right, are data for: Family, Superfamily, Fold, Class, and Decoy.

Since their coverage statistics dominate over the other alignment programs, it can be inferred that DALI and LGA have a tendency to over-align residues at the cost of a poorer fit between the structures. This behaviour has been observed on more careful manual study of many alignments and superpositions (similar to what was observed above for the hard structural alignment cases described by Sippl and Wiederstein (2008).) TM-Align's RMSD profile, although poorer than programs such as MMLigner, CE, and FatCat, remains reasonably well behaved for the Family, Superfamily and Fold groups. However, as noted in the previous section, upon manual inspection of individual cases, TM-Align tends to greedily align spurious residues when they appear spatially proximal even though that is by chance, yielding more complex alignments than what is acceptable. This behaviour can be observed by studying the plots for TM-Align alignment complexity which steadily increases from Family through to Decoy and is substantially larger than that of other alignment programs (except for LGA) from Fold through to Decoy.

Analysing the spread of the interquartile (IQR) range across the alignment programs for coverage and RMSD, MMLigner appears to produce the most balanced alignment of all. The compactness of these boxes indicates its consistency over other programs to identify meaningful structural alignments. The non-overlap of the notches (that, as indicated earlier, signifies the 95% confidence interval) of MMLigner plots suggest that the medians are statistically different. No other program produces non-overlapping notches between boxes across all groups for the coverage statistics. Notice that MMLigner coverage gradually decreases moving from Family to Superfamily to Fold. However coverage falls significantly when moving from Fold to Class and decreases further for the Decoy group. The median coverage MMLigner produces for the Class group sits at 35 correspondences, and the median for the decoy set sits at 18 correspondences. These correspondences are due to MMLigner rightly aligning up to two distinct supersecondary structures, often involving long helices, for the Class data set. For the Decoy set, these correspondences are found at the level of one supersecondary structure on average. The concavity of the RMSD median lines for MMLigner highlights what should be expected of this data set, where the RMSD grows from Family to Fold as the structural distance grows, but decreases sharply for Class and Decoy groups, as the relationship is rather a simple one involving supersecondary structural matches.

This concavity is also seen in the alignment complexity plots (third row of Figure 6.4) for MMLigner, CE, DALI, and FatCat where the descriptive complexity of alignments increases with the growing structural distance moving from Family to Fold, but decreases for Class and furthermore for Decoy groups where the relationship is, on average, a simple one. The continuously increasing alignment complexities for TM-Align and LGA is suggestive of over-alignment when the relationship is increasingly tenuous.

The final (fourth row) of Figure 6.4 shows the I -value information criterion, the total compression gained (in bits) by the alignment programs compared to the null model description of the structural coordinates. It is not surprising that MMLigner gives the best compression overall, as this is the objective of the heuristic search (see Section 6.3) that MMLigner is trying to optimise. However, it provides useful information as the horizontal line at 0 bits is the discriminating line for statistical significance in the I -value information-theoretic framework. When compression is positive (above the zero line), the alignment provides a more compact *lossless* explanation of the coordinates of the protein structural pair being aligned than their null model (coordinates stated independently) description and, hence, that alignment should be accepted as statistically significant. On the other hand, when the compression is negative (below the zero line) the lossless explanation is longer than the null model message length and, hence, the alignment should be rejected. Based on this, MMLigner's boxes (hence, their corresponding alignments) for Family, Superfamily and Fold are always above the zero line suggesting that

these alignments are significant as measured by the I -value. The first quartile line for Class group and the second quartile line for Decoy group lies on zero. Surprisingly, excepting CE, on average other alignment programs produce alignments for the SCOP Fold group which are below the zero line. Note that DALI does not return any alignment for most of the Class and Decoy sets. The program is trained to discriminate true negatives, but is less accurate when aligning proteins that are true positives in terms of evolutionary relationship.

It is useful to compare how alignments produced by MMLigner perform when using other criteria for alignment quality. The scoring functions used here are the same as those used previously in Chapters 4 and 5. Those scoring functions are: DALI z-score (Holm and Sander, 1993), TM-Score (Zhang and Skolnick, 2004), MI and SI (Kleywegt and Jones, November 1994), GDT_TS and LGA_S3 (Zemla, 2003), SAS (Subbiah et al., 1993), and GSAS (Kolodny et al., 2005).

As above, Figures 6.5 and 6.6 present a matrix of box-and-whisker plots where, in this case, the six columns correspond to each of the six structural alignment programs: MMLigner, CE, DALI, TM-Align, FatCat, and LGA. The ten rows, spread over the two figures, correspond to the ten alignment quality criteria. Each plot in the matrix summarises, as notched box-and-whisker plots, the distribution of data for a specific alignment program measuring a specific alignment quality criterion across each of the 5 levels of the SCOP heirarchy: Family, Superfamily, Fold, Class and Decoy.

The performance of the various alignment programs is summarised, from Figures 6.5 and 6.6 in Table 6.3. This table shows the best performing alignment program as ranked by the various structural alignment quality measures across the levels of the SCOP hierarchy.

Table 6.3: A summary of the (median) best performing alignment program (rows) according to the scoring functions (columns) for each level of the SCOP hierarchy. Detailed results are shown in Figures 6.5 and 6.6

	I -value compression	STRUCTAL_score	DALI z-score	TM-Score	GDT_TS	LGA_S3	MI	SI	SAS	GSAS
Family	MMLigner	TM-Align	TM-Align	TM-Align	MMLigner	MMLigner	MMLigner	CE	CE	CE
Superfamily	MMLigner	TM-Align	TM-Align	TM-Align	MMLigner	MMLigner	TM-Align	CE	CE	CE
Fold	MMLigner	TM-Align	TM-Align	TM-Align	MMLigner	MMLigner	TM-Align	CE	CE	FatCat
Class	MMLigner	FatCat	DALI	TM-Align	MMLigner	MMLigner	TM-Align	MMLigner	MMLigner	–
Decoy	MMLigner	CE	MMLigner/DALI	TM-Align	CE	CE	TM-Align	CE	TM-Align	–

All alignment programs generate alignments that show (in Figure 6.5) the expected decay with the levels of the SCOP heirarchy. It is to be expected that the alignment programs will generate the best alignment according to their native scoring function. This is the case for MMLigner and TM-Align, but not the case for DALI where TM-Align followed by MMLigner perform even better on all levels of the SCOP hierarchy (except domains in the same class). And the native scores for LGA grade MMLigner as producing the best alignments at all levels of the SCOP hierarchy except for decoy domains. Counter-intuitively, MMLigner Class level alignments are scored *better* than MMLigner Fold level alignments using both GDT_TS and LGA_S3. This is likely an artifact of these scoring functions being designed to grade the quality of structures predicted from sequence, where the alignment state string is mostly (or entirely) made up from match states. MMLigner finds very few correspondences between domains that share a Class level relationship, but they are relatively well fitting (see Figure 6.4).

To summarise, MMLigner is able to produce highly consistent structural alignments compared to all other programs being compared in this chapter. True to the MML framework that relies on achieving an accurate trade-off between hypothesis (here, alignment) complexity and its fit with the observed data (here, the lossless explanation of structural coordinates). MMLigner consistently identifies meaningful alignments, avoids pairing up spurious correspondences, and prefers simple alignments over complex ones. Furthermore, structural alignments

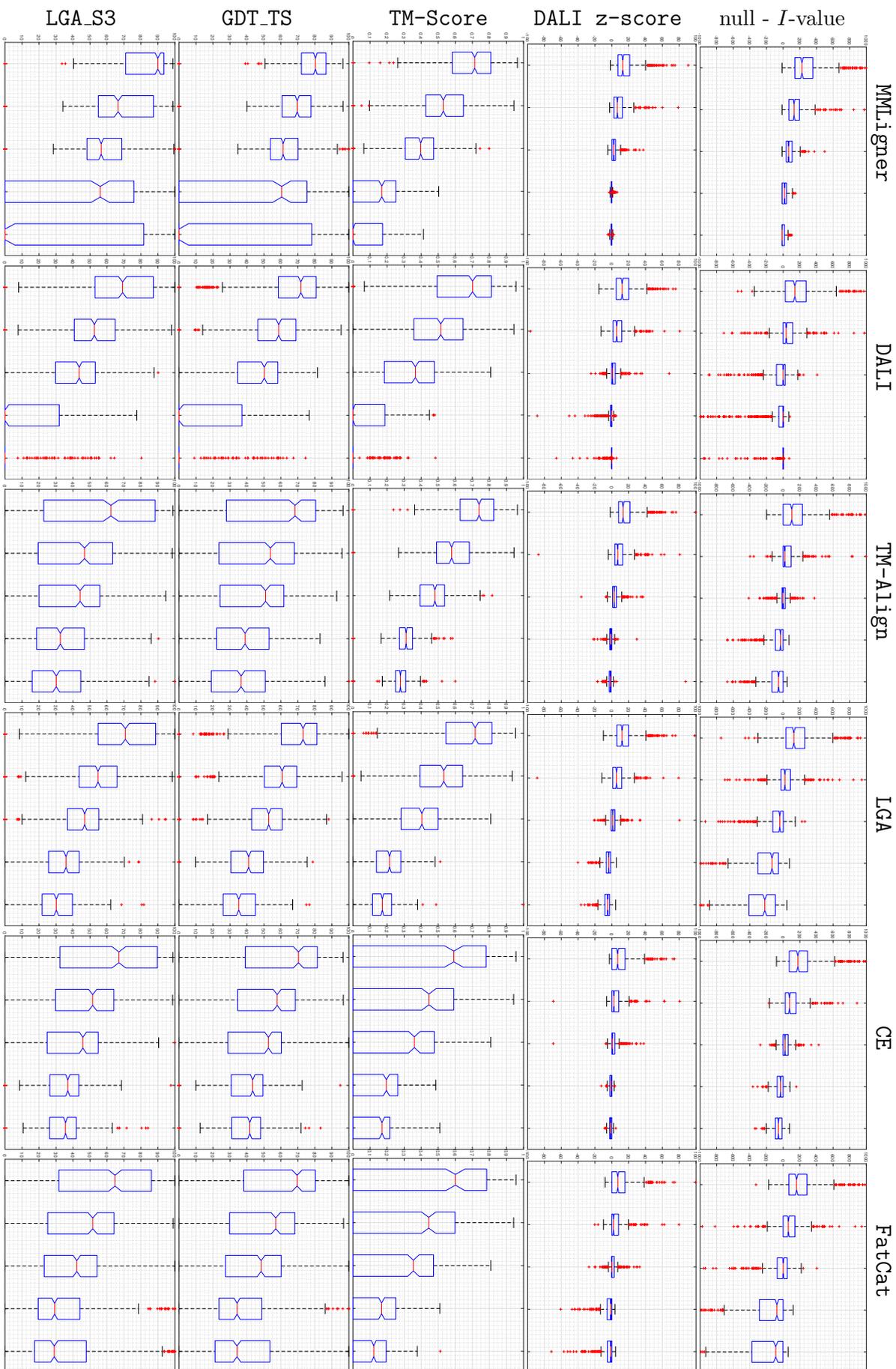


Figure 6.5: Notched box-and-whisker plots for the 2500 alignments. Columns indicate the alignment program: MMLigner, DALI, TM-Align, LGA, CE, and FatCat. Rows indicate the alignment quality criteria. Grouped within each column (left-to-right): Family, Superfamily, Fold, Class, Decoy.

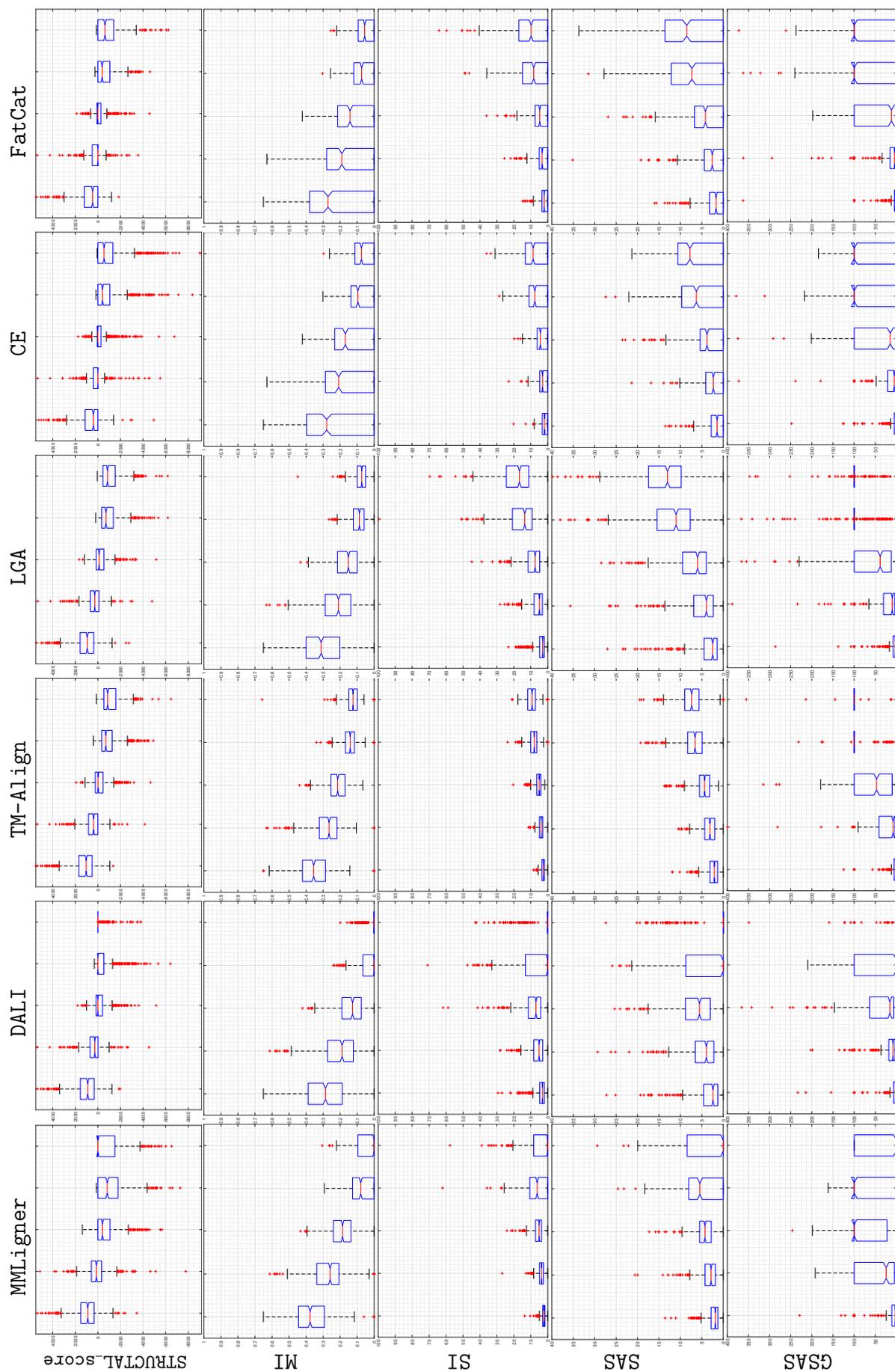


Figure 6.6: (Continued from Figure 6.5). Notched box-and-whisker plots of the 2500 alignments. Columns indicate the alignment program: MMLigner, DALI, TM-Align, LGA, CE, and FatCat. Rows indicate the alignment quality criteria. Grouped within each column (left-to-right): Family, Superfamily, Fold, Class, Decoy.

generated by **MMLigner** are considered at least competitive according to other popular measures of structural alignment quality. Sometimes performing better on a scoring function than the native alignment program, and sometimes revealing flaws in a scoring function when the score is used for general purpose structural alignment.

6.5 Conclusions

The importance of finding biologically meaningful structural alignments has led to the intensive development of methods for generating alignments and evaluating their quality. These methods, however, often produce conflicting results and none has been generally accepted as clearly superior (Kolodny et al., 2005; Sippl and Wiederstein, 2008; Hasegawa and Holm, 2009; Slater et al., 2013; Ma and Wang, 2014).

This chapter builds upon the MML based *I*-value structural alignment quality assessment measure developed in Chapters 4 and 5, to present a very reliable pairwise alignment program called **MMLigner**. **MMLigner** is not only able to find high quality alignments for protein structure pairs, but is also able to explore a range of potential significant alternative alignments. The results of rigorous testing of **MMLigner** on a large data set of domain pairs from the SCOP database and selected difficult to align case studies are presented. As noted in previous chapters, there is not gold standard against which to decide which program produces the best alignments. That being said, the performance of **MMLigner** is highly competitive compared to several popular structural alignment programs, and indeed is more reliable than others in exploring alternate structural alignments (when they exist) and when dealing with hard structural alignment cases.

Chapter 7

Ancillary Methods: (1) Comparing Top k Lists and (2) Sufficient Statistics of Least-Squares Superposition

“The asymptotically best algorithms frequently turn out to be worst on all problems for which they are used.”

— Cantor and Zassenhaus (1981)

This chapter presents the ancillary computational methods used in Chapter 4 to compare ranked lists; and in Chapter 6 to rapidly compute seed alignments. The chapter is divided into two main parts. The first introduces a new information measure for comparing any two top k lists. It provides an objective trade-off between criteria that measure the dis-similarity between lists, addressing pitfalls in the existing measures.

The second part presents a set of sufficient statistics for the least-squares superposition problem under the least squares criterion. These statistics provide an efficient way to operate (via addition and deletion of vectors) on previously computed superpositions. Benchmarking demonstrates a drastic improvement in the computational effort required to compute RMSD using sufficient statistics.

The research presented in this chapter was published in the following papers:
Collier, J. H., Konagurthu, A. S. (2014). An information measure for comparing top- k lists. In *IEEE 10th International Conference on eScience (eScience)*. pp. 127–134.

Konagurthu A. S., Kasarapu, P., Allison, L., Collier, J. H., Lesk, A. M. (2015), On Sufficient Statistics of Least-Squares Superposition of Vector Sets. *Journal of Computational Biology*. **22**(6): 487–497.

7.1 An Information Measure for Comparing Top k Lists

Ranked results are produced in diverse settings, from the web page results of search engines to genes in differential gene co-expression experiments or comparisons of alignment quality across different measures, as seen in Section 4.8.3. A routine task that follows is the assessment of variability between top- k ranked elements between two or more related lists.

Comparing top k lists has received much attention over the past decade. Among the most cited work on this topic is that of Fagin et al. (2003), which proposes an easy-to-compute metric built on Spearman's foot rule (Spearman, 1904). Formally, if π_1 and π_2 define two permutations from the symmetric group S_n of all permutations of n elements, Spearman's foot rule gives the L_1 distance between the ranks of corresponding elements in the two permutations as: $L_1(\pi_1, \pi_2) = \sum_{i=1}^n |\pi_1(i) - \pi_2(i)|$, where any $\pi_1(i)$ or $\pi_2(i)$ is the position (rank) of the i th element in the permutation, given some total ordering of n elements. Fagin et al. (2003) extend this metric to compare two top k lists in the presence of non-overlapping elements (*i.e.*, elements that are in one list but not in the other). This is achieved by fixing the contribution, to the distance, of the non-overlapping elements to a value greater than k , typically $(k + 1)$. Formally, the extended metric for two top k lists τ_1 and τ_2 is defined as $L_1(\tau_1, \tau_2) = 2(k - |\tau_1 \cap \tau_2|)(k + 1) + \sum_{i \in \tau_1 \cap \tau_2} |\tau_1(i) - \tau_2(i)| - \sum_{i \in \tau_1 - \tau_2} \tau_1(i) - \sum_{i \in \tau_2 - \tau_1} \tau_2(i)$ where $\tau_1 \cap \tau_2$ is the set of elements that overlap between the two lists, $|\tau_1 \cap \tau_2|$ denotes the number of overlapping elements, $\tau_1 - \tau_2$ gives the non-overlapping elements in τ_1 , and $\tau_2 - \tau_1$ gives those in τ_2 .

Although this measure can be shown to have good mathematical properties, in practice, it has crucial limitations. Importantly, the term $2(k - |\tau_1 \cap \tau_2|)(k + 1)$ grows quadratically for increasing values of k and decreasing proportion of overlapping elements. In fact, in many applications requiring comparison of top k lists (*e.g.*, web search results), non-overlapping elements form a significant proportion of the lists. Furthermore, this metric is insensitive to the absolute ranks of the overlapping elements in the respective lists; when computing the L_1 distance, the overlapping elements are re-ranked and, hence, ignore the displacement of these elements when comparing two lists.

Another commonly used metric is based on Kendall tau distance (Kendall, 1938), or, colloquially, the *bubble-sort distance*, since it measures the number of adjacent transpositions required to convert (*i.e.*, sort) one permutation to another. Formally, for any two permutations π_1 and π_2 , Kendall tau distance is defined (using the same notations as above) as $K(\pi_1, \pi_2) = \sum_{\forall 1 < i < j \leq n} \kappa_{i,j}(\pi_1, \pi_2)$, where $\kappa_{i,j}(\pi_1, \pi_2) = 0$ if $\pi_1(i) < \pi_1(j)$ and $\pi_2(i) < \pi_2(j)$, or $\kappa_{i,j}(\pi_1, \pi_2) = 1$ otherwise. Extending this idea, the following cost function was proposed to compare two top k lists (Fagin et al., 2006): $K(\tau_1, \tau_2) = (k - |\tau_1 \cap \tau_2|)((2 + p)k - p|\tau_1 \cap \tau_2| + 1 - p) + \sum_{i \in \tau_1 \cap \tau_2} \kappa_{i,j}(\tau_1, \tau_2) - \sum_{i \in \tau_1 - \tau_2} \tau_1(i) - \sum_{i \in \tau_2 - \tau_1} \tau_2(i)$ where, p is a tunable penalty parameter to account for the transposition distance between non-overlapping elements in τ_1 and τ_2 . However, it is easy to see that this metric is also sensitive to the size of non-overlapping elements in the two lists, in addition to the choice of penalty parameter p .

Other measures have been proposed on specialised applications (Bar-Ilan et al., 2006; Budinska et al., 2011; Fury et al., 2006; Pearson, 2007; Jurman et al., 2009, 2012). Noteworthy among these is the Canberra distance (Lance and Williams, 1966) between top k lists (Jurman et al., 2012). This distance is a weighted variant of Spearman's L_1 distance, and ensures that the displacement of elements with higher ranks is penalised more than those with lower ranks.

Here, a new information measure is introduced to compare any two top k lists. The method is built on the statistical framework of minimum message length encoding (MML; Wallace and Boulton (1968); Wallace (2005); see Section 3.3.3), and investigates the compressibility of top k lists. Intuitively, closely related lists have more information in common (and are,

hence, more compressible) than the lists that are poorly related. Thus, the *length* of the losslessly compressed message gives a natural and rigorous measure to estimate the variability between two lists. Unlike previous work, this measure achieves an *objective* trade-off between conflicting criteria, measuring the variability between lists. Mainly, these include: (1) the measurement of dissimilarity, (2) the measurement of disarray of its overlapping elements, and (3) the displacement of the positions (ranks) of these elements.

Note that measuring the *true* information content of any data is incomputable. This follows from the fact that Solomonoff-Kolmogorov-Chaitin Complexity (Kolmogorov, 1963; Solomonoff, 1964; Chaitin, 1966) is undecidable. However, effective and efficient statistical models for data compression provide reasonable upper bounds (*i.e.*, estimates) of true information content. Further, this section provides an approach to estimate the information content in any given pair of top k lists. To keep this approach general, the models of compression use simple priors. However, it is important to note that this information theoretic framework can be adapted to individual contexts by accommodating prior knowledge about rankings in those settings.

The rest of the section is organised as follows. Section 7.1.1 introduces the information measure formally and describes some interesting mathematical properties. Section 7.1.2 explains the practical details involved in estimating the information content of two lists. Section 7.1.3 benchmarks the information measure with other popular distance metrics on ranked lists.

7.1.1 Information Measure for Comparing Ranked Lists

The mathematical underpinning of the information measure to compare any two top k lists is established here. To do this, this section recalls concepts introduced earlier in Section 3.2. For details of how this information measure is realised in practice, see Section 7.1.2.

Definition 1. (*Shannon's information content of an outcome*)

The information content of an outcome E whose probability is $\Pr(E)$ is given by $I(E) = -\log(\Pr(E))$.

We note that $I(E)$ corresponds to the (theoretical) lower bound on the length of the optimal code required to *losslessly* encode the outcome E , as shown by Shannon's seminal work on the mathematical theory of communication (Shannon, 1948).

Lemma 1. (*Measure of Information between two top k lists*)

For two top k lists, τ_1 and τ_2 , the total amount of information contained in them is $I(\tau_1, \tau_2) = I(\tau_1) + I(\tau_2|\tau_1)$

Proof. Using the product rule of probability, the joint probability of the two top k lists, $\Pr(\tau_1, \tau_2)$ is the product of the probability of the first list, $\Pr(\tau_1)$, and the probability of the second list conditioned on the first, $\Pr(\tau_2|\tau_1)$:

$$\Pr(\tau_1, \tau_2) = \Pr(\tau_1) \times \Pr(\tau_2|\tau_1)$$

Taking the negative logarithm on both sides and applying Shannon's insight in Definition 1 to these probabilities, gives the identity: $I(\tau_1, \tau_2) = I(\tau_1) + I(\tau_2|\tau_1)$ \square

Lemma 2. $I(\tau_1, \tau_2) \leq I(\tau_1) + I(\tau_2)$

Proof. When the two top k lists are independent of each other

$$\Pr(\tau_1, \tau_2) = \Pr(\tau_1) \times \Pr(\tau_2|\tau_1) = \Pr(\tau_1) \times \Pr(\tau_2)$$

This implies that $\Pr(\tau_1, \tau_2) \geq \Pr(\tau_1) \times \Pr(\tau_2)$. Translating this into information terms by taking the negative logarithm on both sides, results in $I(\tau_1, \tau_2) \leq I(\tau_1) + I(\tau_2)$. \square

Informally, if τ_1 and τ_2 are independent of each other, that is, if the knowledge of one list does not inform the contents of the other list, the joint information content in these lists is the sum of the information content in each of the lists taken separately, *i.e.*, $I(\tau_1) + I(\tau_2)$. This Section uses the term $NULL(\tau_1, \tau_2)$ in this work to define this upper bound on the joint information in the two top k lists.

Lemma 3. *Given three top k lists, τ_1 , τ_2 and τ_3 ,*

$$I(\tau_1, \tau_2) - I(\tau_1, \tau_3) = \log \left(\frac{\Pr(\tau_3|\tau_1)}{\Pr(\tau_2|\tau_1)} \right)$$

Proof. Using Lemma 1

$$\begin{aligned} I(\tau_1, \tau_2) &= I(\tau_1) + I(\tau_2|\tau_1) \quad \text{and} \\ I(\tau_1, \tau_3) &= I(\tau_1) + I(\tau_3|\tau_1). \end{aligned}$$

Subtracting the two terms,

$$\begin{aligned} I(\tau_1, \tau_2) - I(\tau_1, \tau_3) &= I(\tau_1) + I(\tau_2|\tau_1) - (I(\tau_1) + I(\tau_3|\tau_1)) \\ &= I(\tau_2|\tau_1) - I(\tau_3|\tau_1) \\ &= -\log(\Pr(\tau_2|\tau_1)) + \log(\Pr(\tau_3|\tau_1)) \\ &= \log \left(\frac{\Pr(\tau_3|\tau_1)}{\Pr(\tau_2|\tau_1)} \right). \end{aligned}$$

\square

In other words, the difference above gives the log-odds conditional probability (or posterior) ratio between the lists being compared. This establishes a rigorous statistical framework to compare ranked lists.

A corollary of this property is that the information divergence between any two given top k lists can be defined by treating τ_3 as a separate list which happens to be an identical copy of τ_1 as defined below.

Definition 2. *(Information divergence or cost)*

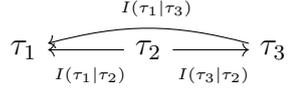
*The information distance between two top k lists is measured as $I(\tau_1, \tau_2) - I(\tau_1, (\tau_3 \equiv \tau_1)) = I(\tau_2|\tau_1) - I((\tau_3 \equiv \tau_1)|\tau_1)$.**

Since the information divergence defined above is a function of conditional information terms on the right hand side, we analyse below the metrical properties of conditional information between top k lists.

Property 1. *(Directed acyclic triangular inequality of conditional information)*

For three top k lists, τ_1 , τ_2 , and τ_3 :

* $I((\tau_3 \equiv \tau_1), \tau_1)$ should not be confused with $I(\tau_1)$ as the former measures the joint information in two separate lists that happen to be identical. For $I(\tau_3 \equiv \tau_1, \tau_1) = I(\tau_1) + I((\tau_3 \equiv \tau_1)|\tau_1) = I(\tau_1)$ implies that the conditional probability $I((\tau_3 \equiv \tau_1)|\tau_1) = 0$, which is the same as saying $\Pr((\tau_3 \equiv \tau_1)|\tau_1) = 1$, suggesting absolute certainty that τ_3 is identical to τ_1 ; this will be incorrect.

$$I(\tau_1|\tau_2) \leq I(\tau_1|\tau_3) + I(\tau_3|\tau_2)$$


Proof. This follows by expanding the joint information in the three lists as follows:

$$\begin{aligned} I(\tau_1, \tau_2, \tau_3) &= I(\tau_3) + I(\tau_1, \tau_2|\tau_3) = I(\tau_2) + I(\tau_1, \tau_3|\tau_2) \\ &= I(\tau_3) + I(\tau_1|\tau_3) + I(\tau_2|\tau_1, \tau_3) \\ &= I(\tau_2) + I(\tau_1, \tau_3|\tau_2) \\ &= I(\tau_3) + I(\tau_1|\tau_3) + I(\tau_2|\tau_3) \\ &\geq I(\tau_2) + I(\tau_1, \tau_3|\tau_2) \end{aligned}$$

Rearranging terms:

$$\begin{aligned} &(I(\tau_3) + I(\tau_2|\tau_3)) + I(\tau_1|\tau_3) \geq I(\tau_2) + I(\tau_1, \tau_3|\tau_2) \\ \implies &I(\tau_2, \tau_3) + I(\tau_1|\tau_3) \geq I(\tau_2) + I(\tau_1, \tau_3|\tau_2) \\ \implies &(I(\tau_2, \tau_3) - I(\tau_2)) + I(\tau_1|\tau_3) \geq I(\tau_1, \tau_3|\tau_2) \\ \implies &I(\tau_3|\tau_2) + I(\tau_1|\tau_3) \geq I(\tau_1, \tau_3|\tau_2) \\ \implies &I(\tau_3|\tau_2) + I(\tau_1|\tau_3) \geq I(\tau_1|\tau_2) + I(\tau_3|\tau_1, \tau_2) \\ \implies &I(\tau_3|\tau_2) + I(\tau_1|\tau_3) \geq I(\tau_1|\tau_2) \end{aligned}$$

□

Property 2. (*Conditional Information is not symmetric*)

$$I(\tau_1|\tau_2) \neq I(\tau_2|\tau_1)$$

Proof. From the product rule of probability (Bayes and Price, 1763):

$$\Pr(\tau_1, \tau_2) = \Pr(\tau_1) \Pr(\tau_2|\tau_1) = \Pr(\tau_2) \Pr(\tau_1|\tau_2).$$

Applying Definition 1:

$$I(\tau_1, \tau_2) = I(\tau_1) + I(\tau_2|\tau_1) = I(\tau_2) + I(\tau_1|\tau_2)$$

Rearranging the terms above:

$$I(\tau_1|\tau_2) = I(\tau_2|\tau_1) + I(\tau_2) - I(\tau_1),$$

proving the asymmetry between $I(\tau_1|\tau_2)$ and $I(\tau_2|\tau_1)$. □

Property 3. (*Near-coincidence of conditional information*)

For any two separate yet identical top k lists τ_1 and $\tau_2 \equiv \tau_1$: $I((\tau_2 \equiv \tau_1)|\tau_1) = \epsilon$, where ϵ is some small constant.

Proof. This follows because $I(\tau_1, \tau_2) = I(\tau_1) + I(\tau_1|\tau_2)$. Since τ_2 is the same as τ_1 , the additional conditional information required to state a list that is an identical copy of another list, is a small constant (in number of bits/nits required to transmit this identity). □

These properties suggest that the information measure defined here possesses near-metrical properties, which are useful to compare top k lists.

7.1.2 Realising the Information Measure in Practice

This section describes an approach to realise an information theoretic measure to quantify the variability of two top k lists. This information measure can be better understood in the context of a communication process between an imaginary pair of transmitter (Alice) and receiver (Bob).

Imagine Alice has access to two top k lists τ_1 and τ_2 . Her goal is to communicate the information in both these lists to Bob *losslessly* – exactly as she sees it. To achieve this, Alice constructs a two-part message. In the first, she will transmit τ_1 taking $I(\tau_1)$ bits. In the second, she uses the commonality (if any) between the two lists so that τ_2 can be transmitted more concisely; this takes $I(\tau_2|\tau_1)$ bits.

In this information theoretic framework the measure of (dis-)similarity between two top k lists is the *total length of this two-part message*: $I(\tau_1) + I(\tau_2|\tau_1)$. It is easy to see that if $\tau_2 \equiv \tau_1$, the second part is extremely concise. On the other hand, if τ_2 is completely unrelated to τ_1 , then the amount of information to transmit the second list given the first, $I(\tau_2|\tau_1)$, cannot be shorter than $I(\tau_2)$.

To achieve lossless transmission of the two lists between Alice and Bob, the following pieces of information are necessary:

1. The size $k = |\tau_1| = |\tau_2|$ of these top k lists.
2. The elements in τ_1 , in the order they appear.
3. The overlapping elements between τ_1 and τ_2 .
4. The ranks of the overlapping elements in τ_2 .
5. The permutation of overlapping elements in τ_2 with respect to the ordering defined by τ_1 .
6. The non-overlapping elements in τ_2 in the order they appear in that list.

In transmitting the above information, two distinct cases have to be considered:

Case 1: When the domain of elements being ranked is *known*, of which τ_1 and τ_2 are (partial) top k lists. For instance, it is common in differential gene expression studies for the total domain of genes and their labels (identifiers) to be known, and a set of top 50 differentially expressed genes between two experiments be considered.

Case 2: Conversely, when the domain of ranked elements remains *unknown*. For instance, in search results from popular web search engines, while the top search results can be seen, the number of pages each search engine indexes is variable (and unknown) and is often smaller than the universe of pages available on the internet.

In the remaining part of this section, these two cases are handled separately and the description the encoding schemes to transmit each of the enumerated pieces of information follows.

Case 1: Known Domain of Elements

Assume the size (N) of the domain is known, along with the labels (or identifiers) of elements in it.

Step 1: Transmitting the size of the top k lists The size of $k \leq N$ is transmitted as an integer code. Since both Alice and Bob know that the top k lists come from a domain of N elements, a simple encoding of k takes $\log(N)$ bits, assuming a uniform distribution over the choices of k in the range $1 \leq k \leq N$. Note that more sophisticated encodings can be defined if there is a prior belief that the distribution of k is non-uniform.

Step 2: Transmitting τ_1 Transmitting the information in τ_1 can be achieved by communicating, over an integer code, the lexicographic index associated with τ_1 in some (mutually agreed) lexicographic ordering of the k -permutations of N elements. Since both Alice and Bob know the domain from which the ranking was generated, the lexicographic ordering of k -permutations can be treated as a part of the code book of communication.

Step 3: Transmitting overlapping elements between τ_1 and τ_2 At this stage Bob already knows τ_1 . To nominate the overlapping elements, that is, the intersection between the two top k lists, a bit mask b_1 is defined, where the set bits indicate the positions in τ_1 where the overlapping elements reside. The transmission complexity of stating the intersection between τ_1 and τ_2 is same as the complexity of this bit mask. An efficient encoding scheme to transmit this bit mask, assuming no prior knowledge about the distribution of the set bits, would be using an adaptive code over a binomial distribution.

The mask b_1 is a binary sequence of length k . The adaptive encoding requires maintaining two running counters that count incrementally the number of 0s and number of 1s, starting from an initial value of 1. Traversing the bit mask left to right, for every symbol in b_1 , Alice estimates its probability by dividing the current state of the symbol's counter by the sum of the two counters. After the probability is estimated, Alice increments the corresponding counter by 1. The code length to state each symbol is the negative logarithm of its estimated probability. Generalising this, if $cnt[0]$ is the number of 0s and $cnt[1]$ be the number of 1s in any bit mask of size k , then the length of the message to transmit this bit mask is $-\log_2 \left(\frac{cnt[0]! \times cnt[1]!}{(k+1)!} \right)$ bits. Figure 7.1(a) gives an example. Notice that both Alice and Bob initialise their counters to 1. Alice encodes each symbol in the bit mask and transmits it before incrementing the corresponding counter at her end. Bob decodes the received symbol using the same estimate of the probability and updates the counters on his side, thus keeping both counters synchronised to achieve a lossless communication.

Step 4: Transmitting absolute positions in τ_2 of the overlapping elements. This again defines another bit mask, b_2 . It is easy to see that there are $\binom{k}{cnt[1]}$ possible candidates for b_2 , given that Bob already knows b_1 . Therefore, assuming these candidates are uniformly distributed, the optimal message length to state b_2 takes $\log \binom{k}{cnt[1]}$ bits. It is important to note that b_2 ignores the permutation of the overlapping elements as they appear in τ_2 (with respect to τ_1) – this is handled in the next step. While the above encoding is optimal, it does not account for the displacement of overlapping elements in terms of their absolute ranks in the list. It might arise in some applications that the displacement is among the criteria of comparing two lists. Hence, a modified adaptive scheme is proposed to account for this displacement. Two counters are used; the first tracks the number of times the symbols in bit masks b_1 and b_2 remain the same at a given position (column); the second tracks the number of times they are different. These counters are used to estimate the probabilities while traversing along b_2 . For example, See Figure 7.1(b).

b_1	0 0 1 1 0 0 1 0 0 0
$cnt[0]$	1 2 3 3 3 4 5 5 6 7
$cnt[1]$	1 1 1 2 3 3 3 4 4 4
Prob.	$\frac{1}{2} \frac{2}{3} \frac{1}{4} \frac{2}{5} \frac{3}{6} \frac{4}{7} \frac{3}{8} \frac{5}{9} \frac{6}{10} \frac{7}{11}$

(a)

b_1	0 0 1 1 0 0 1 0 0 0
b_2	0 0 1 1 1 0 0 0 0 0
$cnt[0 0]$ or $cnt[1 1]$	1 2 3 4 5 5 6 6 7 8
$cnt[0 1]$ or $cnt[1 0]$	1 1 1 1 1 2 2 3 3 3
Prob.	$\frac{1}{2} \frac{2}{3} \frac{3}{4} \frac{4}{5} \frac{1}{6} \frac{5}{7} \frac{2}{8} \frac{6}{9} \frac{7}{10} \frac{8}{11}$

(b)

Figure 7.1: Examples of the adaptive encoding schemes for bit masks described in the main text.

Step 5: Transmitting the permutation of overlapping elements in τ_2 with respect to τ_2 . From the previous step, Bob knows what the overlapping elements between the lists are, but does not know in what order they appear in τ_2 . To transmit the permutation of these overlapping elements efficiently, a lexicographic numbering can be mutually agreed between them (as a part of the code book). Then, transmitting the permutation of these elements requires simply communicating its lexicographic number over some integer code.

However, to make this transmission efficient a factoradic (or mixed factorial) base numbering system can be employed (Knuth, 1999b). This system defines a bijection between the symmetric group S_n to $n!$ possible permutations in that group.

Concretely, let $\pi = \{\pi(1), \pi(2), \dots, \pi(n)\}$ be some permutation of n symbols, where $\pi(i)$ is the rank of the i th element in the permutation. A factoradic of π defines a sequence $f(\pi) = (f^1, f^2, \dots, f^n)_!$, where any f^i is the number of j s greater than i such that $\pi(i) < \pi(j)$. See Figure 7.2 for an example of a lexicographic ordering of the symmetric group S_4 labeled by elements ‘a, b, c, d’, along with its corresponding factoradic sequence of digits. It can be observed that each factoradic digit f^i denotes the number of successive *adjacent transpositions* on π required to move each $\pi(i)$ th element into its correct position. The permutation index (in decimal) can be computed from a factoradic as $\sum_{i=0}^n f^i \times (n - i - 1)!$.

The sequence of digits $f(\pi)$ has several interesting properties. It has been shown that, if permutations in S_n are distributed uniformly, each factoradic digit f^i is also uniformly distributed in the range $0 \leq f^i \leq (n - i - 1)$ (Lehmer, 1960). Also, the factoradic digits f^i are mutually independent of each other because they form projections on independent factors in the product $n \times (n - 1) \times \dots \times 1 \equiv n!$

Thus, transmitting a permutation of overlapping elements in τ_2 involves transmitting its factoradic digits in sequence. For each factoradic digit f^i in the range $0 \leq i < n$ (note: f^n is always 0), any decreasing probability distribution on integers in that range can be used. Specifically a Wallace tree code (Wallace and Patrick, 1993), which defines a code over positive integers[†] by associating each integer with the binary code used to uniquely identify a binary tree. Since this integer code is defined over the infinite space of positive integers, the probability associated with each code is normalised such that the total probability in the finite range $0 \leq f^i \leq n - i - 1$ adds up to 1.

Step 6: Transmitting non-overlapping elements in τ_2 . Given that the domain of elements is known and is of size N , each non-overlapping element can be stated in $\log_2(N - |\tau_1 \cup \tau_2|)$ bits. With this the communication process concludes.

[†]Since factoradic digits start from 0, just add 1 to each digit to map it to the Wallace tree code.

0_{10}	a b c d (0 0 0 0) _!	6_{10}	b a c d (1 0 0 0) _!	12_{10}	c a b d (2 0 0 0) _!	18_{10}	d a b c (3 0 0 0) _!
1_{10}	a b d c (0 0 1 0) _!	7_{10}	b a d c (1 0 1 0) _!	13_{10}	c a d b (2 0 1 0) _!	19_{10}	d a c b (3 0 1 0) _!
2_{10}	a c b d (0 1 0 0) _!	8_{10}	b c a d (1 1 0 0) _!	14_{10}	c b a d (2 1 0 0) _!	20_{10}	d b a c (3 1 0 0) _!
3_{10}	a c d b (0 1 1 0) _!	9_{10}	b c d a (1 1 1 0) _!	15_{10}	c b d a (2 1 1 0) _!	21_{10}	d b c a (3 1 1 0) _!
4_{10}	a d b c (0 2 0 0) _!	10_{10}	b d a c (1 2 0 0) _!	16_{10}	c d a b (2 2 0 0) _!	22_{10}	d c a b (3 2 0 0) _!
5_{10}	a d c b (0 2 1 0) _!	11_{10}	b d c a (1 2 1 0) _!	17_{10}	c d b a (2 2 1 0) _!	23_{10}	d c b a (3 2 1 0) _!

Figure 7.2: All possible permutation of elements a, b, c, d , their lexicographical number in base 10, and their corresponding sequence of digits in a factorial number system. The factoradic system defines a bijection between the permutation and its lexicographic number. For example, $21_{10} = (3, 1, 1, 0)_! = 3 \times 3! + 1 \times 2! + 1 \times 1! + 0$.

Case 2: Unknown Domain of Elements

In this situation Alice and Bob do not know the domain of elements being sorted. In the previous case, Steps 1,2 and 6 depended on knowing the domain and, hence, require modification. The encodings for Steps 3,4, and 5 are as previously described.

Since this framework relies on lossless transmission and there is no prior knowledge of the domain of possible labels in each of the two top k lists, the framework requires the lists (along with its labels) to be explicitly communicated.

To efficiently communicate τ_1 and the non-overlapping elements in τ_2 , consider the union of the two lists, $\tau_1 \cup \tau_2$, such that the top k elements define labels in τ_1 (in that order) and the remainder are the labels of non-overlapping elements in the order they appear in τ_2 .

First, the size of the union $|\tau_1 \cup \tau_2|$ is transmitted using the Wallace tree code defined over all positive integers. (This modifies previous Step 1.) Then the labels in the $\tau_1 \cup \tau_2$ can be compressed using, for instance, a standard, dictionary-based lossless data compression algorithm of Lempel-Ziv-Welch (*LZW*) (Ziv and Lempel, 1978). The length, $|LZW(\tau_1 \cup \tau_2)|$, in bits gives the cost to state the information in τ_1 and non-overlapping elements in τ_2 . (This modifies previous Steps 2 and 6).

Computational complexity of computing various code length terms. For case 1: The code lengths involved in steps 1, 2, and 6 take $O(1)$ time to compute. The adaptive code lengths in Steps 3 and 4 take $O(k)$ time. In Step 5, finding the code length corresponding to the factoradic of a permutation of n overlapping elements can be achieved in $O(n \leq k)$ time (Myrvold and Ruskey, 2001). Computing the code length of each factoradic takes $O(1)$ time. Thus, the total computational complexity to estimate the information content in the two lists grows as $O(k)$. For case 2: Step 1 requires $O(1)$ time. Steps 2 and 6 are dealt together and involves compression of labels in the set $\{\tau_1 \cup \tau_2\}$. It can easily be seen that $k \leq |\tau_1 \cup \tau_2| \leq 2k$. *LZW* compression implemented naively has a time complexity of $O(S|\aleph|)$, where S is the number of input symbols and \aleph is the alphabet. For most practical applications, both S and $|\aleph|$ are $O(k)$ in size. Steps 3, 4, and 5 are the same as in case 1. Thus, the total time complexity to estimate of the information content for this case grows as $O(k^2)$.

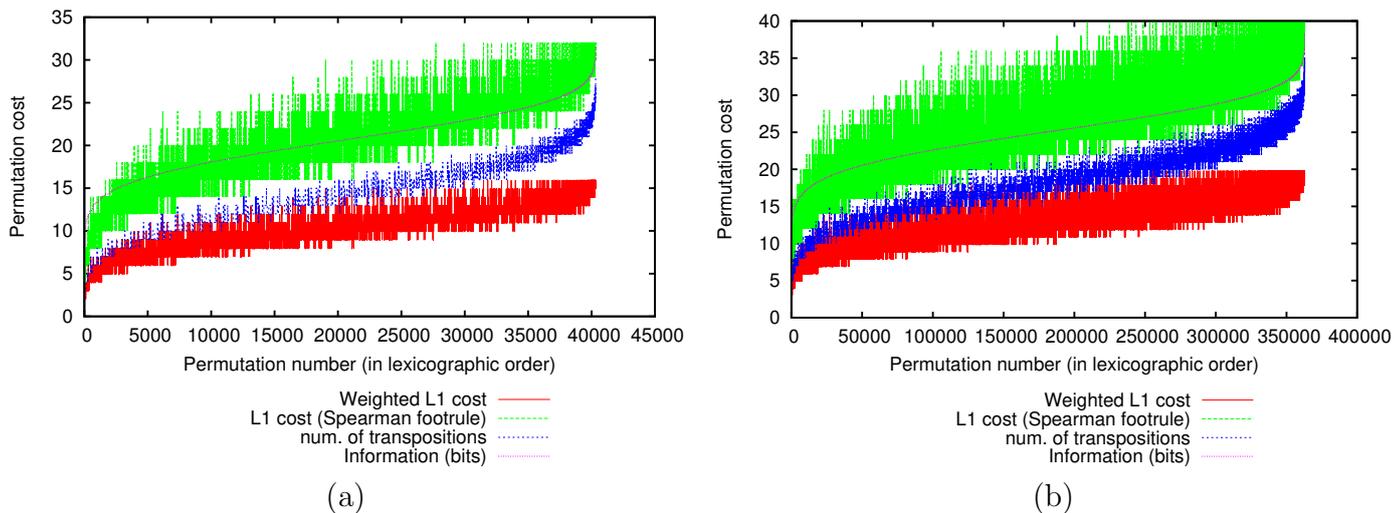


Figure 7.3: Variation of costs over the set of all permutations in the symmetric groups (a) S_8 , and (b) S_9 . Spearman’s foot rule distance is in Green. Kendall tau distance is in Blue. Canberra distance is given in Red. Information measure defined in this work is given in Magenta.

7.1.3 Results and Discussion

Firstly the effect of disarray between permutations as assessed by various popular measures is quantified. Figure 7.3 gives the cost associated with various measures for the set of all permutations in symmetric groups (a) S_8 and (b) S_9 . Specifically, the measures used are: (1) Spearman’s foot rule metric (L_1 distance), (2) Canberra distance (weighted L_1 measure), (3) Kendall’s tau distance, measuring the number of adjacent transpositions to sort a permutation, and (4) the information measure developed in this work. It can be seen that as the information content to describe a permutation increases, all the other measures vary significantly. It is important to note that the costs reported by all four of the considered measures are related to the total number of adjacent transpositions of elements required to sort the permutation. However, the measure of information accounts for the varying magnitude of disarray (given by the permutation’s factoradic digits) of each element, instead of combining and summarising using a simple number. Other measures overlook these individual contributions. For instance, it can be seen from Figure 7.2 that the permutations $adcb = 5_{10} = (0, 2, 1, 0)_1$, $bcda = 9_{10} = (1, 1, 1, 0)_1$, $bdca = 10_{10} = (1, 2, 0, 0)_1$, $cadb = 13_{10} = (2, 0, 1, 0)_1$, and $dabc = 18_{10} = (3, 0, 0, 0)_1$ all require the same number of transpositions ($= 3$), yet differ in the number of transpositions for its individual elements.

To examine the performance of various measures on comparing top k lists, first consider three top 250 movie lists downloaded from goodmovieslist.com, imdb.com and reddit.com. Figure 7.4 shows the comparisons of (left to right) Goodmovies vs. IMDb, Goodmovies vs. Reddit, and IMDb vs. Reddit, varying k from 1 to 250 in increments of one. Qualitatively the lists corresponding to Goodmovies and IMDb are more similar than the other possible pairs. It can be seen from the figure that both Spearman’s foot rule distance and Kendall tau distance grow roughly quadratically with the size of k . This mainly results from the contributions to the respective costs from the set of non-overlapping elements. As this set grows, its contribution to the distance dominates. However, the growth of information distance[‡] is roughly linear. This

[‡]Note that, for these results and those to follow, the measure of information between lists is computed by assuming that the total domain of movies is unknown, *i.e.*, following case 2 described in Section 7.1.2.

makes more sense, as information is additive. When the size of the list increases from k to $k + 1$, the new element that gets added to each of the two lists can in the worst case be independent of the previous information. This implies that, in the worst case, the total information content in the list going from k to $k + 1$ gets augmented by the sum of information in the new elements. Therefore, the quadratic growth of the other measures is questionable.

It is interesting to note that while Spearman's foot rule and Kendall tau distances monotonically increase, the information distance plotted in the figure has 'fluctuations' in the amount of information measured. These variations occurs when new elements (for increasing values of k) cause the set of overlapping elements to grow in size. While this increases the distance to state the permutation of overlapping elements, there is a net saving because the size of the set of non-overlapping elements (which are transmitted using *LZW* compression) decreases, in comparison with the previous values of k .

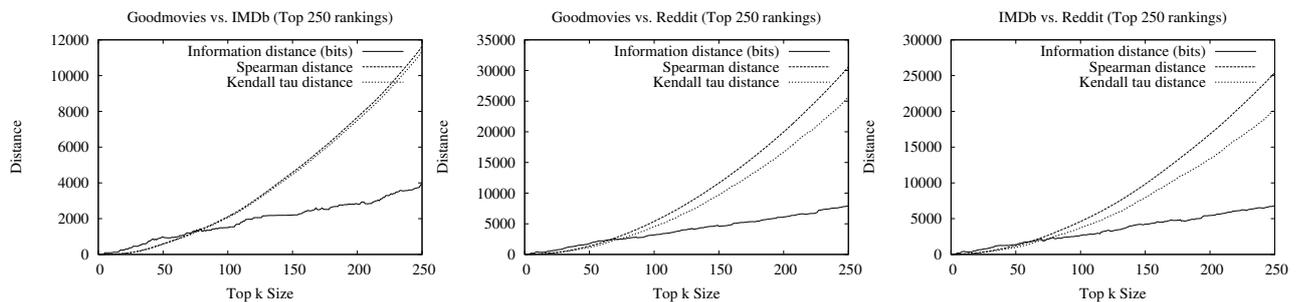


Figure 7.4: Comparison of Information distance, Spearman's distance, Kendall's tau distance on movie rankings from goodmovieslist.com, imdb.com and reddit.com

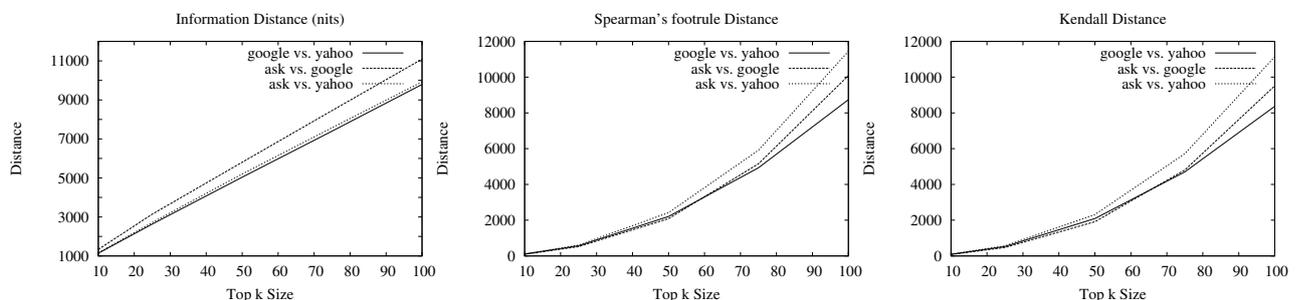


Figure 7.5: Comparison of Information distance, Spearman's distance, and Kendall's tau distance on search results of Google, Yahoo and Ask. The reported values are averaged over 250 top-trending search terms comparing pairs of ranked lists for values of $k = \{10, 25, 75, 100\}$

Further, to undertake this comparison on a larger scale, the search results of three popular web search engines are compared: Google, Yahoo and Ask. This is achieved by selecting 250 top trending search and news terms reported by Google Trends and Yahoo text Analytics for the regions of Australia, US, India, Canada, UK, Singapore and Germany. Figure 7.5 plots the average (mean) distance over the 250 queries computing using Information, Spearman's foot rule and Kendall tau measures. In this experiment, k is varied more coarsely as 10, 25, 50, 75, and 100. In this figure the same growth trends from before emerges, linear growth for information distance and quadratic growth for Spearman's and Kendall's distance. For $k > 50$, the difference between the average distances grows drastically for Spearman's and Kendall's distances, while the same using information distance does not. This suggests a major shortcoming of these measures compared to the measure based on information.

7.2 Sufficient Statistics for Least-Squares Superposition of Vector Sets

7.2.1 Introduction

As seen earlier in Chapters 4-6, the optimal superposition through orthogonal transformation of vector sets is at the core of various methods used in this thesis. Indeed the use of superpositions is widespread in structural biology and bioinformatics (Lesk, 2001b; Eidhammer et al., 2004). This pervasive use is mainly due to its ability to reveal information about similarities and differences between protein substructures.

Recall that the orthogonal transformation involves finding the best rigid-body rotation and translation of two vector sets with one-to-one correspondence. Section 2.2.3 introduced an almost universally used criterion to define the *best* superposition between corresponding vector sets. It involves minimising the *sum-of-squares error* over the entire search space of possible rotations and translations, resulting in a quantitative measure, *root mean square deviation* (or RMSD), after best superposition.

Given the importance of this routine, several approaches have been proposed to address this problem over the years (Kabsch, 1976, 1978; McLachlan, 1982; KenKnight, 1984; Mackay, 1984; Lesk, 1986; Daimond, 1988; Kearsley, 1989; Cohen, 1997; Coutsiias et al., 2004; Koehl, 2001). Among the popular methods to solve this problem is the method of Kabsch (Kabsch, 1978), which solves this problem using Lagrange multipliers that constrain the search to *pure rotations* (and avoid improper ones, like rotoinversions).

As indicated in Section 2.2.3, an equivalent and more elegant approach to solving the same problem was proposed by Kearsley (Kearsley, 1989) using *quaternion* algebra (Hamilton and Hamilton, 1866). Quaternions are generalisations of complex numbers with direct applications to transformations in three dimensional space. Specifically, the space group corresponding to unit quaternions is equivalent to the group of all possible pure rotations in three dimensions (3D) defined about an arbitrary origin. That is, any 3D pure rotation by an angle θ about some normalised axis \hat{n} passing through the origin can be represented using a unit quaternion as follows: $[\cos(\frac{\theta}{2}), \hat{n} \sin(\frac{\theta}{2})]$. Among the key advantages of using Kearsley's quaternion method to solve the least-squares superposition problem are: (1) the problem can be solved analytically in quaternion parameters, and (2) the method avoids problems with singularities (and rotoinversions) that can result from using Kabsch's approach, where these oddities are handled explicitly after the solution is found (Kearsley, 1989; Coutsiias et al., 2004). In general, the least-squares superposition involves a computational effort that asymptotically grows *linearly* with the number of corresponding points being superimposed.

Least-squares superpositions are used extensively in the structural alignment problem, and in this thesis. As described in Section 6.2.1, many popular methods build an alignment between structures using fragment-pair assembly (Lesk, 1986; Shindyalov and Bourne, 1998; Ye and Godzik, 2003; Shatsky et al., 2004; Konagurthu et al., 2006; Shatsky et al., 2002; Vriend and Sander, 1991; Lackner et al., 2000; Kolodny et al., 2005). In fact, the *MMLigner* algorithm defined this thesis (see Chapter 6) also employs a fragment-pair assembly approach to generate an effective seed alignments (see Section 6.3.2). The general strategy involves finding aligned (contiguous) fragment-pairs that are maximally extended, one residue-residue correspondence at a time, starting from some minimum fragment size, until the fragment-pairs superpose within some specified threshold of RMSD. This results in a library of well-fitting fragment pairs, whose

computational effort grows as a cubic in the length of the structures being aligned.[§] Further, by computing the joint superpositions of these well-fitting maximal fragment pairs, a structural alignment can be *assembled* by gathering fragment-pairs that superpose consistently. This involves repeated concatenation of vector sets supporting the fragment-pairs and superposition of those fragment-pairs in the library. Such superpositions are traditionally recomputed from scratch (even though the previous superpositions provide a wealth of information about the joint superpositions). It can be seen that the number of joint superpositions grows quadratically in the size of the fragment-pair library, where each joint superposition of two fragment-pairs taking a linear effort in the size of the concatenated vector sets.

Generating joint superpositions from scratch imposes a significant computational demand when performing a large number of superpositions, as required for computing pairwise structural alignments. One can anticipate that the amount of time spent in superposing fragments becomes computationally impractical when aligning multiple protein structures simultaneously, where the multiple structural alignment is commonly built using all-vs-all pairwise structural alignments, each of which makes a very large number of calls to the superposition routine.

In this section, the least-squares superposition problem is examined and a set of statistics that are sufficient to compute the optimal superposition parameters (RMSD of best superposition, and its corresponding rotation and translation) are derived, with mathematical guarantee of their optimality under the least-squares measure. These *sufficient statistics* (Hogg and Craig, 1994) are additive. Thus, they can be used to compute new superpositions in *constant time*, by relying purely on updating the statistics of the partial superpositions. Constant time update to compute the optimal joint superpositions results in a drastic speed up, when compared to the time it takes to recompute the joint superposition from scratch (by treating it as a new instance of the superposition problem).

The rest of this section is arranged as follows: Section 7.2.2 provides the basic background of the least-squares superposition problem using the widely-used least-squares criterion. Section 7.2.3 introduces the statistical aspects of sufficient statistics, and derives the full set of sufficient statistics for the optimal least-squares superposition problem. Section 7.2.4 provides the update rules to compute superpositions using the sufficient statistics of superpositions of component vector sets, under addition and symmetric difference operations. Section 7.2.5 describes an approach to speed up the diagonalisation step used in the Kearsley approach. Finally, Section 7.2.6 benchmarks the performance gain derived from using sufficient statistics.

7.2.2 Summary of Least-Squares Superposition

Let $\mathbf{U} = \{\vec{u}_1, \dots, \vec{u}_n\}$ and $\mathbf{V} = \{\vec{v}_1, \dots, \vec{v}_n\}$ denote two 3D vector sets with one-to-one correspondence. Let the (x, y, z) components of each \vec{u}_i be represented here as $(\vec{u}_i(x), \vec{u}_i(y), \vec{u}_i(z))$ (a similar representation holds for \vec{v}_i or any other vector). The least-squares superposition problem is a constrained optimisation problem that involves finding the best rotation (matrix) \mathbf{R} and translation (vector) \vec{t} with the optimality criterion defined as in Equation 2.1:

$$\xi = \min \sum_{i=1}^{N_e} \|\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i\|^2$$

where $\|\vec{x}\|$ is the \mathcal{L}^2 -norm of the vector \vec{x} , \mathbf{R} is a 3×3 pure rotation matrix, and \vec{t} is a translation vector.

[§] $O(N^2)$ number of superpositions, each taking $O(N)$ superposition effort, where N is the number of residues in the structures being aligned.

Under this least-squares criterion, the translation with respect to the optimal superposition is independent of rotation. This can be easily seen by differentiating ξ with respect to \vec{t} and evaluating it at its extremum:

$$\begin{aligned} \frac{\partial \xi}{\partial \vec{t}} &= \frac{\partial}{\partial \vec{t}} \sum_{i=1}^{N_e} \|\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i\|^2 = 2 \sum_{i=1}^{N_e} \left(\frac{\partial(\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i)}{\partial \vec{t}} (\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i) \right) = 0 \\ \implies \sum_{i=1}^{N_e} (\mathbf{R}\vec{u}_i + \vec{t} - \vec{v}_i) &= 0 \implies \mathbf{R} \sum_{i=1}^{N_e} \vec{u}_i + N_e \vec{t} - \sum_{i=1}^{N_e} \vec{v}_i = 0 \\ \implies \vec{t} &= \underbrace{\frac{1}{N_e} \sum_{i=1}^{N_e} \vec{v}_i}_{\mathbf{V} \text{ centre-of-mass}} - \mathbf{R} \underbrace{\frac{1}{N_e} \sum_{i=1}^{N_e} \vec{u}_i}_{\mathbf{U} \text{ centre-of-mass}} = \mathbf{Centroid}(\mathbf{V}) - \mathbf{Centroid}(\mathbf{U}) \end{aligned}$$

It follows that moving each of the vector sets to an origin at its centroid, about which the rotation is defined, gives us a modified (but equivalent) objective which is independent of the translation \vec{t} :

$$\xi = \min \sum_{i=1}^n |\mathbf{R}\vec{u}'_i - \vec{v}'_i|^2$$

where, $\vec{u}'_i = \vec{u}_i - \frac{\sum_{i=1}^n \vec{u}_i}{n}$ and $\vec{v}'_i = \vec{v}_i - \frac{\sum_{i=1}^n \vec{v}_i}{n}$.

As seen in Section 2.2.3, Kearsley (1989) proposed an elegant method that removes the non-linear aspect of this problem by transforming it to an eigenvalue problem of the form $\mathbf{Q}\vec{q} = \lambda\vec{q}$, where \mathbf{Q} is a 4×4 square symmetric matrix.

$$\left(\begin{array}{cccc} \sum(x_m^2 + y_m^2 + z_m^2) & \sum(y_p z_m - y_m z_p) & \sum(x_m z_p - x_p z_m) & \sum(x_p y_m - x_m y_p) \\ \sum(y_p z_m - y_m z_p) & \sum(x_m^2 + y_p^2 + z_p^2) & \sum(x_m y_m - x_p y_p) & \sum(x_m z_m - x_p z_p) \\ \sum(x_m z_p - x_p z_m) & \sum(x_m y_m - x_p y_p) & \sum(x_p^2 + y_m^2 + z_p^2) & \sum(y_m z_m - y_p z_p) \\ \sum(x_p y_m - x_m y_p) & \sum(x_m z_m - x_p z_p) & \sum(y_m z_m - y_p z_p) & \sum(x_p^2 + y_p^2 + z_m^2) \end{array} \right) \quad (7.1)$$

$\vec{q} = (q_1, q_2, q_3, q_4)^T = (\cos(\frac{\theta}{2}), \hat{n}(x) \sin(\frac{\theta}{2}), \hat{n}(y) \sin(\frac{\theta}{2}), \hat{n}(z) \sin(\frac{\theta}{2}))^T$ are the (unknown or to be solved) quaternion components associated with some rotation θ about a normalised axis \hat{n} , and λ is an (unknown) eigenvalue. In Equation 7.1, the notation x_m is used to denote the component-wise difference $\vec{v}'_i(x) - \vec{u}'_i(x)$ (and similarly y_m and z_m) and x_p to denote the component-wise sum $\vec{v}'_i(x) + \vec{u}'_i(x)$ (similarly y_p and z_p). From this point onwards, the term *quaternion matrix* is used to indicate the 4×4 square symmetric matrix in Equation 7.1 and denote it as \mathbf{Q} .

Diagonalising this matrix yields four eigenvalues and (corresponding) eigenvectors. The eigenvector corresponding to the smallest eigenvalue, λ_{\min} , corresponds to the rotation producing the least-squares error, and the RMSD is computed as $\sqrt{\frac{\lambda_{\min}}{n}}$

Time complexity: The computational effort that takes to solve the rigid-body superposition problem using Kearsley's quaternion approach (or equivalently Kabsch's approach) grows linearly with the number of vectors being superimposed. In Kearsley's approach this is dominated by the computation of the \mathbf{Q} where each of 10 distinct terms in the matrix requires $O(n)$ effort. The diagonalisation of \mathbf{Q} is independent of n and shows a rapid convergence with numerical methods such as Jacobi's diagonalisation algorithm (Jacobi, 1846).

7.2.3 Sufficient Statistics

The rigid-body superposition problem is a geometric instance of the general regression problem using total least-squares, where a regression line is determined that minimises the sum of the squared errors of the observed data with respect to it.

It is widely known that solving the regression problem produces error terms that are normally distributed as $\mathcal{N}(0, \sigma)$, where the mean μ is 0 and σ is the standard deviation which is minimised by the problem. In fact, the least squares estimator of σ is also its maximum likelihood estimator.

More formally, consider the standard normal distribution of some random variable x :

$$\mathcal{N}(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x - \mu)^2}{2\sigma^2}\right]$$

This normal density can be reparameterised into a general form denoting the family of exponential distributions:

$$f(x|\vec{\eta}) = h(x)g(\vec{\eta}) \exp(\vec{\eta}^T \vec{U}(x))$$

where $h(x) = \frac{1}{\sqrt{\pi}}$, $g(\eta_2) = \sqrt{-\eta_2} \exp\left(\frac{\eta_1^2}{4\eta_2}\right)$, $\vec{\eta}^T = \left(\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}\right)$, $\vec{U}^T(x) = (x, x^2)$.

This transformation can be used to show certain important properties that allows efficient computation of the maximum likelihood estimators of μ and σ .

Considering a sample set of observations that are normally distributed $\mathbf{X} = \{x_1, x_2, \dots, x_n\}$. The likelihood for these samples is given by:

$$f(\mathbf{X}|\vec{\eta}) = \left(\prod_{i=1}^n h(x_i)\right) (g(\vec{\eta}))^n \exp(\vec{\eta}^T \sum_{i=1}^n \vec{U}(x_i))$$

Taking natural logarithms on both sides gives us the log likelihood:

$$\log(f(\mathbf{X}|\vec{\eta})) = \kappa + n \log(g(\vec{\eta})) + \vec{\eta}^T \sum_{i=1}^n \vec{U}(x_i)$$

where $\kappa = \sum_{i=1}^n \log(h(x_i))$ is a term independent of $\vec{\eta}$.

To find the maximum likelihood estimators $\hat{\eta}$, take the gradient with respect to $\vec{\eta}$ and set to 0. This results in:

$$\begin{aligned} n \nabla_{\hat{\eta}} [\log(g(\hat{\eta}))] + \sum_{i=1}^n \vec{U}(x_i) &= 0 \\ \implies -\nabla_{\hat{\eta}} [\log(g(\hat{\eta}))] &= \frac{1}{n} \sum_{i=1}^n \vec{U}(x_i) \\ &= \frac{-1}{g(\hat{\eta})} \nabla_{\hat{\eta}} g(\hat{\eta}) = \frac{1}{n} \sum_{i=1}^n \vec{U}(x_i) \end{aligned}$$

Notice that the maximum likelihood estimate $\hat{\eta}$ depends on the statistic $\sum_{i=1}^n \vec{U}(x_i)$ rather than on the individual data. This suggests that to obtain the maximum likelihood estimate, the data is not needed explicitly as it can be derived from that statistic. This sufficiency to derive the

maximum likelihood estimator without explicit consideration of data makes $\sum_{i=1}^n \vec{U}(x_i)$ a *sufficient statistic* for the exponential family of functions. As seen earlier for normal distribution, $\vec{U}(x_i) = (x_i, x_i^2)$ gives the sufficient statistics of $\sum_{i=1}^n x_i$ and $\sum_{i=1}^n x_i^2$ (Hogg and Craig, 1994).

Sufficient statistics of least-squares superposition

For the least-square superposition, each x , y , and z component of the error vector $\xi_i = \mathbf{R}\vec{u}_i' - \vec{v}_i'$, is assumed to be normally distributed: *i.e.*, $\xi_i(x) \sim \mathcal{N}(\mu = 0, \sigma)$, $\xi_i(y) \sim \mathcal{N}(\mu = 0, \sigma)$, and $\xi_i(z) \sim \mathcal{N}(\mu = 0, \sigma)$. The sufficient statistics are now derived for σ of the $\xi_i(x), \xi_i(y), \xi_i(z)$ terms, which is equivalent to $\frac{\text{RMSD}}{\sqrt{3}}$ after least-squares superposition. The likelihood of the observed normally distributed errors after superposition, $\mathbf{E} = \{\xi_1(x), \xi_1(y), \xi_1(z) \dots, \xi_n(x), \xi_n(y), \xi_n(z)\}$, can be written as:

$$\begin{aligned} f(\mathbf{E}|\sigma) &= \prod_{i=1}^{3n} (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{R}\vec{u}_i' - \vec{v}_i'\|^2\right) \\ &= (2\pi\sigma^2)^{-\frac{3n}{2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^n \|\mathbf{R}\vec{u}_i' - \vec{v}_i'\|^2\right) \end{aligned} \quad (7.2)$$

Examining the decomposition of:

$$\|\xi_i\|^2 = \|\mathbf{R}\vec{u}_i' - \vec{v}_i'\|^2 = \|\vec{u}_i'\|^2 + \|\vec{v}_i'\|^2 - 2\vec{v}_i'^T \mathbf{R}\vec{u}_i' \quad (7.3)$$

From Equation 7.1, the matrix \mathbf{Q} is made up of terms of the form:

$$A_m = v_i'(A) - u_i'(A) \text{ and } A_p = v_i'(A) + u_i'(A)$$

where each A and B take the values $\{x, y, z\}$ denoting vector components. Rewriting:

$$v_i'(A) = \frac{A_p + A_m}{2} \text{ and } u_i'(A) = \frac{A_p - A_m}{2}$$

The first two terms on the right hand side of Equation 7.3 can be expanded as follows:

$$\begin{aligned} \|\vec{u}_i'\|^2 + \|\vec{v}_i'\|^2 &= (u_i'(x)^2 + u_i'(y)^2 + u_i'(z)^2) + (v_i'(x)^2 + v_i'(y)^2 + v_i'(z)^2) \\ &= \frac{1}{2}(x_m^2 + x_p^2 + y_m^2 + y_p^2 + z_m^2 + z_p^2) \\ &= \frac{1}{2} \sum_{A \in \{x, y, z\}} A_m^2 + \frac{1}{2} \sum_{A \in \{x, y, z\}} A_p^2 \end{aligned} \quad (7.4)$$

The last term on the right hand side of Equation 7.3 can be expanded as $\vec{v}_i'^T \mathbf{R}\vec{u}_i' = \vec{v}_i'^T [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{r}_3] \vec{u}_i'$ where $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ are column vectors of the 3×3 rotation matrix \mathbf{R} . Therefore:

$$\vec{v}_i'^T \mathbf{R}\vec{u}_i' = (\vec{v}_i' \cdot \mathbf{r}_1) u_i'(x) + (\vec{v}_i' \cdot \mathbf{r}_2) u_i'(y) + (\vec{v}_i' \cdot \mathbf{r}_3) u_i'(z) \quad (7.5)$$

Take the first term on the right hand side of Equation 7.5. This can be expanded as:

$$\begin{aligned} (\vec{v}'_i \cdot \mathbf{r}_1) u'_i(x) &= r_{11} v'_i(x) u'_i(x) + r_{12} v'_i(y) u'_i(x) + r_{13} v'_i(z) u'_i(x) \\ &= \frac{r_{11}}{4} (x_p + x_m)(x_p - x_m) + \frac{r_{12}}{4} (y_p + y_m)(x_p - x_m) + \frac{r_{13}}{4} (z_p + z_m)(x_p - x_m) \\ &= \frac{r_{11}}{4} (x_p^2 - x_m^2) + \frac{r_{12}}{4} (y_p x_p - y_p x_m + y_m x_p - y_m x_m) \\ &\quad + \frac{r_{13}}{4} (z_p x_p - z_p x_m + z_m x_p - z_m x_m) \end{aligned}$$

where r_{11}, r_{12}, r_{13} are the terms in the \vec{r}_1 column vector in \mathbf{R} . More generally:

$$(\vec{v}'_i \cdot \mathbf{r}_1) u'_i(x) = c_1 A_p^2 + c_2 A_m^2 + c_3 A_p B_p + c_4 A_m B_m + c_5 A_m B_p \quad (7.6)$$

where c_k are constants in terms of components of \vec{r}_1 .

Similarly, $(\vec{v}'_i \cdot \mathbf{r}_2) u'_i(y)$ and $(\vec{v}'_i \cdot \mathbf{r}_3) u'_i(z)$ can be expanded as above and will have the same form as (7.6) but with different constants. Therefore, combining Equations 7.4-7.5, the equation 7.3 can be written as:

$$\|\xi_i\|^2 = \zeta_1 \sum_A A_p^2 + \zeta_2 \sum_A A_m^2 + \zeta_3 \sum_{\forall A \neq B} A_p B_p + \zeta_4 \sum_{\forall A \neq B} A_m B_m + \zeta_5 \sum_{\forall A \neq B} A_m B_p$$

where ζ_k are constants. Hence, the likelihood function can be written as

$$f(\mathbf{E} = \{\xi_1(x), \xi_1(y), \xi_1(z) \dots, \xi_n(x), \xi_n(y), \xi_n(z)\} | \sigma) = (2\pi\sigma^2)^{-\frac{3n}{2}} \exp\left(-\frac{1}{2\sigma^2} \mathbf{U}\right) \quad (7.7)$$

where

$$\mathbf{U} = \sum_{i=1}^n \left(\zeta_1 \sum_A A_p^2 + \zeta_2 \sum_A A_m^2 + \zeta_3 \sum_{\forall A \neq B} A_p B_p + \zeta_4 \sum_{\forall A \neq B} A_m B_m + \zeta_5 \sum_{\forall A \neq B} A_m B_p \right)$$

and $A, B \in \{x, y, z\}$

Using Equation 7.7, the negative log-likelihood is given as:

$$\mathcal{L}(\mathbf{E} = \{\xi_1(x), \xi_1(y), \xi_1(z) \dots, \xi_n(x), \xi_n(y), \xi_n(z)\} | \sigma) = \frac{3n}{2} \log(2\pi) + 3n \log \sigma + \frac{1}{2\sigma^2} \mathbf{U} \quad (7.8)$$

The maximum likelihood estimate $\hat{\sigma}$ can be determined by minimising Equation 7.8 and evaluating the corresponding σ . That is:

$$\frac{\partial \mathcal{L}}{\partial \sigma} = 0 \implies \hat{\sigma}^2 = \frac{\mathbf{U}}{3n} \quad (7.9)$$

\mathbf{U} involves statistics that do not take into account the data explicitly, and are sufficient to estimate σ (or RMSD). Therefore, the set of *sufficient statistics* for the least-squares superposition problem can be defined as:

$$\Psi = \left\{ \sum_{i=1}^n A_m, \sum_{i=1}^n A_p, \sum_{i=1}^n A_m B_m, \sum_{i=1}^n A_m B_p, \sum_{i=1}^n A_p B_p \right\} \quad (7.10)$$

where A and B take the values $\{x, y, z\}$, $A_m = \vec{v}'_i(A) - \vec{u}'_i(A)$ is the component-wise difference (similarly B_m), and $A_p = \vec{v}'_i(A) + \vec{u}'_i(A)$ is the component-wise sum (similarly B_p). Altogether, the set Ψ consists of 24 distinct statistics.

In addition, using the same notation, the statistics required to compute the centroid are of the form $\sum_{i=1}^n \vec{u}'_i(A)$ and $\sum_{i=1}^n \vec{v}'_i(A)$, and these are equivalent to $\sum_{\forall A} A_m$ and $\sum_{\forall A} A_p$.

7.2.4 Updating Sufficient Statistics

Addition Operation on Vector Sets Using Sufficient Statistics

Consider two pairs of corresponding vector sets: $\mathcal{Q} \leftrightarrow \mathcal{R}$ containing n_1 correspondences and $\mathcal{S} \leftrightarrow \mathcal{T}$ containing n_2 correspondences. Let \mathcal{U} be defined as a combination of vectors \mathcal{Q} and \mathcal{S} and similarly \mathcal{V} as a combination of \mathcal{R} and \mathcal{T} . Let Ψ_1 denote the sufficient statistics of superposing the first pair and Ψ_2 denote the same for the second pair. Define these as:

$$\Psi_1 = \left\{ \sum_{i=1}^{n_1} C_m, \sum_{i=1}^{n_1} C_p, \sum_{i=1}^{n_1} C_m D_m, \sum_{i=1}^{n_1} C_m D_p, \sum_{i=1}^{n_1} C_p D_p \right\} \quad (7.11)$$

$$\Psi_2 = \left\{ \sum_{i=1}^{n_2} E_m, \sum_{i=1}^{n_2} E_p, \sum_{i=1}^{n_2} E_m F_m, \sum_{i=1}^{n_2} E_m F_p, \sum_{i=1}^{n_2} E_p F_p \right\} \quad (7.12)$$

Where C, D, E and F are all $\{x, y, z\}$ denoting the components of the corresponding vectors in the vector sets under consideration. Consistent with the previous notation (see Equation 7.10), C_p and C_m (similarly D_p and D_m) are the component-wise sums and differences between corresponding vectors in \mathcal{Q} and \mathcal{R} . The same definitions hold for E_m (and E_p) and F_m (and F_p), with respect to corresponding vectors in \mathcal{S} and \mathcal{T} .

The aim is to use Ψ_1 and Ψ_2 to compute a new set of sufficient statistics Ψ (defined in Equation 7.10) for the superposition of vector sets $\mathcal{U} = \mathcal{Q} + \mathcal{S}$ with $\mathcal{V} = \mathcal{R} + \mathcal{T}$. Below, the construction of the new sufficient statistics is derived.

The statistics involved in computing the new centroids of the sets \mathcal{U} and \mathcal{V} , $\sum_{i=1}^{n=n_1+n_2} \vec{u}(A)$ and $\sum_{i=1}^{n=n_1+n_2} \vec{v}(A)$, can be trivially updated using the statistics $\sum_{i=1}^{n_1} \vec{q}(C)$, $\sum_{i=1}^{n_1} \vec{r}(D)$, $\sum_{i=1}^{n_2} \vec{s}(E)$, and $\sum_{i=1}^{n_2} \vec{t}(F)$.

To compute the remaining statistics in Ψ , define vectors:

$$\begin{aligned} \vec{\alpha}_1 &= \mathbf{Centroid}(\mathcal{U}) - \mathbf{Centroid}(\mathcal{Q}) & \vec{\beta}_1 &= \mathbf{Centroid}(\mathcal{V}) - \mathbf{Centroid}(\mathcal{R}) \\ \vec{\alpha}_2 &= \mathbf{Centroid}(\mathcal{U}) - \mathbf{Centroid}(\mathcal{S}) & \vec{\beta}_2 &= \mathbf{Centroid}(\mathcal{V}) - \mathbf{Centroid}(\mathcal{T}). \end{aligned}$$

These vectors define the corrections that are required to be made to the previous centroids to recover the updated ones.

Lemma 4. $\sum_{i=1}^{n=n_1+n_2} A_m = [\sum_{i=1}^{n_1} C_m + n_1 \Delta_m^C] + [\sum_{i=1}^{n_2} E_m + n_2 \Delta_m^E]$, where $\Delta_m^C = \vec{\beta}_1(C) - \vec{\alpha}_1(C)$ and $\Delta_m^E = \vec{\beta}_2(E) - \vec{\alpha}_2(E)$ and $A = C = E \in \{x, y, z\}$

Proof.

$$\begin{aligned}
\sum_{i=1}^{n=n_1+n_2} A_m &= \left[\sum_{i=1}^{n_1} \left[(\vec{r}'(C) + \vec{\beta}_1(C)) - (\vec{q}'(C) + \vec{\alpha}_1(C)) \right] \right] \\
&\quad + \left[\sum_{i=1}^{n_2} \left[(\vec{t}'(E) + \vec{\beta}_2(E)) - (\vec{s}'(E) + \vec{\alpha}_2(E)) \right] \right] \\
&= \left[\sum_{i=1}^{n_1} (\vec{r}'(C) - \vec{q}'(C)) + (\vec{\beta}_1(C) - \vec{\alpha}_1(C)) \right] \\
&\quad + \left[\sum_{i=1}^{n_2} (\vec{t}'(E) - \vec{s}'(E)) + (\vec{\beta}_2(E) - \vec{\alpha}_2(E)) \right] \\
&= \left[\sum_{i=1}^{n_1} C_m + \sum_{i=1}^{n_1} \Delta_m^C \right] + \left[\sum_{i=1}^{n_2} E_m + \sum_{i=1}^{n_2} \Delta_m^E \right] \\
&= \left[\sum_{i=1}^{n_1} C_m + n_1 \Delta_m^C \right] + \left[\sum_{i=1}^{n_2} E_m + n_2 \Delta_m^E \right]
\end{aligned}$$

□

Corollary 1. $\sum_{i=1}^n A_p = [\sum_{i=1}^{n_1} C_p + n_1 \Delta_p^C] + [\sum_{i=1}^{n_2} E_p + n_2 \Delta_p^E]$

Lemma 5.

$$\begin{aligned}
\sum_{i=1}^{n=n_1+n_2} A_m B_m &= \left[\sum_{i=1}^{n_1} C_m D_m + \Delta_m^C \sum_{i=1}^{n_1} D_m + \Delta_m^D \sum_{i=1}^{n_1} C_m + n_1 \Delta_m^C \Delta_m^D \right] \\
&\quad + \left[\sum_{i=1}^{n_2} E_m F_m + \Delta_m^E \sum_{i=1}^{n_2} F_m + \Delta_m^F \sum_{i=1}^{n_2} E_m + n_2 \Delta_m^E \Delta_m^F \right]
\end{aligned}$$

where $\Delta_m^C = \vec{\beta}_1(C) - \vec{\alpha}_1(C)$, $\Delta_m^D = \vec{\beta}_1(D) - \vec{\alpha}_1(D)$, $\Delta_m^E = \vec{\beta}_2(E) - \vec{\alpha}_2(E)$, and $\Delta_m^F = \vec{\beta}_2(F) - \vec{\alpha}_2(F)$ $A = C = E \in \{x, y, z\}$ and $B = D = F \in \{x, y, z\}$

Proof.

$$\begin{aligned}
\text{Updated} \left(\sum_{i=1}^{n_1} C_m D_m \right) &= \sum_{i=1}^{n_1} \left[(\vec{r}'(C) + \vec{\beta}_1(C)) - (\vec{q}'(C) + \vec{\alpha}_1(C)) \right] \\
&\quad \left[(\vec{r}'(D) + \vec{\beta}_1(D)) - (\vec{q}'(D) + \vec{\alpha}_1(D)) \right]
\end{aligned}$$

$$\begin{aligned}
&= \sum_{i=1}^{n_1} [(\vec{r}'(C)\vec{r}'(D) - \vec{q}'(C)\vec{q}'(D) - \vec{r}'(C)\vec{q}'(D) + \vec{q}'(C)\vec{r}'(D))] \\
&+ \sum_{i=1}^{n_1} [(\vec{\beta}_1(C)\vec{r}'(D) - \vec{\alpha}_1(C)\vec{r}'(D) - \vec{\beta}_1(C)\vec{q}'(D) + \vec{\alpha}_1(C)\vec{q}'(D))] \\
&+ \sum_{i=1}^{n_1} [(\vec{r}'(C)\vec{\beta}_1(D) - \vec{r}'(C)\vec{\alpha}_1(D) - \vec{q}'(C)\vec{\beta}_1(D) + \vec{q}'(C)\vec{\alpha}_1(D))] \\
&+ \sum_{i=1}^{n_1} [(\vec{\beta}_1(C)\vec{\beta}_1(D) - \vec{\alpha}_1(C)\vec{\beta}_1(D) - \vec{\beta}_1(C)\vec{\alpha}_1(D) + \vec{\alpha}_1(C)\vec{\alpha}_1(D))] \\
&= \sum_{i=1}^{n_1} C_m D_m + \sum_{i=1}^{n_1} \Delta_m^C D_m + \sum_{i=1}^{n_1} \Delta_m^D C_m + \sum_{i=1}^{n_1} \Delta_m^C \Delta_m^D \\
&= \sum_{i=1}^{n_1} C_m D_m + \Delta_m^C \sum_{i=1}^{n_1} D_m + \Delta_m^D \sum_{i=1}^{n_1} C_m + n_1 \Delta_m^C \Delta_m^D
\end{aligned}$$

Similarly:

$$\text{Updated} \left(\sum_{i=1}^{n_2} E_m F_m \right) = \sum_{i=1}^{n_2} E_m F_m + \Delta_m^E \sum_{i=1}^{n_2} F_m + \Delta_m^F \sum_{i=1}^{n_2} E_m + n_2 \Delta_m^E \Delta_m^F$$

Adding the two updated statistics, the lemma follows. □

Corollary 2.

$$\begin{aligned}
\sum_{i=1}^{n=n_1+n_2} A_m^2 &= \sum_{i=1}^{n=n_1+n_2} A_m A_m = \left[\sum_{i=1}^{n_1} C_m C_m + 2\Delta_m^C \sum_{i=1}^{n_1} C_m + n_1 (\Delta_m^C)^2 \right] \\
&+ \left[\sum_{i=1}^{n_2} E_m E_m + 2\Delta_m^E \sum_{i=1}^{n_2} E_m + n_2 (\Delta_m^E)^2 \right]
\end{aligned}$$

Corollary 3.

$$\begin{aligned}
\sum_{i=1}^{n=n_1+n_2} A_p B_p &= \left[\sum_{i=1}^{n_1} C_p D_p + \Delta_p^C \sum_{i=1}^{n_1} D_p + \Delta_p^D \sum_{i=1}^{n_1} C_p + n_1 \Delta_p^C \Delta_p^D \right] \\
&+ \left[\sum_{i=1}^{n_2} E_p F_p + \Delta_p^E \sum_{i=1}^{n_2} F_p + \Delta_p^F \sum_{i=1}^{n_2} E_p + n_2 \Delta_p^E \Delta_p^F \right]
\end{aligned}$$

Corollary 4.

$$\begin{aligned}
\sum_{i=1}^{n=n_1+n_2} A_p^2 &= \sum_{i=1}^{n=n_1+n_2} A_p A_p = \left[\sum_{i=1}^{n_1} C_p C_p + 2\Delta_p^C \sum_{i=1}^{n_1} C_p + n_1 (\Delta_p^C)^2 \right] \\
&+ \left[\sum_{i=1}^{n_2} E_p E_p + 2\Delta_p^E \sum_{i=1}^{n_2} E_p + n_2 (\Delta_p^E)^2 \right]
\end{aligned}$$

Lemma 6.

$$\begin{aligned} \sum_{i=1}^{n=n_1+n_2} A_m B_p &= \left[\sum_{i=1}^{n_1} C_m D_p + \Delta_m^C \sum_{i=1}^{n_1} D_p + \Delta_p^D \sum_{i=1}^{n_1} C_m + n_1 \Delta_m^C \Delta_p^D \right] \\ &+ \left[\sum_{i=1}^{n_2} E_m F_p + \Delta_m^E \sum_{i=1}^{n_2} F_p + \Delta_p^F \sum_{i=1}^{n_2} E_m + n_2 \Delta_m^E \Delta_p^F \right] \end{aligned}$$

where $\Delta_m^C = \vec{\beta}_1(C) - \vec{\alpha}_1(C)$, $\Delta_m^D = \vec{\beta}_1(D) - \vec{\alpha}_1(D)$, $\Delta_m^E = \vec{\beta}_2(E) - \vec{\alpha}_2(E)$, and $\Delta_m^F = \vec{\beta}_2(F) - \vec{\alpha}_2(F)$ $A = C = E \in \{x, y, z\}$ and $B = D = F \in \{x, y, z\}$

Proof.

$$\begin{aligned} \text{Updated} \left(\sum_{i=1}^{n_1} C_m D_p \right) &= \sum_{i=1}^{n_1} \left[(\vec{r}'(C) + \vec{\beta}_1(C)) - (\vec{q}'(C) + \vec{\alpha}_1(C)) \right] \\ &\quad \left[(\vec{r}'(D) + \vec{\beta}_1(D)) + (\vec{q}'(D) + \vec{\alpha}_1(D)) \right] \\ &= \sum_{i=1}^{n_1} [(\vec{r}'(C)\vec{r}'(D) - \vec{q}'(C)\vec{q}'(D) + \vec{r}'(C)\vec{q}'(D) - \vec{q}'(C)\vec{r}'(D))] \\ &+ \sum_{i=1}^{n_1} [(\vec{\beta}_1(C)\vec{r}'(D) - \vec{\alpha}_1(C)\vec{r}'(D) + \vec{\beta}_1(C)\vec{q}'(D) - \vec{\alpha}_1(C)\vec{q}'(D))] \\ &+ \sum_{i=1}^{n_1} [(\vec{r}'(C)\vec{\beta}_1(D) - \vec{r}'(C)\vec{\alpha}_1(D) + \vec{q}'(C)\vec{\beta}_1(D) - \vec{q}'(C)\vec{\alpha}_1(D))] \\ &+ \sum_{i=1}^{n_1} [(\vec{\beta}_1(C)\vec{\beta}_1(D) - \vec{\alpha}_1(C)\vec{\beta}_1(D) + \vec{\beta}_1(C)\vec{\alpha}_1(D) - \vec{\alpha}_1(C)\vec{\alpha}_1(D))] \\ &= \sum_{i=1}^{n_1} C_m D_p + \sum_{i=1}^{n_1} \Delta_m^C D_p + \sum_{i=1}^{n_1} \Delta_m^D C_p + \sum_{i=1}^{n_1} \Delta_m^C \Delta_p^D \\ &= \sum_{i=1}^{n_1} C_m D_p + \Delta_p^C \sum_{i=1}^{n_1} D_m + \Delta_m^D \sum_{i=1}^{n_1} C_p + n_1 \Delta_m^C \Delta_p^D \end{aligned}$$

Similarly:

$$\text{Updated} \left(\sum_{i=1}^{n_2} E_m F_p \right) = \sum_{i=1}^{n_2} E_m F_p + \Delta_m^E \sum_{i=1}^{n_2} F_p + \Delta_p^F \sum_{i=1}^{n_2} E_m + n_2 \Delta_m^E \Delta_p^F$$

Adding the two updated statistics, the lemma follows. \square

Deletion Operation of Vector Sets Using Sufficient Statistics

Consider the case where a superposition needs to be found under a deletion operation. That is, let $\mathcal{Q} \leftrightarrow \mathcal{R}$ and $\mathcal{S} \leftrightarrow \mathcal{T}$ denote two pairs of vector sets that are in correspondence. Let $\mathcal{S} \subset \mathcal{Q}$ and $\mathcal{T} \subset \mathcal{R}$. Under this assumption, let us define $\mathcal{U} = \mathcal{Q} - \mathcal{S}$ and $\mathcal{V} = \mathcal{R} - \mathcal{T}$.

Using the same notations as in the previous section, it is straightforward to see that the sufficient statistics Ψ of the superposition of \mathcal{U} with \mathcal{V} can be derived from the sufficient statistics Ψ_1 (of $\mathcal{Q} \leftrightarrow \mathcal{R}$) and Ψ_2 (of $\mathcal{S} \leftrightarrow \mathcal{T}$). The update rules defining the deletion operation are similar to the ones described above.

7.2.5 Computing the RMSD from Updated Sufficient Statistics

It is easy to see that Kearsley's 4×4 quaternion matrix \mathbf{Q} given in Equation 7.1 can be constructed using the updated sufficient statistics Ψ derived from Ψ_1 and Ψ_2 . The matrix \mathbf{Q} contains 10 distinct elements (given that \mathbf{Q} is square symmetric) which can be computed in constant time.

In practice, \mathbf{Q} is diagonalised using the Jacobi's iterative rotation approach, which annihilates an off-diagonal element with each rotation. This approach has a fast convergence, and requires no additional optimisation. However, in many cases the updated superposition shows only a marginal change from the previous one. For example, if a current superposition were to be extended by one pair of residues, the resultant new transformation will often, in practice, be very close to the previously computed one. This allows the diagonalisation to build on the previous solution.

Let \mathbf{Q} denote the Kearsley's 4×4 matrix corresponding to the superposition of corresponding vector sets \mathcal{U} and \mathcal{V} . From the eigen decomposition theorem, $\mathbf{Q} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1}$, where \mathbf{S} is the matrix of eigenvectors and $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. Also note that \mathbf{Q} is a positive semidefinite matrix with the property $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T$. This implies that all the eigenvectors are orthogonal to each other. This further simplifies the decomposition to $\mathbf{Q} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T$. Also, since \mathbf{S} is an orthogonal matrix, $\mathbf{Q} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T \implies \mathbf{\Lambda} = \mathbf{S}^T\mathbf{Q}\mathbf{S}$.

Now, assume that the corresponding vector sets are augmented from \mathcal{U} and \mathcal{V} to \mathcal{U}' and \mathcal{V}' , resulting in an updated Kearsley's matrix \mathbf{Q}' . The aim is to diagonalise this matrix into $\mathbf{S}'\mathbf{\Lambda}'\mathbf{S}'^T$. Instead of starting the Jacobi's iterative process from scratch, use the previously computed eigenvectors (before the vector sets were augmented), \mathbf{S} , and compute $\tilde{\mathbf{\Lambda}}$ as $\mathbf{S}^T\mathbf{Q}'\mathbf{S}$. Notice that if the augmentation does not include drastic changes, then $\tilde{\mathbf{\Lambda}}$ is nearly diagonal (that is, $\tilde{\mathbf{\Lambda}} \approx \mathbf{\Lambda}'$), thus requiring very few iterations to fully diagonalise it. This provides a further optimisation to the diagonalisation step under update operations on vector sets.

7.2.6 Results and Discussion

Consistency of Superpositions Using Sufficient Statistics

To validate the consistency of superpositions generated using sufficient statistics (under both addition and deletion operations discussed in Section 7.2.4) the following benchmarking is undertaken. 8992 ASTRAL SCOP (Murzin et al., 1995; Chandonia et al., 2004) domains were used as the source structures from which superposable fragments were randomly sampled. The general procedure of sampling is as follows. From the list of source structures, randomly choose any structure. Within this structure choose 2 random fragments of lengths l_1 and l_2 , where each fragment has between 10 and 40 residues. These chosen fragments form the sets \mathcal{Q} and \mathcal{S} . Yet another structure is again chosen randomly from the source list, and two fragments are randomly extracted from it with exactly the same length, l_1 and l_2 . These form the sets \mathcal{R} and \mathcal{T} . Assuming one-to-one correspondence between $\mathcal{Q} \leftrightarrow \mathcal{R}$ the sufficient statistics Ψ_1 of their least-squares superposition are computed. Similarly, the sufficient statistics Ψ_2 are computed for the least-squares superposition between $\mathcal{S} \leftrightarrow \mathcal{T}$. Define $\mathcal{U} = \mathcal{Q} + \mathcal{S}$ and $\mathcal{V} = \mathcal{R} + \mathcal{T}$. The following is computed by iterating this random sampling 100 million times:

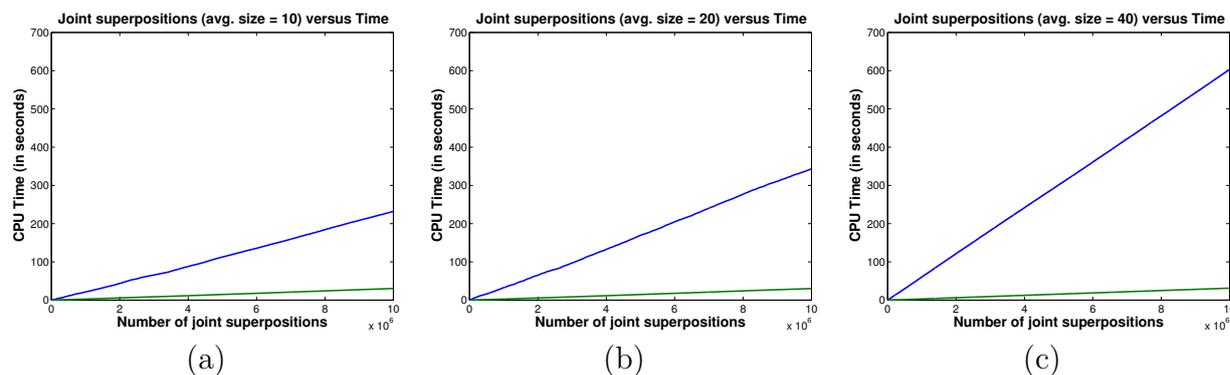


Figure 7.6: The CPU times (in seconds) performing joint superpositions from scratch (Blue line) compared against the same using sufficient statistics (Green line) over 10 million random fragment data sets derived from ASTRAL SCOP domains. The three plots vary in the average superposition size as indicated in the title.

1. the RMSD of superposition of $\mathcal{U} \leftrightarrow \mathcal{V}$ from scratch (using the raw coordinates in the vector sets). Denote this RMSD as ρ_1
2. the RMSD of superposition of $\mathcal{U} \leftrightarrow \mathcal{V}$, but using the sufficient statistics Ψ_1 and Ψ_2 of superpositions of constituent vector sets $\mathcal{Q} \leftrightarrow \mathcal{R}$, and $\mathcal{S} \leftrightarrow \mathcal{T}$. Denote this RMSD as ρ_2 .

We measure the difference between the two RMSD values, $\Delta\rho = \rho_1 - \rho_2$. Over the 100 million samples, the mean and standard deviation of $\Delta\rho$ was found to be zero to a very high precision ($< 10^{-17}$).

The same experiment is repeated to validate superpositions under deletion operation using sufficient statistics, where, in each iteration, the superposition of vector sets $\mathcal{S} \leftrightarrow \mathcal{T}$, is computed with $\mathcal{S} = \mathcal{U} \setminus \mathcal{Q}$ and $\mathcal{T} = \mathcal{V} \setminus \mathcal{R}$. This experiment again confirms the same consistency as observed in the test on addition operation.

Measuring the Performance Gain Using Sufficient Statistics for Superpositions

In Section 7.2.4 it was demonstrated that superpositions under addition and deletion can be updated in constant time, building on the sufficient statistics of the constituent sets. This was also validated empirically by the benchmarking above. Now the gain in performance is measured using this approach by comparing it with superpositions built from scratch.

Figures 7.6(a)-(c) show the runtime plots of three sets of randomly chosen 10 million joint superpositions carried out from scratch (Blue line) and compared against the same superpositions updated using sufficient statistics (Green line). Note that these three sets vary in the (average) size of the joint superpositions being carried out, as indicated in the plot titles.

Note also that, when the joint superpositions are carried out from scratch ignoring the sufficient statistics, the average size of the superpositions introduces a constant factor to the run time. This is expected as each superposition is linear in the size of the vector sets being superposed. Consequently, the slopes of those blue lines across the three plots in Figure 7.6 become steeper with the increase in the superposition size. On the other hand, when the joint superpositions are updated in constant time, the updates are independent of the superposition size. This is because, any update involves recomputing only a small (fixed) number of sufficient statistics. This is clearly reflected in the slopes of the green lines, which remain unchanged across the three plots in Figure 7.6.

Table 7.1: Time taken to perform exhaustive joint superpositions on a library of well-fitting fragment pairs between two structures from different families.

Protein Family	structural pair wwPDB IDs	Number of joint superpositions	Average size of superpositions	Time in seconds (from scratch)	Time in seconds (sufficient stats)
Serine Proteinases	3EST vs. 2PKA	18,486,240	14	419.6	56.0
Calmodulin-like	1NCX vs. 2SAS	67,820,481	18	1618.4	178.2
Serine Proteinases	3EST vs. 2SNV	71,025,321	16	1328.0	187.8
Globins	1HHOA vs. 1HHOB	74,890,441	20	1923.3	194.6

More formally, if $|J|$ is the number of joint superpositions and l is the (average) number of vectors being superposed, the first method (Blue line in Figure 7.6) grows as $O(l|J|)$. Since $l \ll |J|$ there is a linear trend (with a steeper gradient accounting for the multiplier l in the complexity term). In comparison, the results with sufficient statistics (Green line in Figure 7.6) grow only as $O(|J|)$, independently of the superposition size.

Using Sufficient Statistics of Superposition in the Setting of Structural Alignments

As discussed in the introduction, a common heuristic employed to compute a structural alignment between pairs of structures, involves collecting a library of well-fitting fragments. This library is refined by jointly superposing pairs from this library, and a final structural alignment is assembled from these results.

To test the potential performance gain by using sufficient statistics, the time taken to undertake an exhaustive joint superposition on libraries of well-fitting fragments corresponding to a small collection of structural pairs is measured. For each pair of structures chosen, the well-fitting fragments are identified as those that superpose *maximally* within an RMSD threshold of 2 Å. Where maximal means those fragment pairs that cannot be extended any further without violating the RMSD threshold.

Table 7.1 shows the run times of joint superpositions performed exhaustively on a small set of protein structural pairs. As it can be seen from the table, using sufficient statistics for superpositions results in up to an order of magnitude improvement in the run time to carry these superpositions exhaustively. Since performing this task *without* sufficient statistics creates a computational bottleneck, existing structural alignment programs attempt to drastically restrict the number of superpositions, often trading off structural alignment quality for speed. Note that the improvements gained from using sufficient statistics for superpositions will allow these restrictions to be generously relaxed without any effect on the current run times, but potentially improving the structural alignment quality.

All of the above experiments were carried out on a standard laptop with 2.2GHz Intel® CPU and 4GB RAM.

7.3 Conclusions

This chapter presents the two ancillary methods that support this thesis:

1. A new information measure for comparing any two top k lists. By exploring their compressibility, the method provides a statistically rigorous measure of variability between ranked lists. It provides an objective trade-off between criteria that measure the dissimilarity between lists, addressing pitfalls in the existing measures. As a future direction of research, this measure can be used to address the important *rank aggregation problem*:

What is the ‘consensus’ top k ranking that combines the top k results from multiple sources. Notably, this measure is employed in assessing the level of disagreement between structural alignment scoring functions on ranked lists alignments in Section 4.8.3.

2. Least-squares superpositions of vector sets are central to identify similarities and differences between spatial objects. A set of sufficient statistics was derived for the least-squares superposition problem under the least squares criterion. These statistics provide an efficient way to operate (via addition and deletion of vectors) on previously computed superpositions. The results in this chapter demonstrate a drastic improvement in the computational effort required to compute RMSD using sufficient statistics. This speed-up drives the efficiency of **MMLigner** in computing seed alignments in Section 6.3.2 and I -value in computing adaptive superpositions in Section 4.6.

Chapter 8

Conclusions and Future Directions

“Essentially, all models are wrong, but some are useful.”

— G. Box (Box and Draper, 1987)

This thesis fundamentally re-examines the underpinnings of the protein structural alignment problem. In doing so, it proposes a fundamental shift in the way protein structural alignment quality is measured and useful alignments are identified. This is built on the rigorous statistical foundation of the Minimum Message Length (MML) principle (Wallace and Freeman, 1987). This thesis establishes a statistically rigorous criterion, based on MML, for the distinction between the quality of competing alignments by combining Bayesian statistics with lossless data compression.

The development of this criterion culminated in the I -value measure of alignment quality. This measure computes a reliable and objective trade-off between **alignment complexity**, a hitherto poorly addressed characteristic of alignments, and **fidelity of fit** between two protein structures. This is in stark contrast to the ad hoc combination of coverage and RMSD terms frequently used by state-of-the-art structural alignment quality measures. The I -value equation encompasses an objective trade-off between alignment complexity and fidelity of fit as:

$$\underbrace{I(\mathcal{A}, \langle S, T \rangle)}_{I\text{-value}} = \underbrace{I(\mathcal{A})}_{\text{Alignment complexity}} + \underbrace{I_{\text{null}}(S)}_{\text{Constant over all alignments}} + \underbrace{I(T|S, \mathcal{A})}_{\text{Fidelity}}$$

Highlighted in the equation are message length terms representing alignment complexity and structural fidelity. The I -value measure is intuitive: a complex alignment may require a longer message length term, $I(\mathcal{A})$, but may allow the structures to fit very well, requiring a relatively shorter message length term, $I(T|S, \mathcal{A})$. This situation is reversed when the alignment is simpler, requiring a shorter message length term, $I(\mathcal{A})$, but may result in a poorer fit between the structures yielding a longer message length term, $I(T|S, \mathcal{A})$. Therefore, the best alignment is the one that minimises the combined two-part message length of I -value.

Supporting this information theoretic measure are the following useful properties:

1. I -value varies according to the posterior probability of the alignment. This provides a *formal trade-off* between the coverage, as measured by the complexity of the alignment hypothesis ($I(\mathcal{A})$) and the fidelity of the structures given the proposed alignment ($I(T|S, \mathcal{A})$). Unlike previous formulations of structural alignment scoring functions, these

terms are not *ad hoc* approximations of coverage and fidelity. Instead, they rigorously estimate the Shannon information content of the entire alignment and coordinate data based on lossless encoding and compression.

2. The difference between the I -values of any two competing alignments, \mathcal{A}_1 and \mathcal{A}_2 , gives the *log-odds posterior ratio*, $\log \left(\frac{\Pr(\mathcal{A}_2|\langle S, T \rangle)}{\Pr(\mathcal{A}_1|\langle S, T \rangle)} \right)$. This property makes the comparison and discrimination between competing alignments statistically robust. The best alignment is the one that minimises the I -value message length.
3. I -value provides a *natural null hypothesis test*. If the I -value of an alignment \mathcal{A} is worse (longer) than that of the null model encoding of the structural coordinates, then \mathcal{A} must be rejected. That is, reject \mathcal{A} if $I(\mathcal{A}, \langle S, T \rangle) \geq I_{\text{null}}(\langle S, T \rangle)$.

To the best of our knowledge, no other structural alignment scoring function naturally demonstrates these properties.

The accuracy of I -value largely depends on the details of the specific statistical models of encoding used for each term in the I -value equation (see Equation 5.1). The development of these encoding models requires a high degree of innovation. To our best efforts, a set of models and methods were developed to estimate the Shannon information content of the I -value message length terms in Chapter 4 and significantly improved upon in Chapter 5. Our encoding models build on the directional distributions of C_α coordinate data developed by Kasarapu and Allison (2015). All of the assumptions behind these models are open to scrutiny. However, it is conceivable that better statistical models could be developed beyond what is achieved in this thesis. Nevertheless, **the utility of any proposed improvements to the statistical models of encoding are best judged, in the MML paradigm, by the improvement to the lossless two-part message length** (for example, as demonstrated in Chapter 5).

This thesis attempts to rigorously evaluate the I -value measure of alignment quality. However, **no accepted gold standard for structural alignments are available to carry out an objective evaluation of alignment quality measures**. In the absence of a gold standard, various alignment scoring functions are evaluated for their ability to distinguish between protein domains at varying structural distances, as defined by the SCOP classification database (Murzin et al., 1995). **These evaluations clearly demonstrate the highly-consistent performance of the I -value measure in discriminating alignments across the SCOP hierarchy** (see Section 6.4, and Section 4.8). What is also clear from our evaluations is the extent disagreement among structural alignment scoring functions when ranking structural alignments, as previously observed by the host of comparative studies (Kolodny et al., 2005; Hasegawa and Holm, 2009; Sippl and Wiederstein, 2008; Slater et al., 2013; Ma and Wang, 2014).

Furthermore, this thesis develops a method of searching for meaningful structural alignments using I -value as the optimisation criterion. The resulting algorithm, **MMLigner**, presented in Chapter 6, is able to reliably find high-quality alignments for pairs of protein structures. Our evaluations demonstrate the effectiveness of **MMLigner** over other popular structural alignment methods, even simply judging from the combined RMSD and coverage profiles (see Section 6.4). Importantly, **MMLigner** is able to consistently identify alternative structural alignments of comparable quality. This is a challenging problem when aligning oligomeric proteins and protein complexes. Chapter 6 presents case studies where such alternative alignments exist for the same structural pair. It can be seen from these results that **MMLigner consistently outperforms other methods in identifying alternative structural alignments**.

This thesis also benefits from two ancillary contributions:

1. The derivation of sufficient statistics for the least-squares superposition problem. This results in a mathematically-guaranteed numerical method to rapidly compute joint superpositions of vector sets from superpositions of its constituent subsets. These sufficient statistics of superposition support addition and symmetric difference operations on previously superimposed vector subsets. This contribution supports the rapid and exhaustive generation of maximally superposable fragment pairs used to create seed alignments in (Phase 1 of) **MMLigner**. This technique is not limited to the domain of protein bioinformatics, but is broadly applicable to areas as diverse as robotics and signal processing.
2. A measure to compare top k elements of any two given rank orderings. The approach uses the same framework supporting I -value, based on the MML (Wallace and Boulton, 1968) criterion. This information-theoretic measure to compare two top k lists reconciles: the extent of their non-overlapping elements, the amount of disarray among overlapping elements and the measurement of their displacement in ranks (positions). This method is applied to the problem of comparing lists of alignments ranked by respective alignment scoring functions (see Chapter 4).

8.1 Extensions to the Research Presented in this Thesis

There are several clear avenues for further work and improvement to I -value and **MMLigner**.

8.1.1 Evaluating the Quality of Predicted Protein Structures

Today, protein structure is routinely determined using experimental methods such as X-ray crystallography (Kendrew et al., 1958; Muirhead and Perutz, 1963), Nuclear Magnetic Resonance (NMR; Wagner and Wüthrich (1978)) and Cryo-Electron Microscopy (Unwin and Henderson, 1975). However, these methods are relatively expensive compared to the determination of DNA sequence (França et al., 2002) that can be directly translated into protein sequence. The holy grail of molecular biology is the accurate computational prediction of protein structures from sequence.

Every two years the Critical Assessment of protein Structure Prediction (CASP; Moulton et al. (1995)) competition is held to test the state-of-the-art methods for protein structure prediction. Accurate protein structure alignments are critical to the assessment of entries into this competition. Using **MMLigner** and I -value to rank entries to this competition currently stands as partially completed work which we have been unable to include in this thesis.

8.1.2 Improvements to the Encoding Models

It would be useful to add sequence information to structural information for the models of encoding when constructing alignments of protein structures. As mentioned in Section 5.1, rigorous statistical models for protein sequence alignment exist and may be applied in I -value alongside structural data to improve alignment quality. The addition of sequence to the I -value computation is a potentially important improvement.

Second, while C_α atoms are the de-facto atom to use when representing amino acid coordinates, they do not provide any information about the orientation of the side chain. Therefore, it could be useful for extensions of the coordinate encoding models to take into account other main-chain atoms and the C_β atom in order to account for side-chain orientation.

Finally, the `MMLigner` algorithm could be extended to support multiple protein structural alignment. Alignments of more than two structures reveal more information to a biologist than does a pairwise alignment. Therefore, it is important to extend I -value and `MMLigner` to alignments of multiple protein structures.

8.1.3 Identifying Closely Competing Structural Alignments

The `MMLigner` algorithm performs a series of slight modifications in the final step and evaluates each modification using I -value. Recording the k best alignments under the I -value measure and presenting these as closely competing alternatives, gives biologists a powerful mechanism to evaluate points of uncertainty in the alignment produced together with a set of nearby alignments to choose from. Note that this procedure is *in addition* to the distinct alternate alignments produced by `MMLigner` in its current form.

8.1.4 Visualisations of Alignment Quality

The possibility of visualising the landscape of competing alignment hypotheses was touched upon in Section 6.3.1. This landscape allows users to determine whether the alignment obtained is unique or if there are many similar ones available providing a visual mechanism for the improvement mentioned above in Section 8.1.3. Figures 8.1-8.5 show a range of landscapes produced from simple, closely related proteins, to more distant relationships with multiple potential alternates. The height of each cell is equivalent to the I -value (lower is better) of the optimal alignment that “passes through” (an alignment considered as a path travelling from source to sink; see Section 6.3.1). Note that the visualisations make it easy to see the relative significance between alternatives (as the heights of valleys), the existence of areas of closely competing alignments (as valley floors), and the general landscape of alignment quality. The addition of interactivity to these landscapes (where a user is able to “click” on a particular cell to examine the alignment passing through it, and the superposition (as shown below) of the structures given by that alignment) makes for a compelling tool for the analysis of alignments and the structural relationships between proteins. This landscape visualisation is a natural successor to the dot-plot in the age of 3D graphics.

This visualisation was a by-product of a dynamic programming scoring matrix and thus `MMLigner` is unable to build such a visualisation in its current state. Extending `MMLigner` to be able to produce such a visualisation is however, very useful and an important direction this work should take.

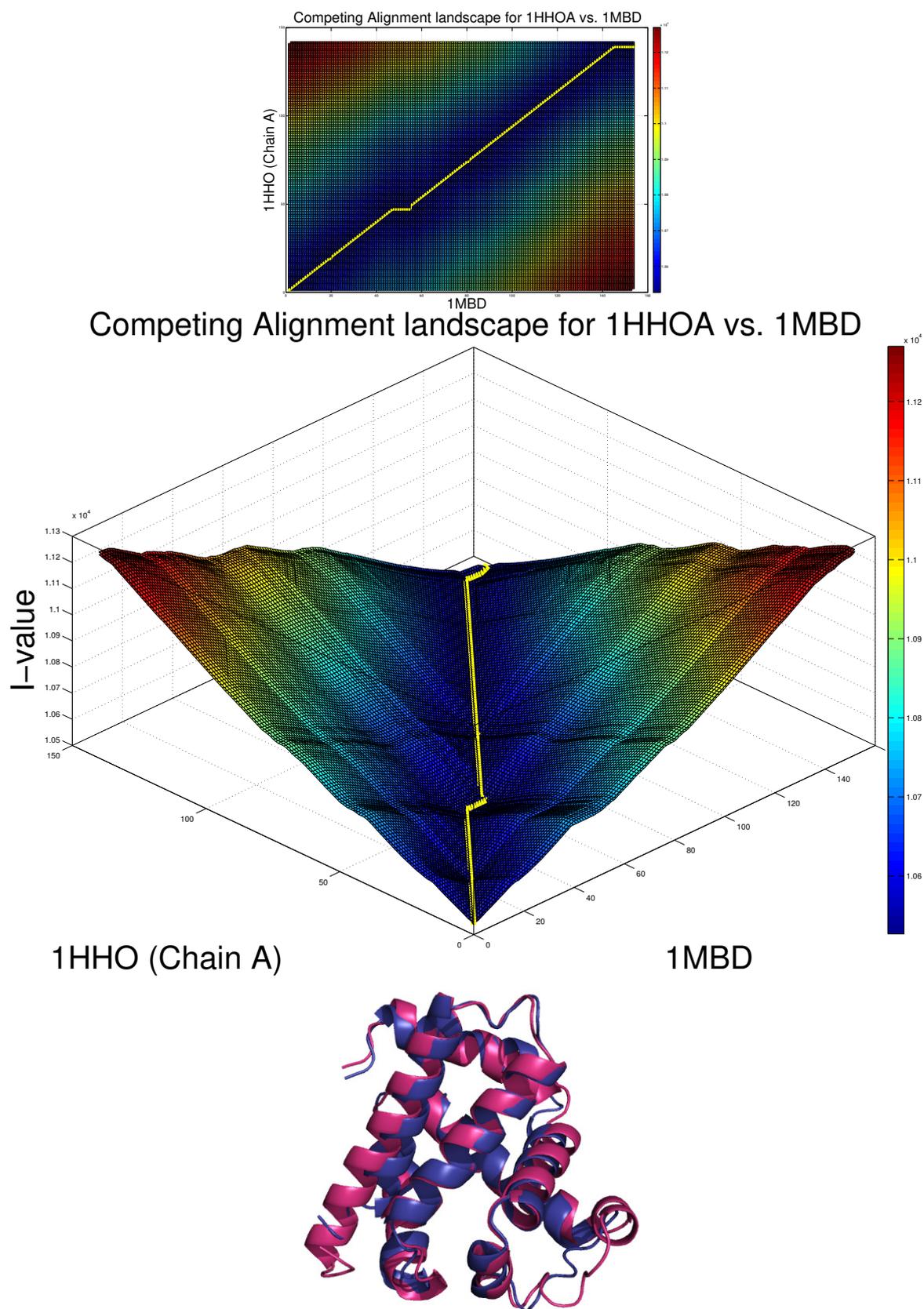


Figure 8.1: Landscape produced when aligning human h emoglobin (wwPDB 1HHO-A) against the closely related sperm whale myoglobin (wwPDB 1MBD). The bottom shows the superposition calculated from the optimal alignment highlighted by the yellow line through the “valley”.

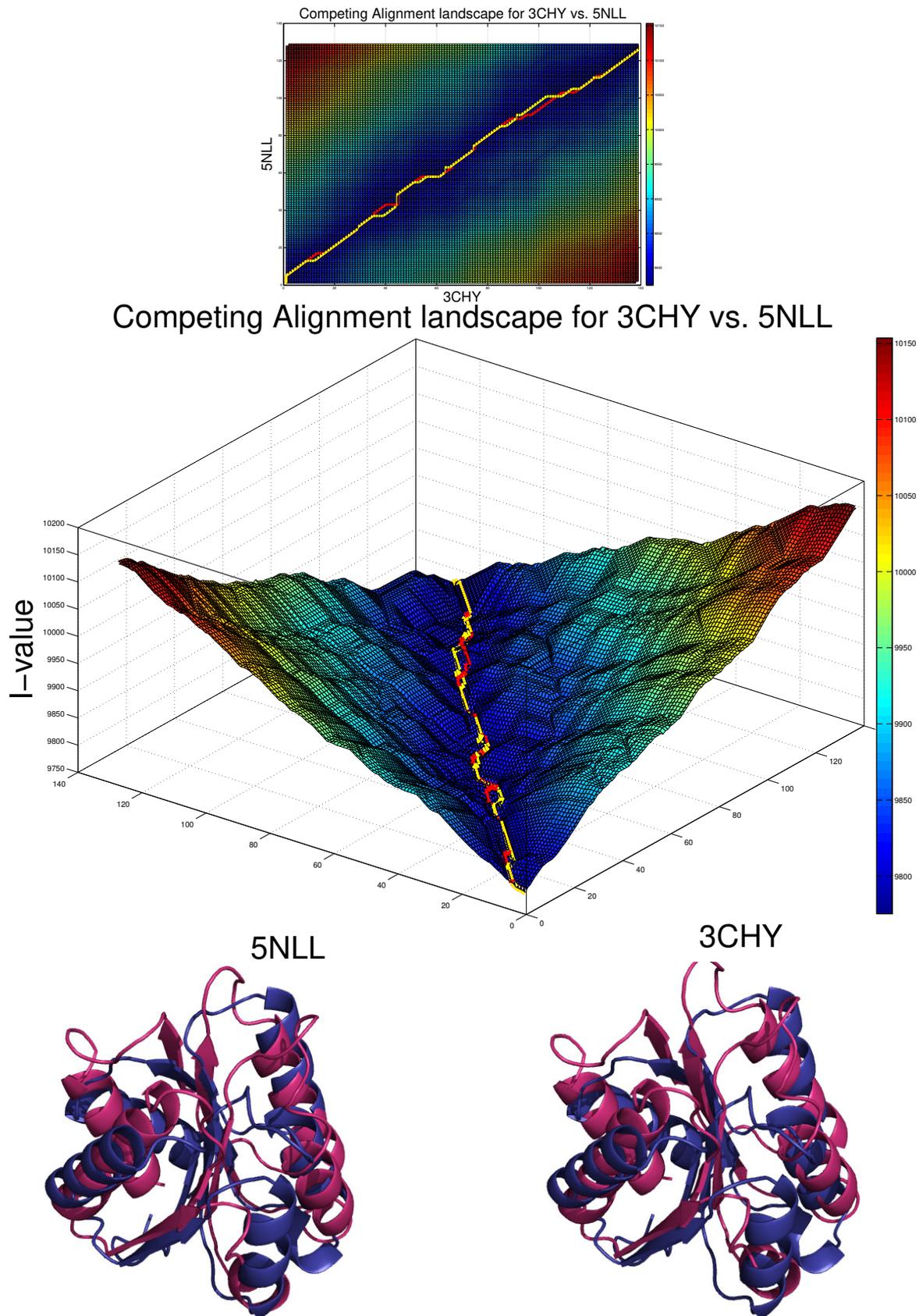


Figure 8.2: Landscape produced when aligning the *Escherichia Coli* chemotaxis protein CheY (wwPDB 3CHY) against *Clostridium beijerinckii* flavodoxin (wwPDB 5NLL). The bottom figures show the superpositions calculated from the optimal alignment, shown as a yellow line, and a closely competing alignment, shown as a red line.

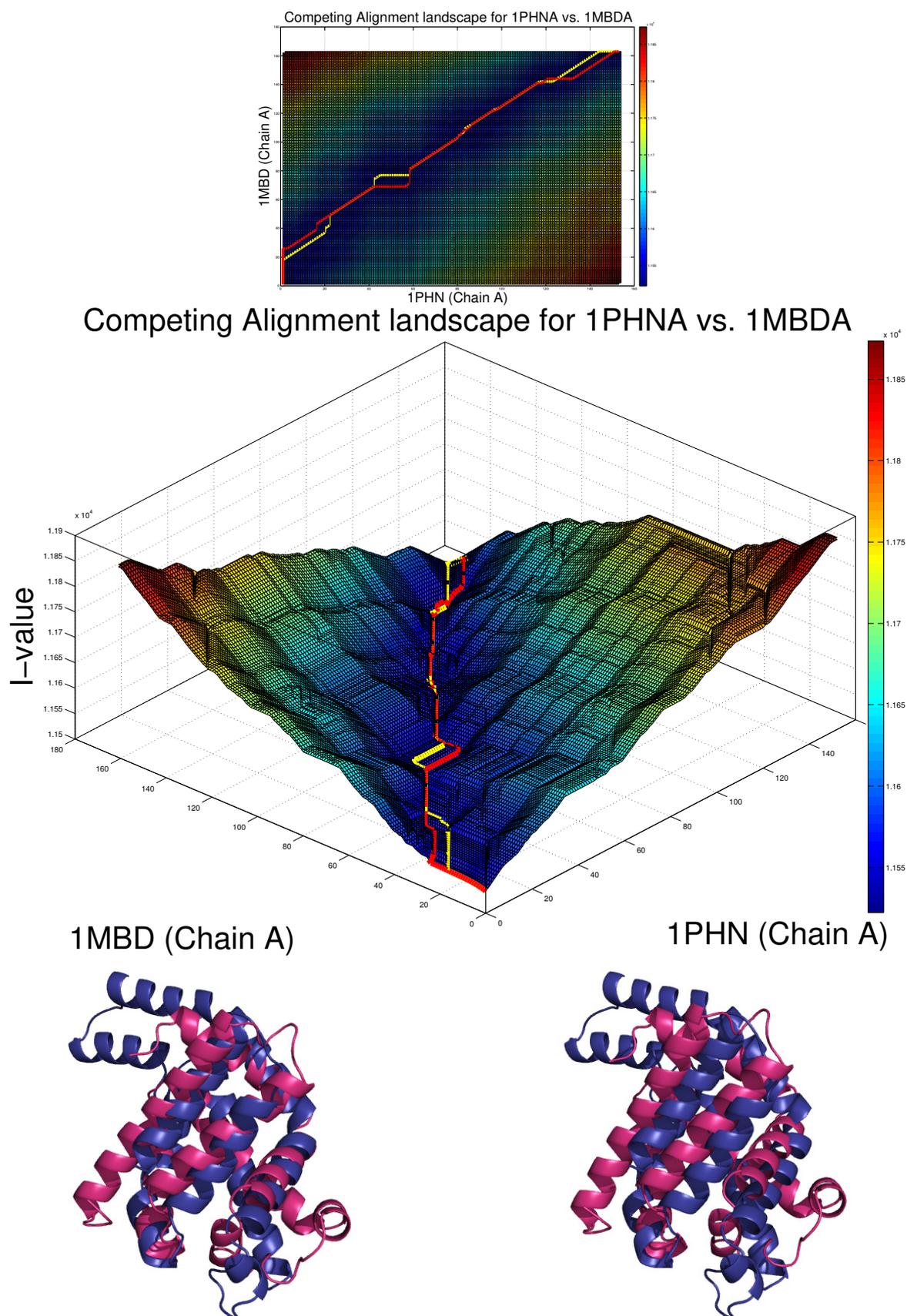


Figure 8.3: Landscape produced when aligning *Cyanidium caldarium* phycocyanin (wwPDB 1PHN) against sperm whale myoglobin (wwPDB 1MBD). The bottom figures show the superpositions calculated from the optimal alignment, shown as a yellow line, and a closely competing alignment, shown as a red line.

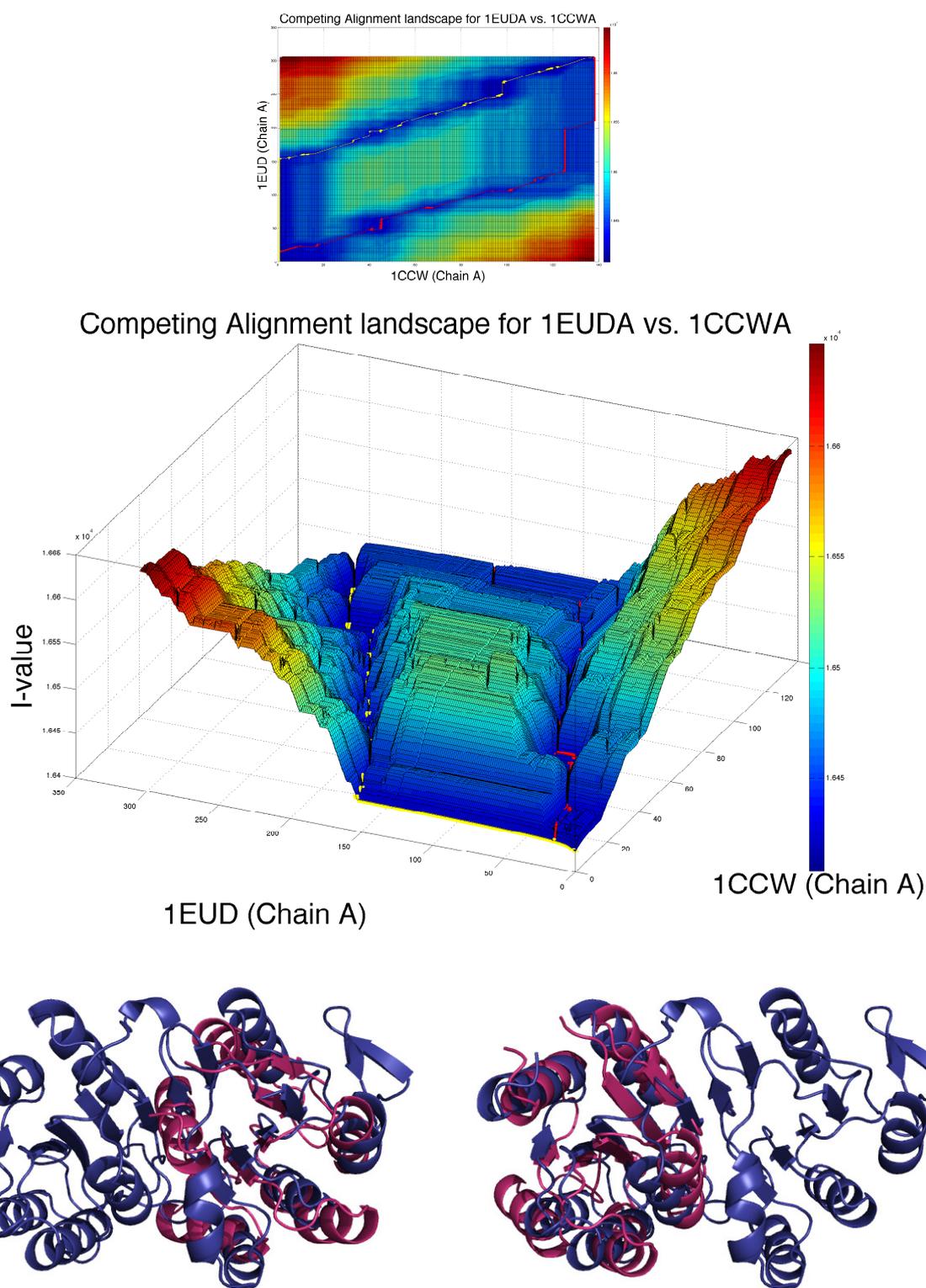


Figure 8.4: Landscape produced when aligning Succinyl-CoA synthetase from *Sus scrofa* (ww-PDB 1EUD-A) against glutamate mutase from *Clostridium cochlearium* (wwPDB 1CCW-A). The bottom figures show the superpositions calculated from the optimal alignment, shown as a yellow line, and a closely competing alignment, shown as a red line.

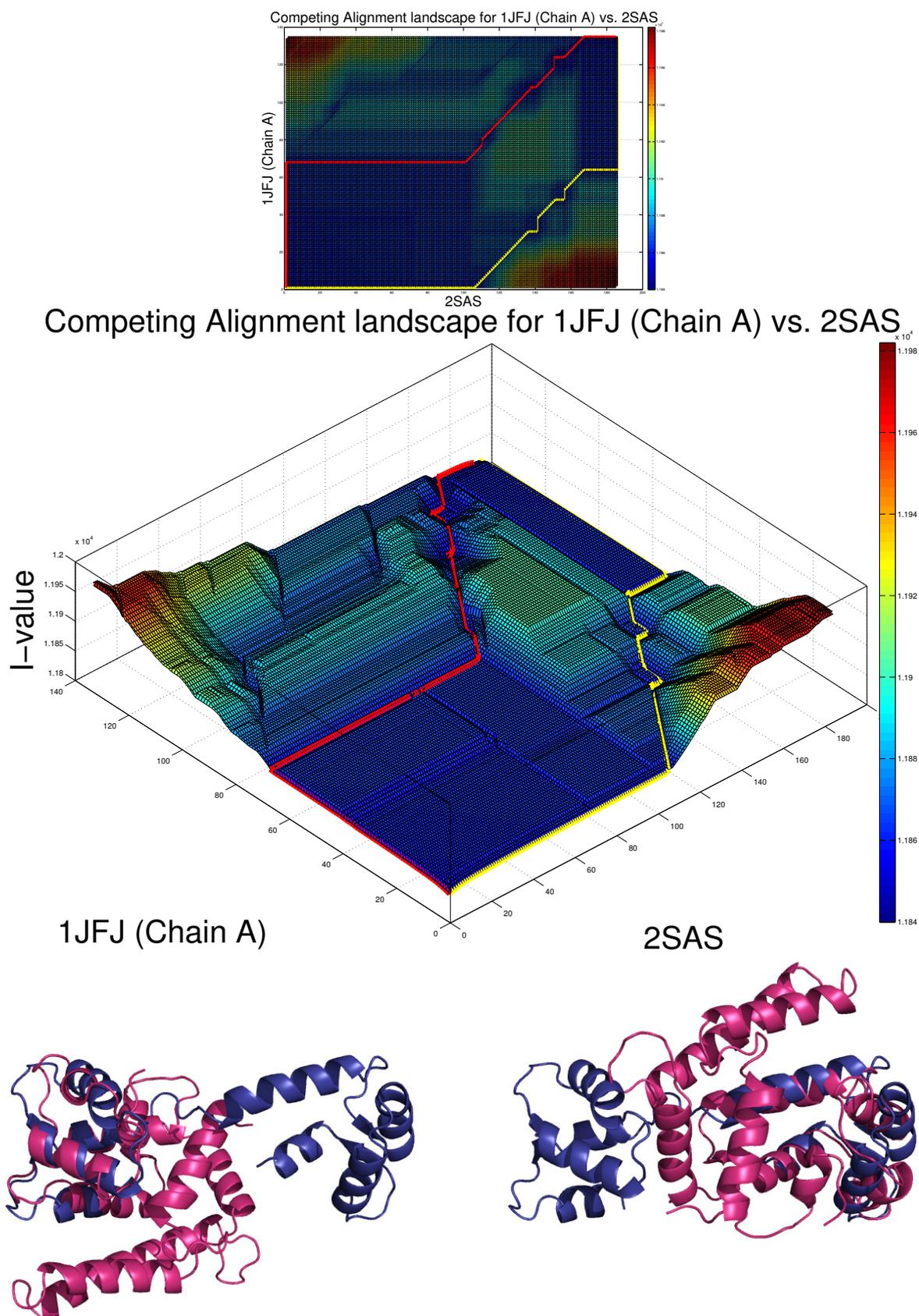


Figure 8.5: Landscape produced when aligning the pair of calcium-binding proteins from *Branchedostoma lanceolatum* (wwPDB 2SAS-A) and *Entamoeba histolytica* (wwPDB 1JFJ-A). The bottom figures show the superpositions calculated from the optimal alignment, shown as a yellow line, and a closely competing alignment, shown as a red line. This is an example of the dynamic programming algorithm (see Section 6.3.1 becoming distracted by close early matches. Though it recovers enough to notice the CC, NC, and CN domain alignments (see Figure 6.3).

References

- Abroi, A. and Gough, J. (2011). Are viruses a source of new protein folds for organisms?—virosphere structure space and evolution, *Bioessays* **33**(8): 626–635.
- Adrian, M., Dubochet, J., Lepault, J. and McDowell, A. W. (1984). Cryo-electron microscopy of viruses, *Nature* **308**(5954): 32–36.
- Akutsu, T. (1996). Protein structure alignment using dynamic programming and iterative improvement, *IEICE Transactions on Information and Systems*, Vol. E79-D, pp. 1629–1636.
- Allison, L., Wallace, C. and Yee, C. (1992). Finite-state models in the alignment of macromolecules, *Journal of Molecular Evolution* **35**(1): 77–89.
- Alt, H., Mehlhorn, K., Wagener, H. and Welzl, E. (1988). Congruence, similarity, and symmetries of geometric objects, *Discrete and Computational Geometry* **3**(3): 237–256.
- Altschul, S. F. (1991). Amino acid substitution matrices from an information theoretic perspective, *Journal of Molecular Biology* **219**(3): 555–565.
- Andersen, C. A. and Rost, B. (2009). Secondary structure assignment, in J. Gu and P. E. Bourne (eds), *Structural Bioinformatics*, John Wiley and Sons, chapter 19, pp. 459–484.
- Anfinsen, C. B. (1973). Principles that govern the folding of protein chains, *Science* **181**(4096): 223–230.
- Banerjee, A., Dhillon, I. S., Ghosh, J. and Sra, S. (2005). Clustering on the unit hypersphere using von Mises-Fisher distributions, *Journal of Machine Learning Research* **6**(Sep): 1345–1382.
- Bar-Ilan, J., Mat-Hassan, M. and Levene, M. (2006). Methods for comparing rankings of search engine results, *Computer Networks* **50**(10): 1448–1463.
- Bayes, T. and Price, R. (1763). An essay towards solving a problem in the doctrine of chance, *Philosophical Transactions of the Royal Society of London* **53**: 370–418.
- Bellman, R. (1952). On the theory of dynamic programming, *Proceedings of the National Academy of Sciences. USA* **38**(8): 716–719.
- Bellman, R. (1957). *Dynamic Programming*, Princeton University Press, Princeton, New Jersey.
- Berman, H., Henrick, K. and Nakamura, H. (2003). Announcing the worldwide Protein Data Bank, *Nature Structural & Molecular Biology* **10**(980).

- Berman, H. M., Battistuz, T., Bhat, T. N., Bluhm, W. F., Bourne, P. E., Burkhardt, K., Feng, Z., Gilliland, G. L., Iype, L., Jain, S., Fagan, P., Marvin, J., Padilla, D., Ravichandran, V., Schneider, B., Thanki, N., Weissig, H., Westbrook, J. D. and Zardecki, C. (2002). The Protein Data Bank, *Acta Crystallographica Section D* **58**(6 Part 1): 899–907.
- Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Weissig, T. B. H., Shindyalov, I. and Bourne, P. (2000). The protein data bank, *Nucleic Acids Research* **28**(1): 235–242.
- Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Jr., E. F. M., Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. and Tasumi, M. (1977). The Protein Data Bank: a computer-based archival file for macromolecular structures, *Journal of Molecular Biology* **112**(3): 535–542.
- Birzele, F., Gewehr, J. E., Csaba, G. and Zimmer, R. (2006). Vorolign–Fast structural alignment using voronoi contacts, *Bioinformatics* **23**(2): e205–e211.
- Björklund, Åsa. K., Ekman, D. and Elofsson, A. (2006). Expansion of protein domain repeats, *PLoS Computational Biology* **2**(8): 1–12.
- Bork, P. (1991). Shuffled domains in extracellular proteins, *FEBS Letters* **286**(1–2): 47–54.
- Boulton, D. M. and Wallace, C. S. (1973). An information measure for hierarchic classification, *The Computer Journal* **16**(3): 254–261.
- Bourne, P. E., Berman, H. M., McMahon, B., Watenpaugh, K. D., Westbrook, J. D. and Fitzgerald, P. M. (1997). Macromolecular crystallographic information file (mmCIF), *Macromolecular Crystallography Part B*, Vol. 277 of *Methods in Enzymology*, Academic Press, pp. 571–590.
- Box, G. E. P. and Draper, N. R. (1987). *Empirical Model-Building and Response Surfaces*, John Wiley & Sons, p. 424.
- Brown, P., Pullan, W., Yang, Y. and Zhou, Y. (2016). Fast and accurate non-sequential protein structure alignment using a new asymmetric linear sum assignment heuristic, *Bioinformatics* **32**(3): 370–377.
- Bu, Z. and Callaway, D. J. (2011). Proteins MOVE! Protein dynamics and long-range allostery in cell signaling, in R. Donev (ed.), *Protein Structure and Diseases*, Vol. 83 of *Advances in Protein Chemistry and Structural Biology*, Academic Press, chapter 5, pp. 163–221.
- Budinska, E., Kugler, K. and Lin, S. (2011). Package topklists for rank-based genomic data integration, in M. G. Schimek (ed.), *Proceedings of IASTED Computational Biology*.
- Budowski-Tal, I., Nov, Y. and Kolodny, R. (2010). FragBag, an accurate representation of protein structure, retrieves structural neighbors from the entire pdb quickly and accurately, *Proceedings of the National Academy of Sciences. USA* **107**(8): 3481–3486.
- Camproux, A. C., Tuffery, P., Chevrolat, J. P., Boisvieux, J. F. and Hazout, S. (1999). Hidden markov model approach for identifying the modular framework of the protein backbone, *Protein Engineering* **12**(12): 1063–1073.
- Cantor, D. G. and Zassenhaus, H. (1981). A new algorithm for factoring polynomials over finite fields, *Mathematics of Computation* **36**(154): 587–592.

- Casella, G. and Berger, R. L. (2001). *Statistical Inference*, 2 edn, Duxbury Press, Pacific Grove, California.
- Chaitin, G. J. (1966). On the length of programs for computing finite binary sequences, *Journal of the ACM (JACM)* **13**(4): 547–569.
- Chambers, J. M. (1983). *Graphical methods for data analysis*, Wadsworth International Group.
- Chandonia, J. M., Hon, G., Walker, N. S., Lo Conte, L., Koehl, P., Levitt, M. and Brenner, S. E. (2004). The ASTRAL compendium in 2004, *Nucleic Acids Research* **32**(suppl 1): D189–D192.
- Chothia, C. (1992). One thousand families for the molecular biologist, *Nature* **357**(6379): 543–544.
- Chothia, C. and Lesk, A. M. (1986). The relation between the divergence of sequence and structure in proteins, *The EMBO Journal* **5**(4): 823–826.
- Chu, C.-H., Tang, C. Y., Tang, C.-Y. and Pai, T.-W. (2008). Angle-distance image matching techniques for protein structure comparison, *Journal of Molecular Recognition* **21**(6): 442–452.
- Cohen, F. E. and Sternberg, M. J. E. (1980). On the prediction of protein structure: The significance of the root-mean-square deviation, *Journal of Molecular Biology* **138**(2): 321–333.
- Cohen, G. (1997). ALIGN: a program to superimpose protein coordinates, accounting for insertions and deletions, *Journal of Applied Crystallography* **30**(6): 1160–1161.
- Conway, J. H. and Sloane, N. J. A. (1984). On the voronoi regions of certain lattices., *SIAM Journal on Algebraic and Discrete Methods* **5**: 294–305.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2009). *Introduction to Algorithms*, 3 edn, MIT Press.
- Coutsias, E. A., Seok, C. and Dill, K. A. (2004). Using quaternions to calculate RMSD, *Journal of Computational Chemistry* **25**(15): 1849–1857.
- Crick, F. (1970). Central dogma of molecular biology, *Nature* **227**(5258): 561–563.
- Daimond, R. A. (1988). A note on the rotational superposition problem, *Acta Crystallographica Section A* **44**(2): 211–216.
- Darwin, C. R. (1859). *On the origin of species or the Preservation of Favored Races in the Struggle for Life*, Murray, London.
- de Oliveira, S. H. P., Shi, J. and Deane, C. M. (2015). Building a better fragment library for *De Novo* protein structure prediction, *PLoS ONE* **10**(4): 1–20.
- Delaunay, B. (1934). Sur la sphère vide. a la mémoire de georges voronoï, *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na* **6**: 793–800.
- Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society. Series B (Methodological)* **39**(1): 1–38.

- Dowe, D. L., Oliver, J. J., Baxter, R. A. and Wallace, C. S. (1996). Bayesian estimation of the von mises concentration parameter, *in* K. M. Hanson and R. N. Silver (eds), *Maximum Entropy and Bayesian Methods: Santa Fe, New Mexico, U.S.A., 1995 Proceedings of the Fifteenth International Workshop on Maximum Entropy and Bayesian Methods*, Springer Netherlands, Dordrecht, pp. 51–60.
- Edgar, R. C. (2004). Muscle: multiple sequence alignment with high accuracy and high throughput, *Nucleic Acids Research* **32**(5): 1792–1797.
- Edgar, R. C. (2010). Quality measures for protein alignment benchmarks, *Nucleic Acids Research* **38**(7): 2145–2153.
- Edwards, H. and Deane, C. M. (2015). Structural bridges through fold space, *PLoS Computational Biology* **11**(9): e1004466.
- Eidhammer, I., Jonassen, I. and Taylor, W. R. (2000). Structure comparison and structure patterns, *Journal of Computational Biology* **7**(5): 685–716.
- Eidhammer, I., Jonassen, I. and Taylor, W. R. (2004). *Protein Bioinformatics: An algorithmic approach to sequence and structure analysis*, J. Wiley & Sons.
- Einstein, A. (1933). On the method of theoretical physics, the Herbert Spencer Lecture.
- Elias, P. (1975). Universal codeword sets and representations of the integers, *IEEE Transactions on Information Theory* **21**(2): 194–203.
- Ellis, R. J. (2006). Molecular chaperones: assisting assembly in addition to folding, *Trends in Biochemical Sciences* **31**(7): 395–401.
- Emekli, U., Schneidman-Duhovny, D., Wolfson, H. J., Nussinov, R. and Haliloglu, T. (2008). HingeProt: automated prediction of hinges in protein structures, *Proteins* **70**(4): 1219–1227.
- Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D. and Vee, E. (2006). Comparing partial rankings, *SIAM Journal on Discrete Mathematics* **20**(3): 628–648.
- Fagin, R., Kumar, R. and Sivakumar, D. (2003). Comparing top k lists, *SIAM Journal on Discrete Mathematics* **17**(1): 134–160.
- Falicov, A. and Cohen, F. E. (1996). A surface of minimum area metric for the structural comparison of proteins, *Journal of Molecular Biology* **258**(5): 871–892.
- Farr, G. and Wallace, C. (2002). The complexity of strict minimum message length inference, *The Computer Journal* **45**(3): 285–292.
- Fischer, D., Barret, C., Bryson, K., Elofsson, A., Godzik, A., Jones, D., Karplus, K. J., Kelley, L. A., MacCallum, R. M., Pawowski, K., Rost, B., Rychlewski, L. and Sternberg, M. (1999). CAFASP-1: Critical assessment of fully automated structure prediction methods, *Proteins: Structure, Function, and Genetics* **37**(Suppl. 3): 209–217.
- Fisher, N. I., Lewis, T. and Embleton, B. J. J. (1987). *Statistical Analysis of Spherical Data*, Cambridge University Press.

- Fox, N. K., Brenner, S. E. and Chandonia, J. M. (2013). SCOPe: Structural classification of proteins – extended, integrating SCOP and ASTRAL data and classification of new structures, *Nucleic Acids Research* **42**(Database issue): D304–D309.
- França, L. T. C., Carrilho, E. and Kist, T. B. L. (2002). A review of DNA sequencing techniques, *Quarterly Reviews of Biophysics* **35**(2): 169–200.
- Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers i. introduction, *Australian Journal of Biological Sciences* **10**(4): 484–491.
- Frauenfelder, H., Sligar, S. G. and Wolynes, P. G. (1991). The energy landscapes and motions of proteins, *Science* **254**(5038): 1598–1603.
- Friedberg, I. and Godzik, A. (2005). Connecting the protein structure universe by using sparse recurring fragments, *Structure* **13**(8): 1213–1224.
- Friedberg, I., Harder, T., Kolodny, R., Sitbon, E., Li, Z. and Godzik, A. (2007). Using an alignment of fragment strings for comparing protein structures, *Bioinformatics* **23**(2): e219–e224.
- Fury, W., Batliwalla, F., Gregersen, P. K. and Li, W. (2006). Overlapping probabilities of top ranking gene lists, hypergeometric distribution, and stringency of gene selection criterion, *IEEE Conference on Engineering in Medicine and Biology Society*, pp. 5531–5534.
- Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*, W. H. Freeman.
- Gerstein, M., Anderson, B. F., Norris, G. E., Baker, E. N., Lesk, A. M. and Chothia, C. (1993). Domain closure in lactoferrin. Two hinges produce a see-saw motion between alternative close-packed interfaces, *Journal of Molecular Biology* **234**(2): 357–372.
- Gerstein, M. and Chothia, C. (1991). Analysis of protein loop closure: Two types of hinges produce one motion in lactate dehydrogenase, *Journal of Molecular Biology* **220**(1): 133–149.
- Gerstein, M., Lesk, A. M. and Chothia, C. (1994). Structural mechanisms for domain movements in proteins, *Biochemistry* **33**(22): 6739–6749.
- Gerstein, M. and Levitt, M. (1998). Comprehensive assessment of automatic structural alignment against a manual standard, the scop classification of proteins, *Protein Science* **7**(2): 445–456.
- Godzik, A., Jambon, M. and Friedberg, I. (2007). Computational protein function prediction: Are we making progress?, *Cellular and Molecular Life Sciences* **64**(19): 2505–2511.
- Golub, G. H. and van der Vorst, H. A. (2000). Eigenvalue computation in the 20th century, *Journal of Computational and Applied Mathematics* **123**(1–2): 35–65.
- Gotoh, O. (1982). An improved algorithm for matching biological sequences, *Journal of Molecular Biology* **162**(3): 705–708.
- Grindley, H. M., Artymiuk, P. J., Rice, D. W. and Willett, P. (1993). Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm, *Journal of Molecular Biology* **229**(3): 707–721.

- Grishin, N. V. and Phillips, M. A. (1994). The subunit interfaces of oligomeric enzymes are conserved to a similar extent to the overall protein sequences, *Protein Science* **3**(12): 2455–2458.
- Guerler, A. and Knapp, E.-W. (2008). Novel protein folds and their nonsequential structural analogs, *Protein Science* **17**(8): 1374–1382.
- Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, Cambridge University Press, chapter 11.
- Hájek, A. (2003). Interpretations of probability, in E. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*.
- Hamilton, W. R. (1844). On a new species of imaginary quantities, connected with the theory of quaternions, *Proceedings of the Royal Irish Academy* **2**: 424–434.
- Hamilton, W. R. and Hamilton, W. E. (1866). *Elements of quaternions*, Longmans, Green, & Company.
- Hartl, F. U. (1996). Molecular chaperones in cellular protein folding, *Nature* **381**(6583): 571–580.
- Hasegawa, H. and Holm, L. (2009). Advances and pitfalls of protein structural alignment, *Current Opinion in Structural Biology* **19**: 341–348.
- Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications, *Biometrika* **57**(1): 97–109.
- Hayward, S. (1999). Structural principles governing domain motions in proteins, *Proteins: Structure, Function, and Bioinformatics* **36**(4): 425–435.
- Hayward, S. and Berendsen, H. J. C. (1998). Systematic analysis of domain motions in proteins from conformational change; new results on citrate synthase and T4 lysozyme, *Proteins: Structure, Function, and Genetics* **30**(2): 144–154.
- Hogg, R. V. and Craig, A. (1994). *Introduction to mathematical statistics*, Prentice Hall.
- Holm, L., Ouzounis, C., Sander, C., Tuparev, G. and Vriend, G. (1992). A database of protein structure families with common folding motifs, *Protein Science* **1**(12): 1691–1698.
- Holm, L. and Sander, C. (1993). Protein structure comparison by alignment of distance matrices, *Journal of Molecular Biology* **233**(1): 123–138.
- Holm, L. and Sander, C. (1998). Dictionary of recurrent domains in protein structures, *Proteins* **33**(1): 88–96.
- Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes, *Proceedings of the IRE*, Vol. 40, pp. 1098–1101.
- Illergård, K., Ardell, D. H. and Elofsson, A. (2009). Structure is three to ten times more conserved than sequence – a study of structural response in protein cores, *Proteins: Structure, Function, and Bioinformatics* **77**(3): 499–508.

- Ilyin, V. A., Abyzov, A. and M. Leslin, C. (2004). Structural alignment of proteins by a novel TOPOFIT method, as a superimposition of common volumes at a topomax point, *Protein Science* **13**(7): 1865–1874.
- Irving, J. A., Whisstock, J. C. and Lesk, A. M. (2001). Protein structural alignments and functional genomics, *Proteins: Structure, Function, and Bioinformatics* **42**: 378–382.
- Jacobi, C. G. J. (1846). Über ein leichtes Verfahren, die in der Theorie der Säkularstörungen vorkommenden Gleichungen numerisch aufzulösen, *Journal für die Reine und Angewandte Mathematik* **30**: 51–95.
- Joseph, A. P., Agarwal, G., Mahajan, S., Gelly, J.-C., Swapna, L. S., Offmann, B., Cadet, F., Bornot, A., Tyagi, M., Valadié, H., Schneider, B., Etchebest, C., Srinivasan, N. and Brevern, A. G. D. (2010). A short survey on protein blocks, *Biophysical Reviews* **2**(3): 137–147.
- Jurman, G., Riccadonna, S., Visintainer, R. and Furlanello, C. (2009). Canberra distance on ranked lists, *Proceedings, Advances in Ranking–NIPS 09 Workshop*, pp. 22–27.
- Jurman, G., Riccadonna, S., Visintainer, R. and Furlanello, C. (2012). Algebraic comparison of partial lists in bioinformatics, *PloS One* **7**(5): e36540.
- Kabsch, W. (1976). A solution for the best rotation to relate two sets of vectors, *Acta Crystallographica Section A* **32**(5): 922–923.
- Kabsch, W. (1978). A discussion of the solution for the best rotation to relate two sets of vectors, *Acta Crystallographica Section A* **34**(5): 827–828.
- Kabsch, W. and Sander, C. (1983). Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features., *Biopolymers* **22**(12): 2577–2637.
- Karlin, S. and Altschul, S. F. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes, *Proceedings of the National Academy of Sciences. USA* **87**(6): 2264–2268.
- Karney, C. F. (2007). Quaternions in molecular modeling, *Journal of Molecular Graphics and Modelling* **25**(5): 595–604.
- Karpen, M. E., de Haseth, P. L. and Neet, K. E. (1989). Comparing short protein substructures by a method based on backbone torsion angles, *Proteins: Structure, Function, and Genetics* **6**(2): 155–167.
- Kasarapu, P. (2015). Modelling of directional data using kent distributions, *arXiv preprint abs/1506.08105*.
URL: <http://arxiv.org/abs/1506.08105>
- Kasarapu, P. (2016). *Statistical Inference Problems with Applications to Computational Structural Biology*, PhD thesis, Monash University, Australia.
- Kasarapu, P. and Allison, L. (2015). Minimum message length estimation of mixtures of multivariate gaussian and von mises-fisher distributions, *Machine Learning* **100**(2): 333–378.
- Kearsley, S. K. (1989). On the orthogonal transformation used for structural comparisons, *Acta Crystallographica Section A* **A45**: 208–210.

- Kendall, M. (1938). A new measure of rank correlation, *Biometrika* **30**(1-2): 81–89.
- Kendrew, J. C., Bodo, B., Dintzis, H. M., Parrish, R. G., Wyckoff, H. and Phillips, D. C. (1958). A three-dimensional model of the myoglobin molecule obtained by X-ray analysis, *Nature* **181**(4610): 662–666.
- KenKnight, C. E. (1984). Comparison of methods of matching protein structures, *Acta Crystallographica Section A* **40**(6): 708–712.
- Kent, J. T. (1982). The Fisher-Bingham distribution on the sphere, *Journal of the Royal Statistical Society: Series B (Methodological)* **44**(1): 71–80.
- Kihara, D. and Skolnick, J. (2003). The PDB is a covering set of small protein structures, *Journal of Molecular Biology* **334**(4): 793–802.
- Kinjo, A. R., Suzuki, H., Yamashita, R., Ikegawa, Y., Kudou, T., Igarashi, R., Kengaku, Y., Cho, H., Standley, D. M., Nakagawa, A. and Nakamura, H. (2012). Protein Data Bank Japan (PDBj): maintaining a structural data archive and resource description framework format, *Nucleic Acids Research* **40**(Database issue): D453–D460.
- Kirkpatrick, S., D., G. C. and Vecchi, M. P. (1983). Optimization by simulated annealing, *Science* **220**(4598): 671–680.
- Kleywegt, G. J. (1996). Use of non-crystallographic symmetry in protein structure refinement, *Acta Crystallographica Section D* **52**(4): 842–857.
- Kleywegt, G. J. and Jones, T. A. (November 1994). A super position, *CCP4/ESF-EACBM Newsletter on Protein Crystallography* **31**: 9–14.
- Knuth, D. E. (1999a). *The art of computer programming*, Vol. 3, Addison Wesley.
- Knuth, D. E. (1999b). *The art of computer programming*, Vol. 3, Addison Wesley, pp. 426–458.
- Koehl, P. (2001). Protein structure similarities, *Current opinion in structural biology* **11**(3): 348–353.
- Kolbeck, B., May, P., Schmidt-Goenner, T., Steinke, T. and Knapp, E.-W. (2006). Connectivity independent protein-structure alignment: a hierarchical approach, *BMC Bioinformatics* **7**(510).
- Kolmogorov, A. N. (1963). On tables of random numbers, *Sankhyā: The Indian Journal of Statistics, Series A* **25**(4): 369–376.
- Kolodny, R., Koehl, P., Guibas, L. and Levitt, M. (2002). Small libraries of protein fragments model native protein structures accurately, *Journal of Molecular Biology* **323**(2): 297–307.
- Kolodny, R., Koehl, P. and Levitt, M. (2005). Comprehensive evaluation of protein structure alignment methods: scoring by geometric measures, *Journal of Molecular Biology* **346**(4): 1173–1188.
- Kolodny, R. and Linial, N. (2004). Approximate protein structural alignment in polynomial time, *Proceedings of the National Academy of Sciences. USA* **101**(33): 12201–12206.

- Konagurthu, A. S., Kasarapu, P., Allison, L., Collier, J. H. and Lesk, A. (2014). On sufficient statistics of least-squares superposition of vector sets, *RECOMB*, Vol. LNCS/LNBI 8394, pp. 144–159.
- Konagurthu, A. S., Lesk, A. M., Abramson, D., Stuckey, P. J. and Allison, L. (2013). Statistical inference of protein “LEGO bricks”, *Thirteenth IEEE International Conference on Data Mining*, pp. 1091–1096.
- Konagurthu, A. S., Lesk, A. M. and Allison, L. (2012). Minimum message length inference of secondary structure from protein coordinate data, *Bioinformatics* **28**(12): i97–i105.
- Konagurthu, A. S., Stuckey, P. J. and Lesk, A. M. (2008). Structural search and retrieval using tableau representation of protein folding patterns, *Bioinformatics* **24**: 645–651.
- Konagurthu, A. S., Whisstock, J. C., Stuckey, P. J. and Lesk, A. M. (2006). MUSTANG: a multiple structural alignment algorithm, *Proteins: Structure, Function, and Bioinformatics* **64**(3): 559–574.
- Krissinel, E. and Henrick, K. (2003). Protein structure comparison in 3D based on secondary structure matching (SSM) followed by C_α alignment, scored by a new structural similarity function., *Proceedings of the Fifth international Conference on Molecular Structural Biology*, Vol. 88, Vienna.
- Krissinel, E. and Henrick, K. (2004). Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions, *Acta Crystallographica Section D* **60**(12): 2256–2268.
- Kruskal, J. B. (1983). An overview of sequence comparison: Time warps, string edits, and macromolecules, *SIAM Review* **25**(2): 201–237.
- Lackner, P., Koppensteiner, W. A., Sippl, M. J. and Domingues, F. S. (2000). ProSup: a refined tool for protein structure alignment, *Protein Engineering, Design & Selection* **13**(11): 745–752.
- Lance, G. and Williams, W. (1966). Computer programs for hierarchical polythetic classification (“similarity analyses”), *The Computer Journal* **9**(1): 60–64.
- Lehmer, D. (1960). Teaching combinatorial tricks to a computer, *Proc. Sympos. Appl. Math. Combinatorial Analysis*, Vol. 10, pp. 179–193.
- Lesk, A. (1986). A toolkit for computational molecular biology. II. on the optimal superposition of two sets of coordinates, *Acta Crystallographica Section A* **42**(2): 110–113.
- Lesk, A. M. (1995). Systematic representation of protein folding patterns, *Journal of Molecular Graphics* **13**: 159–164.
- Lesk, A. M. (2000). The unreasonable effectiveness of mathematics in molecular biology, *The Mathematical Intelligencer* **22**(2): 28–37.
- Lesk, A. M. (2001a). *Introduction to Protein Architecture: The Structural Biology of Proteins*, Oxford University Press.
- Lesk, A. M. (2001b). *Introduction to protein architecture: the structural biology of proteins*, Oxford University Press.

- Lesk, A. M. (2010). *Introduction to Protein Science: Architecture, Function, and Genomics*, 2 edn, Oxford University Press.
- Leslin, C. M., Abyzov, A. and Ilyin, V. A. (2007). TOPOFIT-DB, a database of protein structural alignments based on the TOPOFIT method, *Nucleic Acids Research* **35**(suppl 1): D317–D321.
- Levine, M., Stuart, D. and Williams, J. (1984). A method for systematic comparison of the three-dimensional structures of proteins and some results, *Acta Crystallographica Section A* **40**(5): 600–610.
- Levitt, M. and Chothia, C. (1976). Structural patterns in globular proteins, *Nature* **261**(5561): 552–558.
- Levitt, M. and Gerstein, M. (1998). A unified statistical framework for sequence comparison and structure comparison, *Proceedings of the National Academy of Sciences. USA* **95**(11): 5913–5920.
- Linderstrøm-Lang, K. U. (1952). *Proteins and Enzymes. (Lane Medical Lectures.)*, Vol. 6 of *Stanford University Publications. University Series. Medical Sciences.*, Stanford University Press.
- Lo Conte, L., Ailey, B., Hubbard, T. J., Brenner, S. E., Murzin, A. G. and Chothia, C. (2000). SCOP: a structural classification of proteins database., *Nucleic Acids Research* **28**(1): 257–259.
- Ma, J. and Wang, S. (2014). Algorithms, applications, and challenges of protein structure alignment, *Advances in Protein Chemistry and Structural Biology* **94**: 121–175.
- Mackay, A. L. (1984). Quaternion transformation of molecular orientation, *Acta Crystallographica Section A* **40**(2): 165–166.
- MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, Cambridge, UK.
- Mardia, K. and Jupp, P. (1999). *Directional Statistics*, Probability and Statistics, Wiley, Chichester, England.
- Marsh, R. E. and Donohue, J. (1967). Crystal structure studies of amino acids and peptides, *Advances in Protein Chemistry* **22**: 235–256.
- Marti-Renom, M. A., Capriotti, E., Shindyalov, I. N. and Bourne, P. E. (2009). Structure comparison and alignment, in J. Gu and P. E. Bourne (eds), *Structural Bioinformatics*, John Wiley and Sons, chapter 16, pp. 397–417.
- May, A. C. (1996). Pairwise iterative superposition of distantly related proteins and assessment of the significance of 3-D structural similarity, *Protein Engineering* **9**(12): 1093–1101.
- McLachlan, A. D. (1972). A mathematical procedure for superimposing coordinates of proteins, *Acta Crystallographica Section A* **28**: 656–657.
- McLachlan, A. D. (1982). Rapid comparison of protein structures, *Acta Crystallographica Section A* **38**(6): 871–873.

- McLachlan, G. J. and Basford, K. E. (1987). *Mixture Models*, Vol. 84 of *Statistics: textbooks and monographs*, Marcel Dekker Inc., New York and Basel.
- Meier, S. and Özbek, S. (2007). A biological cosmos of parallel universes: Does protein structural plasticity facilitate evolution?, *BioEssays* **29**(11): 1095–1104.
URL: <http://dx.doi.org/10.1002/bies.20661>
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. and Teller, E. (1953). Equation of state calculations by fast computing machines, *The Journal of Chemical Physics* **21**(6): 1087–1092.
- Miao, X., Bryson, M. G. and Valafar, H. (2008). TALI: Protein structure alignment using backbone torsion angles, *Journal of Bioinformatics and Computational Biology* **6**(1): 163–181.
- Micheletti, C., Seno, F. and Maritan, A. (2000). Recurrent oligomers in proteins: An optimal scheme reconciling accurate and concise backbone representations in automated folding and design studies, *Proteins: Structure, Function, and Bioinformatics* **40**(4): 662–674.
- Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, Complex Adaptive Systems, The MIT Press, Cambridge, Massachusetts.
- Mizuguchi, K., Deane, C. M., Blundell, T. L. and Overington, J. P. (1998). HOMSTRAD: a database of protein structure alignments for homologous families., *Protein Science* **7**(11): 2469–2471.
- Mosca, R., Brannetti, B. and Schneider, T. R. (2008). Alignment of protein structures in the presence of domain motions, *BMC Bioinformatics* **9**(352).
- Moult, J., Fidelis, K., Kryshtafovych, A., Schwede, T. and Tramontano, A. (2014). Critical assessment of methods of protein structure prediction (CASP) – round x, *Proteins: Structure, Function, and Bioinformatics* **82**(Issue Supplement S2): 1–6.
- Moult, J., Pedersen, J. T., Judson, R. and Fidelis, K. (1995). A large-scale experiment to assess protein structure prediction methods, *Proteins: Structure, Function, and Bioinformatics* **23**(3): ii–iv.
- Muirhead, H. and Perutz, M. F. (1963). Structure of hæmoglobin, *Nature* **199**(4894): 633–638.
- Murzin, A. G. (1998). How far divergent evolution goes in proteins, *Current Opinion in Structural Biology* **8**(3): 380–387.
- 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, *Journal of Molecular Biology* **247**(4): 536–540.
- Myrvold, W. and Ruskey, F. (2001). Ranking and unranking permutations in linear time, *Information Processing Letters* **79**(6): 281–284.
- 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.

- Nguyen, M. N., Tan, K. P. and Madhusudhan, M. S. (2011). Click–Topology-independent comparison of biomolecular 3D structures, *Nucleic Acids Research* **39**(suppl. 2): W24–W28.
- Oliver, J. J. and Baxter, R. A. (1994). MML and Bayesianism: Similarities and differences (Introduction to minimum encoding inference – part ii), *Technical Report Tech Report 206*, Department of Computer Science, Monash University, Clayton, Vic. 3168, Australia.
- 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.
- Orengo, C. A. and Taylor, W. R. (1990). A rapid method for protein structure alignment., *Journal of Theoretical Biology* **147**: 517–551.
- Orengo, C. A. and Taylor, W. R. (1996). SSAP: sequential structure alignment program for protein structure comparison, *Computer Methods for Macromolecular Sequence Analysis*, Vol. 266 of *Methods in enzymology*, Academic Press, pp. 617–635.
- Orengo, C., Jones, D. and Thornton, J. (1994). Protein superfamilies and domain superfolds, *Nature* **372**(6507): 631–634.
- 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.
- Pandit, S. B. and Skolnick, J. (2008). Fr-TM-Align: A new protein structural alignment method based on fragment alignments and the TM-score, *BMC Bioinformatics* **9**(531).
- Pauling, L. and Corey, R. (1951). Configurations of polypeptide chains with favored orientations around single bonds: Two new pleated sheets., *Proceedings of the National Academy of Sciences. USA* **37**: 729–740.
- Pauling, L., Corey, R. and Branson, H. R. (1951). The structures of proteins: Two hydrogen bonded helical configurations of polypeptide chain., *Proceedings of the National Academy of Sciences. USA* **37**: 205–211.
- Pearson, R. (2007). Reciprocal rank-based comparison of ordered gene lists, *IEEE Workshop on Genomic Signal Processing and Statistics workshop*, pp. 1–3.
- Perutz, M. F., Kendrew, J. C. and Watson, H. C. (1965). Structure and function of haemoglobin. II. Some relations between polypeptide chain configuration and amino acid sequence, *Journal of Molecular Biology* **13**: 669–678.
- Ramachandran, G., Ramakrishnan, C. and Sasisekharan, V. (1963). Stereochemistry of polypeptide chain configurations, *Journal of Molecular Biology* **7**: 95–99.
- Ramachandran, G. and Sasisekharan, V. (1968). Conformation of polypeptides and proteins, *Advances in Protein Chemistry* **23**: 283–437.
- Rao, S. T. and Rossmann, M. G. (1973). Comparison of super-secondary structures in proteins, *Journal of Molecular Biology* **76**(2): 241–256.

- Raymond, J. W. and Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures, *Journal of Computer-Aided Molecular Design* **16**(7): 521–533.
- Richardson, J. S. (1981). The anatomy and taxonomy of protein structure., *Advances in protein chemistry* **34**: 167–339.
- Richardson, S. and Green, P. J. (1997). On bayesian analysis of mixtures with an unknown number of components, *Journal of the Royal Statistical Society. Series B (Methodological)* **59**(4): 731–792.
- Rissanen, J. (1978). Modeling by shortest data description, *Automatica* **14**(5): 465–471.
- Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length, *Annals of Statistics* **11**: 416–431.
- Rooman, M. J., Rodriguez, J. and Wodak, S. J. (1990). Automatic definition of recurrent local structure motifs in proteins, *Journal of Molecular Biology* **213**(2): 327–336.
- Roy, A., Kucukural, A. and Zhang, Y. (2010). I-TASSER: a unified platform for automated protein structure and function prediction, *Nature Protocols* **5**: 725–738.
- Russell, S. J. and Norvig, P. (2009a). *Artificial Intelligence: A Modern Approach*, 3 edn, Pearson Education.
- Russell, S. J. and Norvig, P. (2009b). *Artificial Intelligence: A Modern Approach*, 3 edn, Pearson Education, chapter 4.
- Rustici, M. and Lesk, A. M. (1994). Three-dimensional searching for recurrent structural motifs in data bases of protein structures, *Journal of Computational Biology* **1**(2): 121–132.
- Sael, L., Li, B., La, D., Fang, Y., Ramani, K., Rustamov, R. and Kihara, D. (2008). Fast protein tertiary structure retrieval based on global surface shape similarity, *Proteins: Structure, Function, and Bioinformatics* **72**(4): 1259–1273.
- Sali, A. and Blundell, T. L. (1990). Definition of general topological equivalence in protein structures. A procedure involving comparison of properties and relationships through simulated annealing and dynamic programming, *Journal of Molecular Biology* **212**(2): 403–428.
- Scheeff, E. D. and Fink, J. L. (2005). *Fundamentals of Protein Structure*, Vol. 44, John Wiley & Sons, Inc., chapter 2.
- Schwede, T., Kopp, J., Guex, N. and Peitsch, M. C. (2003). SWISS-MODEL: an automated protein homology-modeling server, *Nucleic Acids Research* **31**(13): 3381–3385.
- Schymkowitz, J., Borg, J., Stricher, F., Nys, R., Rousseau, F. and Serrano, L. (2005). The FoldX web server: an online force field, *Nucleic Acids Research* **33**(Web Server Issue): w382–w388.
- Shannon, C. E. (1948). A mathematical theory of communication, *Bell Systems Technical Journal* **27**: 379–423.
- Shatsky, M., Nussinov, R. and Wolfson, H. J. (2002). Flexible protein alignment and hinge detection, *Proteins: Structure, Function, and Bioinformatics* **48**(2): 242–256.

- Shatsky, M., Nussinov, R. and Wolfson, H. J. (2004). A method for simultaneous alignment of multiple protein structures, *Proteins: Structure, Function, and Bioinformatics* **56**(1): 143–156.
- Shindyalov, I. N. and Bourne, P. E. (1998). Protein structure alignment by incremental combinatorial extension (ce) of the optimal path., *Protein Engineering, Design & Selection* **11**(9): 739–747.
- 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.
- Simm, A. M., Baldwin, A. J., Busse, K. and Jones, D. D. (2007). Investigating protein structural plasticity by surveying the consequence of an amino acid deletion from TEM-1 β -lactamase, *FEBS Letters* **581**(21): 3904–3908.
- Simons, K. T., Bonneau, R., Ruczinski, I. and Baker, D. (1999). Ab initio protein structure prediction of CASP III targets using ROSETTA, *Proteins: Structure, Function, and Bioinformatics* **Suppl. 3**: 171–176.
- Singh, A. P. and Brutlag, D. L. (1997). Hierarchical protein structure superposition using both secondary structure and atomic representations, *International Conference on Intelligent Systems in Molecular Biology*, pp. 284–293.
- Sippl, M. J. (2008). On distance and similarity in fold space, *Bioinformatics* **24**(6): 872–873.
- Sippl, M. J. and Wiederstein, M. (2008). A note on difficult structure alignment problems, *Bioinformatics* **24**(3): 426–427.
- Slater, A., Castellanos, J. I., Sippl, M. J. and Melo, F. (2013). Towards the development of standardized methods for comparison, ranking and evaluation of structure alignments, *Bioinformatics* **29**: 47–53.
- Solomonoff, R. J. (1964). A formal theory of inductive inference. part i, *Information and control* **7**(1): 1–22.
- Spearman, C. (1904). The proof and measurement of association between two things, *The American Journal of Psychology* **15**(1): 72–101.
- Stivala, A., Wybrow, M., Wirth, A., Whisstock, J. and Stuckey, P. (2011). Automatic generation of protein structure cartoons with Pro-origami, *Bioinformatics* **27**(23): 3315–3316.
- Subbiah, S., Laurents, D. V. and Levitt, M. (1993). Structural similarity of DNA-binding domains of bacteriophage repressors and the globin core, *Current Biology* **3**(3): 141–148.
- Szustakowski, J. D. and Weng, Z. (2000). Protein structure alignment using a genetic algorithm, *Proteins: Structure, Function, and Bioinformatics* **38**(4): 428–440.
- Szustakowski, J. D. and Weng, Z. (2003). *K2: protein structure comparisons and their statistical significance*, Morgan Kaufmann, San Francisco, CA, pp. 61–86.
- Taylor, W. R. (2000). Protein structure comparison using SAP, in D. M. Webster (ed.), *Protein Structure Prediction: Methods and Protocols*, Vol. 143 of *Methods in Molecular Biology*, Humana Press, Totowa, New Jersey, chapter 2, pp. 19–32.

- Taylor, W. R. and Orengo, C. A. (1989). Protein structure alignment, *Journal of Molecular Biology* **208**(1): 1–22.
- Teichert, F., Bastolla, U. and Porto, M. (2007). SABERTOOTH: Protein structural alignment based on a vectorial structure representation, *BMC Bioinformatics* **8**(425).
- Theobald, D. L. (2005). Rapid calculation of rmsds using a quaternion-based characteristic polynomial, *Acta Crystallographica Section A* **61**(4): 478–480.
- Turing, A. M. (1950). Computing machinery and intelligence, *Mind* **59**(236): 433–460.
- Unger, R., Harel, D., Wherland, S. and Sussman, J. L. (1989). A 3D building blocks approach to analyzing and predicting structure of proteins, *Proteins: Structure, Function, and Bioinformatics* **5**(4): 355–373.
- Uniprot, C. (2010). Ongoing and future developments at the universal protein resource, *Nucleic Acids Research* **39**(Database issue): d214–d219.
- Unwin, P. N. T. and Henderson, R. (1975). Molecular structure determination by electron microscopy of unstained crystalline specimens, *Journal of Molecular Biology* **94**(3): 425–440.
- van Laarhoven, P. J. and Aarts, E. H. (1987). *Simulated Annealing: Theory and Applications*, Mathematics and Its Applications, Springer, Netherlands.
- Velankar, S., Alhroub, Y., Alili, A., Best, C., Boutselakis, H. C., Caboche, S., Conroy, M. J., Dana, J. M., van Ginkel, G., Golovin, A., Gore, S. P., Gutmanas, A., Haslam, P., Hirshberg, M., John, M., Lagerstedt, I., Mir, S., Newman, L. E., Oldfield, T. J., Penkett, C. J., Pineda-Castillo, J., Rinaldi, L., Sahni, G., Sawka, G., Sen, S., Slowley, R., da Silva, A. W. S., Suarez-Uruena, A., Swaminathan, G. J., Symmons, M. F., Vranken, W. F., Wainwright, M. and Kleywegt, G. J. (2011). PDB: Protein data bank in europe, *Nucleic Acids Research* **39**(Database issue): D402–D410.
- Vesterstrøm, J. (2006). *Heuristic Algorithms in Bioinformatics*, PhD thesis, University of Aarhus, Denmark.
- Vesterstrøm, J. and Taylor, W. R. (2006). Flexible secondary structure based protein structure comparison applied to the detection of circular permutation, *Journal of Computational Biology* **13**(1): 43–63.
- Vetter, I. R., Baase, W. A., Heinz, D. W., Xiong, J., Snow, S. and Matthews, B. W. (1996). Protein structural plasticity exemplified by insertion and deletion mutants in T4 lysozyme, *Protein Science* **5**(12): 2399–2415.
- Vogel, C., Bashton, M., Kerrison, N. D., Chothia, C. and Teichmann, S. A. (2004). Structure, function and evolution in multidomain proteins, *Current Opinion in Structural Biology* **14**(2): 208–216.
- Voronoi, G. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les paralléloèdres primitifs., *Journal für die reine und angewandte Mathematik* **134**: 198–287.
URL: <http://eudml.org/doc/149291>

- Vriend, G. and Sander, C. (1991). Detection of common three-dimensional substructures in proteins, *Proteins: Structure, Function, and Bioinformatics* **11**(1): 52–58.
- Wagner, G. and Wüthrich, K. (1978). Dynamic model of globular protein conformations based on nmr studies in solution, *Nature* **275**(5677): 247–248.
- Wallace, C. S. (1998). Intrinsic classification of spatially correlated data, *The Computer Journal* **41**(8): 602–611.
- Wallace, C. S. (2005). *Statistical and Inductive Inference using Minimum Message Length*, Information Science and Statistics, SpringerVerlag.
- Wallace, C. S. and Boulton, D. M. (1968). An information measure for classification, *Computer Journal* **11**(2): 185–194.
- Wallace, C. S. and Boulton, D. M. (1969). The information content of a multistate distribution, *Journal of Theoretical Biology* **23**(2): 269–278.
- Wallace, C. S. and Boulton, D. M. (1975). An invariant Bayes method for point estimation, *Classification Society Bulletin* **3**: 11–34.
- Wallace, C. S. and Dowe, D. L. (1999). Minimum message length and kolmogorov complexity, *The Computer Journal* **42**(4): 270–283.
- Wallace, C. S. and Freeman, P. R. (1987). Estimation and inference by compact coding, *Journal of the Royal Statistical Society. Series B (Methodological)* **49**(3): 240–265.
- Wallace, C. S. and Freeman, P. R. (1992). Single-factor analysis by minimum message length estimation, *Journal of the Royal Statistical Society. Series B (Methodological)* **54**(1): 195–209.
- Wallace, C. S. and Patrick, J. (1993). Coding decision trees, *Machine Learning* **11**(1): 7–22.
- Walle, I. V., Lasters, I. and Wyns, L. (2005). SABmark—A benchmark for sequence alignment that covers the entire known fold space, *Bioinformatics* **21**(7): 1267–1268.
- Wang, L. and Jiang, T. (1994). On the complexity of multiple sequence alignment, *Journal of Computational Biology* **1**(4): 337–348.
- Westbrook, J. D. and Fitzgerald, P. M. (2009). The PDB format, mmCIF formats, and other data formats, in J. Gu and P. E. Bourne (eds), *Structural Bioinformatics*, John Wiley and Sons, chapter 10, pp. 271–291.
- Westbrook, J., Ito, N., Nakamura, H., Henrick, K. and Berman, H. M. (2005). PDBML: the representation of archival macromolecular structure data in XML, *Bioinformatics* **21**(7): 988–992.
- Westhead, D. R., Slidel, T. W. F., Flores, T. P. J. and Thornton, J. M. (1999). Protein structural topology: Automated analysis and diagrammatic representation, *Protein Science* **8**(4): 897–904.
- Wetlaufer, D. B. (1973). Nucleation, rapid folding, and globular intrachain regions in proteins, *Proceedings of the National Academy of Sciences. USA* **70**(3): 697–701.

- Wheelan, S. J., Marchler-Bauer, A. and Bryant, S. H. (2000). Domain size distributions can predict domain boundaries, *Bioinformatics* **16**(7): 613–618.
- Wolfson, H. and Rigoutsos, I. (1997). Geometric hashing: an overview, *Computational Science Engineering, IEEE* **4**(4): 10–21.
- Xu, J. and Zhang, Y. (2010). How significant is a protein structure similarity with TM-score = 0.5?, *Bioinformatics* **26**(7): 889–895.
- Yang, Y., Zhan, J., Zhao, H. and Zhou, Y. (2012). A new size-independent score for pairwise protein structure alignment and its application to structure classification and nucleic-acid binding prediction, *Proteins: Structure, Function, and Bioinformatics* **80**(8): 2080–2088.
- Ye, Y. and Godzik, A. (2003). Flexible structure alignment by chaining aligned fragment pairs allowing twists, *Bioinformatics* **19**(suppl. 2): ii246–ii255.
- Zemla, A. (2003). LGA: a method for finding 3D similarities in protein structures, *Nucleic Acids Research* **31**(13): 3370–3374.
- Zhang, Y. and Skolnick, J. (2004). Scoring function for automated assessment of protein structure template quality, *Proteins: Structure, Function, and Bioinformatics* **57**(4): 702–710.
- 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. USA* **102**(4): 1029–1034.
- Zhang, Y. and Skolnick, J. (2005b). TM-align: a protein structure alignment algorithm based on the TM-score, *Nucleic Acids Research* **33**: 2302–2309.
- Ziv, J. and Lempel, A. (1978). Compression of individual sequences via variable-rate coding, *Information Theory, IEEE Transactions on* **24**(5): 530–536.
- Zotenko, E., Dogan, R. I., Wilbur, W. J., O’Leary, D. P. and Przytycka, T. M. (2007). Structural footprinting in protein structure comparison: the impact of structural fragments, *BMC Structural Biology* **7**(53).
- Zu-Kang, F. and Sippl, M. (1996). Optimum superimposition of protein structures: ambiguities and implications, *Folding and Design* **1**: 123–132.

Appendix A

List of randomly selected SCOP heirarchy domains

Benchmarking of various alignment programs on a dataset containing 2500 structural domain pairs selected randomly from SCOPe domain database (Fox et al., 2013). No two domains in the dataset share more than 40% sequence identity.

The dataset is identified using the following procedure. SCOPe (Fox et al., 2013) **version 1.75B** domains are used and separated out into buckets depending on their sizes (number of residues). Two domains in the same bucket differ no more than 50 residues in their lengths. The SCOP hierarchy for all the domains within each bucket is recorded. A pivot domain is randomly chosen from the entire SCOPe collection. Assume that this pivot domain falls with the i -th bucket. Using the SCOP (Murzin et al., 1995) hierarchy in this bucket, we select (at random):

- one domain that belongs to the same SCOP Family as the pivot
- one domain that belongs to the same SCOP Superfamily as the pivot, but not the Family
- one domain that belongs to the same SCOP Fold the pivot, but not the Family or Superfamily.
- one domain that belongs to the same SCOP Class the pivot, but not the Family, Superfamily or Fold.
- one domain that belongs to a different class (a Decoy domain).

This selection process is repeated until we have 500 distinct pivots and their respective five domains. This results in 2500 distinct structural domain pairs from 3000 unique domains. A table containing a list of the SCOP identifiers for these domains can be found overleaf.

190 APPENDIX A. LIST OF RANDOMLY SELECTED SCOP HEIRARCHY DOMAINS

Pivot	Family	Super Family	Fold	Class	Decoy	Pivot	Family	Super Family	Fold	Class	Decoy
d2d00a3	d2d00a2	d2i7na2	d1ekea.	d1pn0a2	d1rqga1	d1l9xa.	d1t3ta2	d1n57a.	d1l1a4a2	d2g5gx1	d1rh1a1
d2c51c1	d1k8rb.	d1j0ga.	d1g3ec.	d2grga1	d1bcoo1	d1ybea1	d1vlpa2	d1b5ta.	d1b5ta.	d3kqsa.	d2i1jra1
d1f5ma.	d2o9ca1	d2o9ca.	d1l3la2	d2gtia2	d1ig3a2	d2hjsa2	d2cvoa2	d1h6da2	d2gvia1	d1hzt.	d1na6a1
d1c8na.	d1f2na.	d1za7a.	d1m06g.	d1dq3a1	d1t35a.	d2pcca1	d2ns9a1	d1t17a.	d1f46a.	d2pnra2	d3c55g.
d2ah2a2	d1so7a.	d3b7ea.	d3amra.	d1f8v.1	d1kbla3	d1qypa.	d1tfla.	d1yuaa.	d1wgea1	d1wef.	d2f69a1
d2pida.	d3t2la.	d2d5ba2	d1ct9a1	d1xhba2	d2o6xa.	d1lxta.	d1omca.	d1c6wa.	d1mx5b5	d2fc6a1	d1m93.1
d1pgua2	d1nr0a2	d3uvka.	d1jtdb.	d1jeya1	d2hyec3	d1n8yc2	d3b2ua1	d1w8aa.	d1d89a.	d1j1a1a3	d2idba2
d1zj8a2	d2akja2	d3mca2	d1s99a.	d2nrka1	d1u5ka2	d1kloa2	d1kloa1	d1egfa.	d1uoya.	d2dkt1a	d1wkta.
d1gaa.	d2fbaa.	d1h54a1	d1x9da1	d1qsa1	d3qoma.	d3bwuc1	d1p5va1	d2bvua.	d2dpka1	d1ut7a.	d1qvea.
d1e4mm.	d1uwsa.	d1uoka2	d1reqb1	d1miob.	d1vdka.	d2ba0a2	d2nm612	d1v8qa.	d1bdoa.	d1o9ya.	d1j21a.
d1pgua1	d2ovrb2	d2pbib.	d1utca2	d2c3fa1	d1uw7a2	d1vh6a.	d1orja.	d3iqca.	d1t6ua.	d1p22a1	d1ada3
d1mm3a.	d2di0a1	d1wiva.	d1ejja.	d3ag3h.	d2fa8a1	d1lqza.	d1fxda.	d2fdna.	d1ffgb.	d3cjsb1	d1iea.
d1gjwa2	d1g5aa2	d1ggq1a	d1eexa.	d1hk8a.	d2p0ma1	d1sazal	d1g99a1	d2f5ja1	d1wb9a3	d1q3qa2	d1sk7a.
d2piaa1	d1tvca1	d1i8da2	d2dy1a1	d1so9a.	d2azea1	d1qxma1	d1ggpb2	d1dqga.	d2p39a.	d1yrka.	d2hvx2
d2f9wa1	d2f9wa2	d2bh1a2	d1p90a.	d2ioja1	d1hca1	d1uw4b.	d1hu3a.	d1t06a.	d3efza1	d2fm9a1	d1v12a2
d3mmsa.	d3nm6b.	d1ybf.	d1vhoa2	d1qwja.	d1x2ga2	d2vxxa.	d2c41a.	d2gyqa1	d2g3b1	d1d2zb.	d1t1ja.
d2d29a1	d1s1qa1	d1w07a2	d2pnra1	d1pprm1	d2qna1	d1ckma2	d1p16a2	d1a01a2	d2nu8b1	d1j55a.	d1j4a2
d1jaka2	d2gjxa2	d1l8na2	d1o29a.	d1j5ya2	d1kyqa.	d1rtqa.	d1fnoo4	d2q7sa1	d2gfga1	d2b9ea1	d2a1aa2
d1n7oa1	d1x1ia1	d1qaza.	d2sqca1	d1aora1	d1rrma.	d1jbea.	d2jbaa.	d1y71a.	d1dzfa1	d2fug1	d2a2aa2
d1q90b.	d1ppjc2	d1kqfc.	d1y5ic1	d1rwta.	d1t3ta7	d1ecaa.	d1q1fa.	d2kxia.	d1nek1	d1tj1a1	d2veea3
d1x58a1	d1akha.	d1gv2a2	d1k6ya1	d1utca1	d2ciob.	d1a3qa2	d1bvoo.	d1t4wa.	d1e2wa1	d1gwya.	d1r0da.
d2i1za.	d2gvka1	d2cb2a1	d2i0ka1	d1j10a.	d2mzwa1	d2vc7a.	d3pnza.	d2imra2	d3bl1a.	d2b17a1	d1tvfa2
d1djqa1	d1gtea2	d3krua.	d1tj1a2	d1woha.	d1v5va2	d1vema2	d1j1ia3	d1uuqa.	d1o1ta.	d1x31a.	d1g6sa.
d1m65a.	d2anua1	d1v6ta.	d1v6ta.	d1wvza1	d1v96a3	d1w96a3	d1w96a3	d2io8a3	d1xda.	d1v4aa2	d3hxa.
d1nupa.	d1k4ma.	d1jhda2	d2c5sa1	d3zqua.	d2jmu1	d1es9a.	d2hsja1	d1z8ha.	d2uubb1	d2yuba.	d1n1ba1
d2ax3a1	d1kyha.	d3pza.	d1o5za2	d2z6ra.	d1llaa2	d2etja1	d1i39a.	d1bcoa2	d2fxua2	d1ooya1	d1vdwa.
d2fhfa3	d2j43a1	d1ac0a.	d2h6ua.	d2qazd.	d2w6ka1	d2gu3a1	d2gu3a2	d3imab.	d1ugia.	d2yvgq1	d1ppj1
d2ywx.	d3rg8a.	d1qcza.	d1tlla2	d2j07a2	d2avue1	d1a9xb2	d1gpm2	d1fyea.	d2o14a2	d1j3ba2	d1zt2b1
d1j8ia.	d1nr4a.	d1zxta.	d2c67a1	d1hv2a.	d1i8na.	d3fsaa.	d2ov0a.	d1hfu1	d1kzqa2	d3qe1a.	d1t3q1
d1r17a1	d1n67a1	d2bcm.	d1x91a.	d1wjxa.	d1w75a1	d2ab0a1	d2fexa1	d1p80a1	d2c4wa.	d1rqp2	d1kxua1
d1ecfa2	d1q15a2	d1gk9.1	d2ntka1	d3f5vb.	d1v77a.	d3hh8a.	d1toaa.	d3cx3a.	d2xwpa.	d31jwa.	d2p0ya2
d2zbl.	d1f3pa.	d1h12a.	d1w6ka1	d1ux5a.	d3o83a.	d1vk0a.	d3mxa.	d1kcf2	d3ezwa2	d2fug12	d1j77a.
d1p28a.	d3cz1a.	d3k1ea.	d2ciwa1	d1j3d1	d1wj4a.	d2f2ha4	d1zy9a2	d1kfw1	d2choa2	d2f7fa1	d1le1a1
d3buxb1	d1h8ba.	d1k9ua.	d1iioa.	d1c75a.	d1lnqa2	d1r85a.	d1qw9a2	d1kfw1	d1muwa.	d1zmra.	d2g5da1
d2c9a1	d2fxta1	d2rfra1	d1x9ya2	d1ae9a.	d1aoca.	d1tx2a.	d1eyea.	d2ycix.	d1ykw1	d2c42a4	d1h9aa2
d2gu2a1	d1y4a1	d1r3na1	d1c8ba.	d1lara1	d1svba2	d2i1qa1	d1szpa1	d1u91a.	d1a0pa1	d1smye.	d2dkt2
d1ugka.	d1w6ma.	d2k3ua1	d1dcea2	d2f9ha1	d1gg3a1	d1vl2a1	d1gpm1	d1efva1	d1o6fa.	d1wpna.	d1igra3
d2go2a.	d3iira.	d1wbaa.	d2vxti.	d2edma1	d1evya2	d1g7sa1	d4ac9a2	d2f11a2	d1qfja1	d1v76a.	d1t50a.
d1l0wa2	d1c0aa2	d2p8ta2	d1ru0a.	d2uubc2	d1ux6a2	d1l5pa.	d1czpa.	d1rmc2	d1tifa.	d1wiba.	d1o7d.2
d2l1ja2	d2c4ja2	d2b5ea2	d1qmha1	d1hf2a2	d1b67a.	d1o7fa3	d1i5za2	d1ft9a2	d1vj2a.	d1ra0a1	d2pu9a.
d2q1ma.	d2wqla.	d2i15a1	d1xsza2	d1odha.	d1k3ra2	d1igra2	d1n8yc1	d1koha1	d1io0a.	d2xbla.	d1e5ha1
d1hx0a2	d1w2aa2	d1qbba3	d1aoca.	d3k40a.	d1zhxa.	d2dixa1	d1di2a.	d1pkpa2	d1dq3a2	d2hg7a1	d1ada2
d1q15a1	d1kqpa.	d2e18a.	d2a84a.	d1vpta.	d1rrka1	d2cp8a1	d1pgya.	d1v92a.	d1bvs1a1	d1z1va1	d1ekh2
d1sefa.	d1sq4a.	d1xrua1	d1w9ya1	d1upsa1	d1g5ga1	d1mf1a.	d1ub4a.	d3hpwa.	d1ng2a2	d1jly2a.	d1m9za.
d3t4na.	d2qrda.	d3osea.	d2c4ba1	d2uca1	d2gfa2	d1bd8a.	d1awcb.	d3twra.	d1lm5a.	d3trwa.	d1nu1a
d3g2sa.	d1dvpa1	d1hf8a2	d1k8kg.	d2euca1	d1kafa.	d1y7ea2	d1y0ya3	d2odfa1	d1boub.	d1p9ag.	d1okca.
d1a11b.	d1b8da.	d3pt8a.	d1gtea1	d1y91a.	d2ra5a1	d2v0na2	d1s8na.	d1qo0d.	d1vi6a.	d2jaqa.	d1n08a.
d1b34a.	d1d3bd.	d2rb6a1	d2f5tx1	d2pu9b.	d1jwqa.	d1xova2	d1xova2	d1q71.1	d1p7ha.	d2ptha.	d1vpa.
d1ws8a.	d2ux6a.	d2dyrb1	d1kzqa1	d1lcy1a	d2yx6a.	d2vaca.	d1vkka.	d2fh1a3	d2gnxa2	d2rfua.	d1pjqm3
d2c35a1	d1y14a.	d1yt3a1	d1m6ya1	d2au5a1	d2mmla1	d1fx7a3	d2h3ja1	d1igqa.	d1xyja.	d1nh2d2	d1q3a2
d2pfsa.	d3hgma.	d1tq8a.	d1coza.	d1p5dx1	d1zta2	d1z84a2	d1z84a1	d1vlra1	d2r7ja1	d3h0na.	d1vmoa.
d2d1ka1	d1vega.	d1wgl.	d1ufza.	d1g8qa.	d1mka2	d1twf12	d1pfta.	d1yuaa2	d1pd0a5	d1v1vka.	d1ojha.
d3i3qa.	d2iuwa1	d1nx4a.	d1sfna.	d2vhka.	d2qfia2	d1ks8a.	d1clca1	d3p2ca.	d2g0da.	d2o6ia1	d1rh5a.
d1h4ra2	d1m1xa2	d2v76a.	d1ywy1	d1tg7a1	d1gg4a3	d1pfpa.	d1gvpa.	d1n9wa.	d2exda1	d2cmea1	d2ct5a1
d2spca.	d1usp1a1	d2f31a.	d2s9pa1	d1xcpa1	d2cr9a1	d1z6ca2	d1zbsa1	d1e4ft1	d1hjra.	d2ftsaa3	d2nr9a2
d1l2ha.	d2il2a.	d2fdbm1	d1n4ka2	d1js8a2	d2b8e1	d1ex2a.	d2ama1	d1u14a.	d1w94a1	d1k1ja.	d1r8ia.
d3hyfa.	d1zbf1a1	d1cxqa.	d2w6ba.	d1f7na1	d1ppj1	d1nf4a.	d1v1ga.	d2it2a.	d2g2da.	d1r4va.	d1w8a1
d3ea1a.	d2plca.	d2zkmr4	d1vppqa.	d1q74a.	d1lsha2	d1lsha2	d1piia2	d1i4na.	d3gd1a.	d3a5fa.	d2v03a.
d2dg1a.	d1pjxa.	d2p4oa1	d1o6za.	d1vq8b1	d1ryoa.	d2daq1	d3qbya.	d2jnga1	d2do3a1	d1sddb1	d2d1pc1
d2cg5b.	d2jq4a1	d1dnya.	d1qrj1	d2nn4a1	d1h75a.	d1g8ka1	d2jioa1	d3ouga.	d1n10a2	d1g86a.	d3m5ra.
d1bx7a.	d1skza1	d2pw8i.	d2g81i1	d1e8ra.	d2clyc.	d1h9ka1	d1h9ra1	d1guta.	d2qo0a1	d1okia1	d2oee1a
d1vjxa.	d1jiga.	d2ib0a1	d2idxa.	d1wi1ja.	d1n8yc4	d1cida2	d2d9qb1	d1rhfa1	d1cvra1	d1uz5a1	d1vcca.
d1t0ia.	d1rtta.	d2z98a.	d1pjca2	d2cc0a1	d1tkea2	d1s28a.	d1xkpb1	d3epua.	d2od0a1	d1ckva.	d1cqx3
d1e8ga1	d1wvfa1	d1w1oa1	d1hbna2	d1u8xx2	d1sr8a.	d1pyfa.	d1us0a.	d3n6ga.	d1ec7a1	d1nd6a.	d1vyr1a
d1oiha.	d3pvja.	d2r6sa.	d1zx5a1	d1ja1a1	d2q0ta1	d3bpse1	d1lmja1	d2j5ha1	d3b2ua2	d2i9wa3	d1y43.1
d1l10c.	d1xjva2	d1je5a.	d3q46a.	d1n67a2	d1z9ha1	d1t6ca1	d1u6za2	d1woqa2	d1pv9a1	d1gg4a1	d1kpa3
d1aym3.	d1qqp3.	d3c1ib.	d1hx6a1	d2pbka.	d1b5ea.	d1edqa2	d1goia2	d1jaka1	d2qapa1	d1yvua2	d3ivra.
d1eh1a2	d1iowa2	d1eucb2	d1s68a.	d1sawa.	d2p62a1	d2bgwa2	d1j24a.	d1y88a2	d2g6a1	d2hy5c1	d1hm6a.
d3u5wa.	d3ec6a.	d2hq7a1	d2rdea1	d1f39a.	d1p5dx3	d1j30a.	d1yuzal	d2oh3a1	d1niga.	d2f2ab1	d1kf6d.
d1y7ba2	d3cu9a.	d1vkda.	d1s1da.	d1jeyb1	d1taza.	d1zh8a2	d1t1ta2	d1drua2	d1ygya1	d2fbd.	d1sfa.
d1pbyb.	d1jmx.	d1l0qa2	d2ebsa2	d2bbva.	d3arca.	d1tdja2	d1tdja2	d1drua2	d1wga1	d1t95a2	d1lua2
d1wdpa.	d1bf2a3	d1w6ga1	d1nth.	d2impa.	d1rt8a.	d2ejqa1	d3e11a1	d1c7ka.	d1nowa2	d1dm9a.	d1jhna3
d2tbva.	d1opoa.	d2mev1.	d1vpsa.	d1qexa.	d2elba1	d1uoca.	d2d5ra1	d1w9ha1	d1k8ka.	d1sa3a.	d1v0wa1
d3mmha.	d1ysqa.	d1mc0a2	d2p7ja2	d2hiya1	d2yvxa3	d1sfsa.	d1jfxa.	d2waga.	d1vhca.	d2pw6a1	d1l1fa.
d1yfua1	d1zvf1a1	d1ylla1	d1s4ca.	d3qzra.	d2hxa1	d1vq8t1	d2j01y1	d2j01t1	d1i1ja.	d1o1ma2	d2nllb.
d3dssb.	d1n4qb.	d1hzfa.	d1g6ma.	d2ebfx1	d2aox1	d1q0qa3	d1j5pa3	d1lc0a2	d2glza1	d2cs0a1	d1qb2a.
d2hbva1	d2gwga1	d1ra0a2	d2ejaa.	d1w36c3	d1snza.	d1v5da.	d1kwfa.	d1nc5a.	d1r76a.	d3m5qa.	d1y4wa2
d1v8ca1	d1vjka.	d2hj1a1	d3fila.	d1xmta.	d2ayda.	d1em2a.	d1ln1a.	d2de6a2	d2fpa1	d1ml1a.	d1jm1a.
d1auya.	d1ddl1a.	d1qqp1.	d2bpa2.	d1y0a.	d1ay0a.	d1g4ma2	d1g4ma1	d4a02a.	d1a02a.	d1bd7a.	d2ieaa3
d1kq1a.	d2y1ba.	d1ycya1	d1zq1a1	d2xcmc.	d1kg1a.	d3bw6a.	d1ifqa.	d2j3wa.	d3nula.	d1ewq4	d1kf6c.
d1h81a1	d1uwy1a1	d2b59b2	d1vema1	d2hf1a1	d1t3ua.	d1hxva.	d1jnsa.	d1o1ga3	d3eipa.	d1wiha.	d1j6ua1
d2zjda.	d3d32a.	d1wfy.	d2ah7a.	d2ae6a1	d1rha1	d1op4a.	d1l3wa3	d1u2ca1	d1e3ja.	d3czza.	d1z8a2
d1jha.	d2h0va.	d2phda1	d2piga.	d2piga.	d2hja1	d1ysja1	d2q43a1	d1de4c3	d2a8ya.	d1h9a.	d1nh1a.
d2g0wa1	d2q02a1	d1yx1a1	d1jz8a5	d1u6ka.	d3rlfg.	d2fr5a.	d1r5ta.	d1vk9a.	d1oioa.	d3cls2	d2apla1
d2chu1	d2w18a.	d1pp4a.	d2xvya.	d1zmba1	d3prna.	d2i9aa1	d1szba2	d1lr7a1	d1cbha.	d3coqa1	d1e3ha4
d2csba2	d2csba1	d2a1ja1	d1yt3a2	d1ykh1a1	d1g4yb.	d1gqxa.	d1ys7a1	d1fc3a.	d1r71a.	d1or7a2	d1wira.

Pivot	Family	Super Family	Fold	Class	Decoy	Pivot	Family	Super Family	Fold	Class	Decoy
d2ccma.	d2hpsa.	d1sraa.	d2ciwa2.	d2z16a.	d2ia1a1.	d1nyc.	d1oh1a.	d1smpi.	d2gt1n1.	d3kt9a.	d2f1ra.
d1m7xa3.	d1lwha2.	d1x38a1.	d1m7ja3.	d1zq1a2.	d2zbx.	d2qypb.	d1l91a.	d3bqpa.	d1m1eb.	d1m1eb.	d3qmx.
d1th5a1.	d1veha.	d3cq1a.	d1ib8a2.	d3dnja.	d1nkza.	d1oz2a2.	d1oz2a1.	d3flga.	d1ixda.	d2f2ha1.	d1igr1a.
d1v5ta.	d1v86a.	d1wz3a1.	d1wmha.	d2p7ja1.	d1n7da7.	d3br8a.	d2bjda.	d1urra.	d1u0sa.	d2ffga1.	d1l8ca.
d1eu3a1.	d2z81a1.	d1c4qa.	d1bkba2.	d1gh5a.	d1a79a2.	d1nekb2.	d3c8ya2.	d1xlqa.	d2byea1.	d2hg6a1.	d1o70a1.
d1h6ka1.	d1h6ka3.	d1xqra1.	d1ouva.	d2es4d1.	d1u8va2.	d1b4fa.	d1pk3a1.	d3bs5b.	d1doqa.	d2c9wa1.	d2bskb1.
d2bkya.	d2bkyx.	d1vm0a.	d3faua.	d1jg5a.	d1t16b1.	d1yr2a2.	d1o12a2.	d2ffia1.	d1ur3m.	d1v7za.	d1slqa.
d1ckaa.	d3rnja.	d2v1qa.	d1vbva1.	d1v1ha1.	d2ew1a1.	d2g9wa1.	d1sd4a.	d1bm9a.	d1igna2.	d1iq0a1.	d1v2ba.
d1gsoa1.	d1vkza1.	d1e2wa2.	d2pnrc1.	d2ccva1.	d2djaa1.	d1yuda1.	d1xe7a.	d1ep0a.	d1o7fa2.	d1pkha.	d1n8yc3.
d1od5a2.	d1lrha.	d1x82a.	d1yoxa1.	d2brja1.	d1m4ua.	d1ecea.	d1edga.	d1kwga2.	d1iq8a1.	d2oz1a1.	d3n2wa.
d1nh2a2.	d1nh2a1.	d1mpga2.	d1v8ca2.	d2jdqd.	d1tpga2.	d1h6ga1.	d1t01a2.	d2b0ha1.	d1k04a.	d1tf5a1.	d1xo1a2.
d1geqa.	d3nz1a.	d1znaa1.	d1xi3a.	d2ieaa1.	d2ozbb1.	d2nrha2.	d3bexa2.	d1okja2.	d1o13a.	d1t3ka.	d2q7a1.
d1s69a.	d2gkma.	d3mkbb.	d1kf6b1.	d2eiaa2.	d1pc6a.	d1mqva.	d1s05a.	d256ba.	d1gs9a.	d1lkoa1.	d1jfla2.
d1l6xa1.	d1l6xa2.	d1fo0b.	d2qfra1.	d1ou8a.	d1rmja.	d1kmta.	d3t5gb.	d1s6na.	d2nqda.	d1dkza2.	d1xjca.
d1d7qa.	d1jjga.	d1twfh.	d3chbd.	d1whia.	d2p0sa1.	d1dfca4.	d1dfca1.	d1hcd.	d1pwaa.	d2ofca.	d1zzwa.
d1v5qa.	d1ohl.	d3dz1a.	d1r30a.	d1j7xa.	d1ta8a.	d1odza.	d1rh9a1.	d1tg7a5.	d1dosa.	d1ir6a.	d2d0ta1.
d2y5fl.	d2b5fl.	d1k1oa3.	d1t0za.	d1xjha.	d1l7ba.	d1r12a2.	d3bzka4.	d1k3ra1.	d1v1ia1.	d2h3ra1.	d1v1ia1.
d1gdt1.	d1jwcv.	d1x41a1.	d3cjr1.	d2er1a.	d1z1hb1.	d1u01a1.	d1vq8a2.	d1woca.	d2k5qa1.	d1zgha1.	d1t1va.
d3e6sa.	d3ak8i.	d2z90a.	d3ci3a.	d3brja1.	d1d40a.	d1th7a1.	d1d3ba.	d2vxf1.	d1b8a1.	d1fc6a3.	d2q7a1.
d1xc3a2.	d2hoea2.	d1t6ca2.	d1j11a.	d1knxa2.	d1slhb.	d1vcoa1.	d1k9vf.	d1vhqa.	d3f6ya.	d2b0a2.	d2nubd1.
d1iyjb4.	d1ue1a.	d1b8aa1.	d2k5wa1.	d1oa8a.	d1tx9a1.	d2ja9a1.	d2ba0a1.	d1u5ka1.	d2p5zx1.	d1nqja.	d1ybz1.
d2pg3a1.	d3rk1a.	d1wy5a1.	d1g8fa2.	d2wqka.	d2o34a1.	d1pea2.	d1wuaa2.	d1kvka2.	d1qd1a1.	d1t3ta3.	d1i12a2.
d1efub3.	d1x2b1.	d1oai.	d1ixsa.	d2oy9a1.	d1l6ja3.	d1p4ua.	d1gyva.	d2g30a1.	d1m1xa2.	d1jb3a.	d2pv4a1.
d1vuyq1.	d2bsya2.	d3p48a.	d1tula.	d2hrva.	d1em8b.	d2c9qa1.	d2c9qa1.	d2aqfa1.	d1wfwj.	d1x3aa1.	d1iow1.
d1q3ja.	d1dkca.	d1c4ea.	d1hyka.	d2pf1a2.	d1ugx.1.	d1pula1.	d1w1ma1.	d1eg3a1.	d1c3za.	d1r8a1.	d1sf8a.
d1hxm2a.	d1uvq1.	d3kvqa.	d1yq2a2.	d2j9na.	d1f81a.	d1ulja1.	d1ulja2.	d1r3sa.	d2n1ia.	d1yovb.	d2v09a.
d1ytda2.	d1qpoa2.	d1vlp1a1.	d1fm0e.	d1puca.	d1abva.	d1r2f12.	d1fp5a2.	d1oag1.	d1cvva3.	d3ivva.	d1uz5a3.
d3oeqa.	d1ew4a.	d2fug71.	d1sgoa.	d1u58a2.	d3dtub2.	d3k1sa.	d3l8ra.	d2e2aa.	d1neka1.	d1kv9a1.	d1m18a.
d2te1a.	d1od5a1.	d1o16a.	d1pg6a.	d2f2ha2.	d1e12a.	d1iufa1.	d1hlva1.	d1irza.	d1irp3a1.	d1x13c1.	d3ag31.
d3atya.	d3l51a.	d1q45a.	d4ubpc2.	d1mxma.	d3c0na2.	d1ujpa.	d1nsja.	d1yxya1.	d2tpsa.	d1uana.	d1omra.
d3z6a.	d2vwsa.	d1m3ua.	d3mu7a.	d1nbaa.	d2bs2c.	d1jnda1.	d1n0wa1.	d1n0wa1.	d3n3ma.	d1xkua.	d1wd3a1.
d1j17a.	d1sv0c.	d1dxsa.	d1y88a1.	d1ny9a.	d2btia.	d1l0oc.	d1or7a1.	d1smyf1.	d1sfea1.	d2qalt1.	d1se0a.
d1a8pa1.	d1fdra1.	d1hzea.	d2ey4c1.	d2dlba1.	d3cxf5f1.	d3k0ta.	d3c3ma.	d1dz3a.	d1ykg1a1.	d2ozl2b2.	d1zodb.
d1myna.	d1ozza.	d1fjna.	d1lu0a.	d1bbga.	d1t1ra3.	d3psma.	d4aaza.	d1n8ma.	d1gl4a2.	d2fcwb1.	d1nh2b.
d1lkoa2.	d1yuza2.	d1vzia2.	d1zina2.	d1sopa.	d1yjb2.	d1sisa.	d1pvza.	d1mm0a.	d1i8xa.	d1sh1a.	d1pnb.1.
d3nxb.	d3u51a.	d3d7ca.	d1v9a1.	d3ryce.	d2b2ya1.	d2iyga.	d2h2na.	d2h2na.	d2qlxa.	d3ci0k3.	d1qjpa.
d2ozja.	d3hqxa.	d2pa7a1.	d1wpg1a1.	d1m5q1.	d2h5ma1.	d1v44a.	d1d16a.	d1wiia.	d1pg5b2.	d1vz2a2.	d1p35a1.
d1rx0a1.	d2c12a1.	d1x07a1.	d1xg2b.	d1bg1a1.	d1l7g7a.	d2bhua1.	d1ulva2.	d3rgha.	d1f00i1.	d2qs8a1.	d1z60a1.
d1p1xa.	d2a4aa1.	d1v1ia2.	d1uasa2.	d1zcz2a2.	d2ffja1.	d3c2wa3.	d2ool2a2.	d1l18a.	d3cpta.	d3ksva.	d3ft1a.
d3ctaa2.	d2vbua.	d1mrza1.	d1cqxa2.	d1wkaa1.	d2176a2.	d2dma1.	d1qfha2.	d2q0zx2.	d2vzsa1.	d3hcg.	d1sqga1.
d1k66a.	d1a20a1.	d1r8ja2.	d1t1ja.	d1hc7a1.	d1u61a.	d1w5a.	d1jt8a.	d3ersx.	d2ch4w1.	d1zata1.	d1jb0d.
d2f5a1.	d1nrja.	d2vg1m1.	d2a21a1.	d1oo0a.	d2fyga1.	d1nz6a.	d1iura.	d2o37a.	d2ca5a1.	d1hp8a.	d2jfga1.
d3bzka1.	d3bzka1.	d2csba3.	d1ci4a.	d1ujsa.	d1lpva.	d1a17a.	d1hh8a.	d2hr2a1.	d1wy6a.	d1ile1a.	d3iwt.
d1ezwa.	d1rhca.	d1luca.	d1t29a.	d1vxva.	d1mtp.1.	d1zunb1.	d1xe1a.	d1sqra.	d1kz1a1.	d1f1sa3.	d1mgta2.
d1ufaa1.	d2b5dx1.	d3bvua1.	d1nu9c1.	d1t3wa.	d2nm6h3.	d1tugb.	d1i5na.	d1sr2a.	d1n1xa.	d1seda.	d1vq8y1.
d2j5ya.	d1xvha1.	d3qwoc.	d1gvna.	d2f66c1.	d1auua.	d1ug3a1.	d1h6ka2.	d3b7sa1.	d1r8se.	d3b57a1.	d1nyoa.
d1nh8a2.	d1h3da2.	d3dfea.	d2e29a1.	d1xhca3.	d2huc.	d1noa2.	d2v3ga.	d2aama1.	d2wzma.	d1c0pa1.	d1qmg1.
d1fsea.	d1l31a1.	d1opca.	d2oa4a1.	d2ahma1.	d2cu2a1.	d1rhfa2.	d1ev2e2.	d1ev2e2.	d1cvva2.	d2pp6a1.	d2qsb1.
d2ch5a2.	d1zbsa2.	d2i7na1.	d2uubk1.	d1pdoa.	d2k0ua1.	d1om2a1.	d2isy1a1.	d2hoea1.	d1vz0a1.	d2b3a1.	d2dbs1.
d1ew0a.	d1v9ya.	d2g3ja.	d2inwa1.	d2hnga1.	d1mdal.	d1t2ya.	d1vk4a.	d2c4ea.	d1p9oa.	d1wht.1.	d2dra1.
d1jmva.	d1mjha.	d2duma.	d1dlja3.	d2q07a2.	d2f0ca1.	d3kucb.	d1l1fa.	d1l42a.	d1nyra2.	d1nea1.	d1vho1a.
d1ft9a1.	d2zcva1.	d1u5tb2.	d1v0ua.	d2dt6a1.	d1y8xb1.	d1kgsa1.	d2f4a1.	d1p4wa.	d1skn.	d1p4skn.	d1j8ba.
d1fjba1.	d1n26a1.	d2mxyb1.	d2je8a2.	d2b97a.	d3pc7a.	d1d0na3.	d1kcca.	d2xfaa.	d2qtva4.	d3mjza.	d1ez3a.
d2vo8a1.	d1aoha.	d1exha.	d1ayo.	d2geca1.	d1reva.	d1y05c1.	d1puee.	d1b6aa1.	d1mgt1a1.	d1vmaa1.	d1uc8a1.
d1ep6a.	d1odma.	d1ds1a.	d1qrva.	d3liza.	d1t0qa.	d1iuya.	d1tuva.	d1tuva.	d1vg3a.	d3kpb.	d1ijya.
d1y11a.	d1j78a.	d2wmya.	d1iiba.	d1m0wa1.	d3n90a.	d2uub01.	d2nn6i1.	d3kdfb.	d1k28a1.	d2fhfa4.	d1j6ua2.
d1jgta1.	d1xnga1.	d1efvb.	d3mxta.	d1qlaa.	d1yuj.	d1nr0a1.	d1n0wa1.	d1n0wa1.	d1yfga.	d1n7za.	d1zro2a.
d1xbia1.	d1zwa1.	d1dt9a2.	d3c9ua1.	d2hc5a1.	d1jkea.	d1bqca.	d3emca.	d2oxna.	d1lqaa.	d1tuga1.	d1ujna.
d4av5a.	d2j2zb1.	d1amxa.	d2dewx1.	d1f35a.	d1yb3a1.	d1sfxa.	d2d1ha1.	d2gmga1.	d1l8qa1.	d1jr3a1.	d1p5dx2.
d1ug1a.	d1ayja.	d1pe4a.	d1wqj1.	d1ha8a.	d2o2a1.	d1v0aa.	d1v3a.	d3b6na.	d3byqa1.	d1vbk2.	d2hqv1.
d1b9ra.	d1doia.	d1kf6b2.	d2gria1.	d2bwka.	d2oara1.	d2c42a5.	d1jnr.	d1xera.	d1kkha2.	d2qda.	d1q5za.
d1bupa1.	d1e4ft2.	d2fsja2.	d1chma1.	d2bfwa1.	d3gp6a.	d1ub9a.	d2a61a1.	d1u5ta1.	d1whua.	d1q06a.	d1qcsa2.
d1zcca1.	d1vd6a.	d3no3a.	d1ct5a.	d2btoa1.	d2vdfa.	d2qlvb1.	d1z0na1.	d2nqca1.	d2a4ca.	d1sg5a1.	d1vioa2.
d2rxta.	d2wnxa.	d1qbaa2.	d1f01a1.	d2p5zx2.	d1f01a3.	d1krha1.	d2bmwa1.	d1kz1a2.	d2c78a1.	d1hdfa.	d1hb2p4.
d1ag7a.	d1cmna.	d1lmra.	d1oiga.	d1k81a.	d1biaa2.	d1rqb2.	d1sr9a2.	d1of8a.	d1j93a.	d3l1na.	d1xf1a.
d2gdqa1.	d1kkoa1.	d2akza1.	d1vcfa1.	d3ryia.	d2yaya.	d1joga.	d1wtya.	d1o3ua.	d2mhra.	d2enda.	d1usub.
d1qzga.	d1kx1a.	d1fl1a.	d2f4ia1.	d1jlya1.	d1pfva1.	d1jmca1.	d1wjja.	d3irba.	d1enf1a1.	d2vsmb.	d2c1ya1.
d1p1ma2.	d2paja2.	d2icsa2.	d3taob.	d1m3ka1.	d1vb9a1.	d2e1za2.	d1saza2.	d1ig8a1.	d2f96a1.	d2pyya.	d2b1ba1.
d1vgpa.	d3e5ua2.	d2bgca2.	d3bb6a1.	d1pm4a.	d2fefal.	d1zzma1.	d1yixa1.	d1xrt2.	d1twia2.	d1thuxa.	d4geva.
d1f7da.	d2bsya1.	d2xcca.	d4ubpb.	d1zrua3.	d3kxsa.	d2g9hd1.	d1v1oa1.	d1prtf.	d2z1ca1.	d1gmua1.	d2isy2a.
d1b24a2.	d1b24a1.	d1dq3a3.	d3bp3a.	d1i3ja.	d1adu1.	d1jca1.	d2fxua1.	d2hoea3.	d1qtm1a1.	d1nula.	d1ug7a.
d5nula.	d1f4pa.	d1rlja.	d1mx3a2.	d1lk5a1.	d1qgha.	d2a13a1.	d2cr5a1.	d2bwfa.	d1wxq2.	d1nija2.	d1k1fa.
d1vi7a2.	d2cvea2.	d1t95a3.	d3iwl.	d1tiga.	d1g8ea.	d2c60a1.	d1oeya.	d1c9fa.	d2fug13.	d2fdo1a1.	d2qk1b1.
d2b3ra.	d3fbka.	d2yrba1.	d2huha1.	d1ti6a1.	d1vlb1.	d3iara.	d2pgfa.	d1itua.	d1ur4a.	d3bvua3.	d2ix0a4.
d2bhua3.	d1h3ga3.	d1ug6a.	d2qeea.	d1lnsa3.	d3kcg1.	d1a6ca3.	d1a6ca2.	d1f15a.	d1y9a1.	d1pd0a2.	d2q4ma1.
d3cnpq.	d1v5ib1.	d1kn6a.	d1h98a.	d1pu1a.	d2iuba2.	d1p0za.	d3by8a.	d2basa2.	d3cptb.	d1ycsb1.	d1dcq2.
d1mb6a.	d1y29a.	d1x17a.	d1m1xb4.	d1mkca.	d2hjq2.	d1gr1a1.	d1gl5a.	d3iqla.	d1v7ja1.	d2hnua.	d1ve9a.
d2ig3a.	d2zs0b.	d1dlwa.	d2bs2b1.	d1nxha.	d1nek.	d2dvt1.	d2f6ka1.	d2i9ua2.	d3igxa.	d1dc1a.	d2wjnm.
d2buda1.	d2f5ka1.	d3m9qa.	d2diga1.	d1kjq1.	d1c1ca2.	d1clca2.	d1jia1.	d2oxga.	d3qzba.	d1slua.	d2qfaa.
d1vca1.	d1zxqa1.	d1fhga.	d1fira.	d2anea1.	d2crs1a.	d1u9ca.	d1qvva.	d1t0ba.	d1ydg.	d1vr0a1.	d1mw7a.
d1ueba1.	d1khia1.	d1vq8q1.	d2dy8a1.	d1r5ba2.	d2c2qa1.	d1se9a.	d1w2ya.	d1wxaa1.	d1l4db.	d2pbdp.	d1i30e.
d1e5da1.	d1oboa.	d2arka1.	d2naca2.	d2phpa1.	d1x0ea.	d1a9xa6.	d1vkza3.	d2r85a2.	d1x9na3.	d1wdua.	d3btaa2.
d1vyxa.	d2o9sa.	d1i07a.	d2eyqa1.	d1s98a.	d1vmga.	d1p5qa1.	d1ihga1.	d2ff4a2.	d1bpoa1.	d1k8ke.	d1q90d.
d3ebta1.	d2geya1.	d1nwwa.	d1iq8a4.	d1jida.	d1nc7a.	d2bcq2.	d1jmsa3.	d3arcu.	d2h80a1.	d3hqa.	d2i4ra1.
d2c0sa1.	d1dv0a.	d1otra.	d1sr9a1.	d1rqt.	d1x4sa1.	d3l18a.	d2rk3a.	d1kwga3.	d1rl1a.	d1tfr2.	d1xs1a.
d1svba1.	d1ok8a1.	d2vya1.	d2q3za3.	d2mev.1.	d3hmsa.	d1qnra.	d1h1na.	d1fcqa.	d3bofa2.	d1xfka.	d1rq0a.

192 APPENDIX A. LIST OF RANDOMLY SELECTED SCOP HEIRARCHY DOMAINS

Pivot	Family	Super Family	Fold	Class	Decoy	Pivot	Family	Super Family	Fold	Class	Decoy
d1nu0a	d1vhxa	d3b2ka5	d1evqa3	d1e0ta3	d2p02a2	d1b10a1	d1d5ya2	d1igna1	d1rq6a	d1kdxa	d2nsfa2
d3clsd1	d1efpa1	d2gm3a1	d3mlaa	d1b15a2	d1r33a1	d1vr3a1	d1zrra1	d1d9wa	d3b2a	d2in5a1	d1n62c2
d3eb2a	d3s5oa	d3oexa	d3bofa1	d1rkqa	d1te5a	d2cv1a	d1pfa5a	d3lmea	d2ohwa1	d2c6ua	didmu.1
d2ewca1	d1qu9a	d1ufya	d1ipaa2	d1wpua	d1c1da2	d3m5va	d3daqa	d3e0ia	d1qwa	d2b3ya1	d1d3ya
d1cnva	d1edta	d2uy2a	d1xg4a	d1wcwa	d1k92a2	d1tuza	d2zkmx1	d1m45a	d1ooha	d1eexg	d1oxwa
d1cpza	d3cjk1	d2xmwa	d2x3da	d3l7ha	d3arcz	d2cyya2	d1li2a	d1q8ba	d1ysja2	d1lnia	d2g38a1
d3dc7a1	d3bzwa1	d1k7ca	d1x38a2	d2x61a	d1xa1a	d2cypa1	d1u6fa1	d3ue2a	d1b3ta	d1e3ha5	d1f3ub
d2vn6a	d3bwza	d3zuca	d1p7h12	d2fr2a1	d1ohta	d3bmva4	d1r7aa2	d1vffa1	d1d8wa	d1y9za	d1qfma1
d1d4ba	d1f2ri	d2npta1	d2bvca1	d1e3ha3	d2j01u1	d1o4ua2	d2f7fa2	d1yira2	d2tpta3	d2evra2	d1gaka
d1qhoa1	d1h3ga1	d1im3d	d1l0qa1	d1h6wa1	d1u97a	d2fbl1a1	d2fmla1	d1t6xa	d1j1va	d1n62a1	d1u2ma
d3elna	d2gm6a1	d3l0ia	d2arca	d1o75a3	d1ulya	d2ifga1	d3b5ha2	d1u2ha	d1vzia1	d1ei5a1	d2ftxb1
d2d7pa1	d1v05a	d2v33a	d1xaka	d1e44b	d1gz0a2	d2y5ca	d3lxf6a	d2bt6a	d1i17a	d1m45a	d1m2da
d1d8ba	d2elfa	d2hbj1a	d1jmsa1	d1tuka	d2fgga1	d1vc4a	d1qopa	d1thfd	d1v93a	d1o98a2	d1vjua
d1t33a1	d3c07a1	d1k78a2	d2uubr1	d3rnm	d1vj0a	d1dr9a2	d2nxyb2	d1ogad2	d1c9va4	d1r8o.1	d1qoq1
d1w3ca	d1azsa	d2v0na3	d1mg7a2	d2atza1	d1kmza	d2je8a5	d7a3ha	d1h19a2	d5ruba1	d1y8aa1	d2i06a
d1d1da1	d1a8oa	d3lhra	d2gyka1	d1x3zb	d2dewx2	d2r1rfg	d2pi0b	d1xb4a1	d2cxya	d2pp4a1	d1oedc
d1ycsb2	d3i35a	d1w6xa	d1dfup	d1jb0e	d3cx5h	d2ix0a2	d2asba1	d1ckma1	d2qw7a1	d1b35.1	d1kve.1
d2esh1a	d1xmaa	d2a5yb1	d1jnga	d1v64a	d1yp2a1	d3cg4a	d3eqza	d1dcfa	d1oi7a2	d2q4qa	d1dk8a
d3f5da	d3duka	d2owpa1	d2ch9a	d1e71a2	d1oeda	d2pd1a1	d1y0ha	d3hf5a	d2czvc1	d2g30a2	d1c88a2
d1brza	d1jxca	d2i61a	d2f78a1	d4mt2a	d1a79a1	d1qkka	d2pl1a	d1c4ka1	d2csua2	d2j01p1	d2jek1
d1pdkb	d2axwa1	d2okma	d1leaqa	d1ca1a2	d1n6za	d3l0ca1	d1t56a1	d1g2ha	d2htja1	d1st6a6	d1ezga
d3igs1a	d3ct1a	d1qo2a	d1twda	d1tubb1	d1b12a	d2qrra1	d2qswa1	d1zvp2a	d2hiqa	d2jxta1	d2za4b
d2j73a	d2j43a2	d3bmva2	d2g2na	d1xrt1a	d1d0qa	d2qpv1a	d3f08a	d2ffsa1	d2pwwa1	d1wf2a	d1o6da
d1nqqa1	d2y6wa	d3kwoa	d1noga	d1dvka	d2f8ta	d1gkra2	d2ftwa2	d2z26a	d1nqka	d3nioa	d1mkia
d1oi1a2	d1w9a	d1h3za	d2wgsa	d1bhua	d1p26a	d1j26a	d1xvwa	d1nepa	d1v8ha1	d1l6pa	d2o0qa1
d1ygya3	d1sc6a3	d2f06a2	d1wja1	d2dira1	d1ppj1a	d1x4ha1	d3mdfa	d1uw4a	d1aopa1	d2fma	d1fft2
d2nuha	d1ukua	d4affa	d2v5a2	d1vlra2	d1j0pa	d1o7ia	d1iyj3	d1x9na2	d4dffa	d2wraa	d2r1da1
d1x4a1	d3h2ve	d3md1a	d2ewha1	d2ccqa1	d1h4vb1	d3l03a	d2gffa	d1ml1a	d1iwga4	d1vqza1	d2j01s1
d1qlb	d1w18a1	d1z10a2	d1reqb2	d2pju1a1	d1vq8m1	d1zhva2	d1zhva1	d2nzca1	d1k6pa	d1dzfa2	d1g3pa1
d2cqka1	d1zh5a1	d1s61a1	d1a04a1	d1t1ha	d3d3ra1	d2fx0a1	d1rkt1a1	d1iufa2	d1ku3a	d2o31a1	d1ny8a
d1nfga2	d1gkpa2	d2puza2	d1ht6a2	d1o0sa1	d1v0ta	d1a04a2	d1p6qa	d3heba	d1e1wa	d1e8ca1	d2g45a
d1pnha	d1px9a	d1gpsa	d1clvi	d2i9wa2	d1n13.1	d1mxa2	d1gcya2	d3n9ka	d1zfa1a	d1gy8a	d2o3ia1
d2cqva1	d1iray2	d2hp4a	d1rowa	d1ni5a3	d1m37a	d3l21a	d2wkja	d3e96a	d2d69a1	d3m7va1	d2p9a1
d1x6oa1	d1iz6a1	d2j0ya1	d1c0ma1	d1e0ta1	d3ccda	d1k36a	d1dx5i2	d1b9wa1	d1ehda2	d1ansa	d2bcg2
d1k68a	d1u0sy	d3a0ua	d1fyxa	d2fy8a1	d1seia	d3s33a	d2yyua	d2y88a	d1jpm1a	d2i9ia1	d1h5wa
d2h26a1	d1hdmb1	d1ccza1	d1cwwa1	d2nwt1a1	d1j0ka	d1nara	d2ebna	d1tvna1	d1zcca	d1o0sa2	d1uynx
d1jcf2a	d2zgya1	d1u6za3	d1vq8n1	d1d7ya1	d1gm5a2	d1bdf1a	d1twwc1	d1twwc1	d1zq1c2	d1by2a	d1j55a
d2b4aa1	d3q15c	d2ayxa2	d1ccwa	d2iel1a1	d1c26a	d2dk8a1	d2dk5a1	d2p6ra1	d2p6ra1	d1r3pa2	d1x9aa
d1hxm1a	d1tjgh1	d1cs6a2	d1goda3	d1bw3a	d2o3bb	d2bhma1	d2cc3a1	d3cnxa1	d3gaxa	d1jrma	d2j8ga1
d1ty0a1	d1et9a1	d1qb5d	d1yeza1	d2jn4a1	d2a6aa2	d1txka1	d1wz1a1	d3c8da1	d1bhga1	d1whga	d2nt2a
d1qtwa	d3aala	d1k77a	d2bas1a1	d1wiwa	d1jw7a2	d1jmc2a	d3ull1a	d2z01a	d3ma2b	d2z0ta	d1whra
d2nv0a	d1i7qb	d2vrna	d2csua3	d1gtea3	d2oqma1	d1edm	d1moxc	d1b9wa2	d1l1pa1	d1e8pa	d1ppj1
d1g2914	d1g2913	d1fr3a	d1uwva1	d1k8ma	d2oqfa1	d1fu3a	d1glpa	d1l1pa	d1deca	d2d81a	d2vt1.1
d1exra	d2mysb	d3nxaa	d1d9ea	d1px5a1	d2gog1	d3qu3a	d3f8ba	d1ka8a	d2d4wa1	d1s4ka	d2nwva1
d2f4pa1	d1o4ta	d2bmma2	d1h9db	d1r5sa	d1h9db	d1z05a2	d2gupa2	d2e12a1	d2v32a1	d2foka4	d1ydua1
d1e6ia	d1eqfa1	d2grca	d2gsc1a1	d1vq8p1	d2g2jva1	d2rfrba	d1p1ja2	d1y17a2	d1xaaa2	d2dm9a1	d1rf8b
d2qs8a2	d3mkva2	d2ob3a	d1lt7a	d1g6oa	d2rfa	d1hyrc1	d1dm0b2	d1dn0b2	d1b1ha3	d1b4ra	d3arce
d1hsta	d1ussa	d1mkma1	d1z1ja	d2pqra	d1vkr1a	d1guwa	d1g6za	d3r93a	d1ugva	d3r9aa	d1ng7a
d1esfa1	d3seba1	d1prtd	d1orxx1	d1alxa	d1q8ca	d1pgl21	d1ny711	d1ng0a	d1m3ya1	d2ag4a	d1jb01
d1ur1a	d1w91a2	d2cksa	d1vhna	d2r8oa2	d1v0ta	d1vtxa1	d1itxa1	d1vj2a	d2e6fa	d2x06a	d2ajaa1
d2a1jb1	d1x2ia1	d1kfta	d1z3eb1	d1or7c	d1qe0a1	d1gl4b	d1ol2a1	d1k5na1	d2b51b1	d1wv3a1	d2ae8a1
d1u78a2	d2ezia	d1k78a1	d1vq8i1	d1f6va	d1m1ha1	d1rjja	d1tr0a	d1vqsa	d1q8ka2	d1o1ka1	d1r2ja1
d1vtxa	d1d10a	d1p8ba	d1q9ba	d1shyb2	d1h8d.1	d1x5sa1	d1ufwa	d1koha2	d1w9ja1	d2v94a1	d1yk4a
d1q6a	d2czda	d1tqja	d1xm3a	d1h6da1	d1yu0a2	d1l3ka1	d1x4ea1	d1jmta	d1rwua	d1ghha	d1pn5a1
d1l9na1	d1g0da1	d1ix2a	d1xq4a	d2bv4a	d2cfa1	d1skra	d2fqa1	d1etxa	d3bwga1	d2jrx1a	d3arcb
d1jvna2	d2abwa1	d2gk3a1	d2fzva1	d1bifa2	d1qwa	d3grda	d3fgya	d1tuha	d2oqa2	d1z0ca1	d1a3wa3
d2crya1	d1mjul1	d1iam1a1	d1o75a1	d1fnta1	d1p26a	d2od6a1	d2op5a1	d1s7ia	d1m1ha2	d1u2ca2	d1yksa1
d1nkra2	d1glca	d2yxma	d1o75a2	d1e5ba	d1o8ba1	d1e43a2	d2guya2	d2nt0a2	d1jr1a1	d2dul1a1	d1oj7a
d2qam01	d2j0151	d1vq8z1	d1jnea1	d1wffa	d2gtvx1	d2cdqa2	d2cdqa3	d1phza1	d1zpxw1	d1mhm.1	d1ts9a
d1ug1a	d1k4us	d1yn8a	d3dkma	d1f56a	d1vk3a4	d1hxb2	d1mjul2	d1qfoa	d1tyeb1	d1r75a	d2q37a1
d1yyva1	d1z7ua1	d2foka2	d1mi1a	d2crua1	d1g3pa2	d3b7ca1	d3bb9a1	d2k54a1	d1w6ga3	d1ayaa	d2i15a1
d1v70a	d1yhfa1	d2pyta1	d1cx4a1	d1t2wa	d2gwf1a1	d1to3a	d1o1ja	d1nvma2	d1vpya	d2vbaa1	d1t5ra
d1dgsa2	d1fvia1	d2ccqa1	d1nxxa	d2j97a	d2gyrb2	d1hdma1	d1o0va1	d3oska	d3qzma	d1njha	d1mo9a2
d1sxea	d1sxd1a	d3bq7a	d3ci0k1	d2gf5a2	d1i0a	d1jgsa	d2hr3a1	d1v4ra1	d2gla1a1	d1hy5a	d2hgsa2
d1hkqa	d2nrac1	d1fnna1	d1xsva	d1ngna	d2hh8a1	d1wx8a1	d2zeqa	d1wspa1	d1jq4a	d2ba0g2	d1f53a
d1oj4a2	d1ueka2	d1k47a2	d1n0ua5	d3m7va2	d2f31a1	d3brda1	d1my7a	d3es6b1	d2vzsa3	d1ik9a1	d1zyna1
d2f1fa1	d2fgca1	d2hmf3a3	d2bopa	d2dpia1	d3d9ta1	d1xwya1	d1j6oa	d1bf6a	d2zra2	d1omza	d2aioa
d1m9sa2	d1m9sa3	d2akka1	d1whma	d1n7oa2	d2sp1a1	d1acwa	d1r1ga	d1icaa	d1hy9a	d1d6ga	d1jmu.1
d2q1ya	d1gy7a	d3ef8a1	d1nnpva	d2elba2	d2fi9a1	d2j3sa2	d2w0pa	d1jmx3a	d1iarb2	d2iaa	d2g1da1
d1gvea	d3eaa	d3n2ta	d1ceoa	d1p5ja	d1u60a	d1iyj5	d1jb7a2	d1a01a1	d1uapa	d2de6a1	d1qh4a1
d2yr1a	d3b4ua	d1gvfa	d1vp5a	d2fywa1	d2bona1						
d1pgl22	d1a6ca1	d1x36a	d3rqva	d1i0ra	d1uxya1						
d2a15a1	d1ohpa	d3blza1	d2bo9b2	d2r7ja2	d1kke2						
d1m94a	d1wgda	d1gg3a3	d1bmlc3	d2a6qa1	d1hs7a						
d1xn5a	d3elia1	d3ijta	d1j3ma	d3reaa	d1x14a2						
d1oywa1	d2axla1	d1pp7u	d1kxxa	d1ej5a	d1q3qa3						
d1x14a1	d1n9pa	d1ex0a1	d2aqma	d1h7za	d1xa6a1						
d1rwsa	d1ryja	d2qjla	d2piaa3	d1x3xa	d1pzra						
d1u8sa1	d1u8sa2	d1y7pa2	d1gh8a	d1hc7a3	d1kke1a						
d1he7a	d3b5ha1	d1zoxa	d2mcca	d1y7ma1	d1v2za						
d3broa1	d2bv6a1	d1u2wa1	d1ig6a	d1wh8a	d1c3ma						
d1ug3a2	d2nsza1	d1r24a2	d3ag3e	d1sbxa	d2clyb1						
d2z3va	d1q77a	d1vbk1a1	d1m8a3	d2a8na1	d1nofa1						
d1av3a	d1rmka	d1d1ha	d1fle1	d1gli1	d1ctaa						
d1ixra1	d1cuka2	d2duya1	d1v38a	d1ofcx3	d1rc9a2						
d1hmb2	d1vcl1a1	d1upsa2	d1drc1a3	d2blna1	d1cwa5						

Appendix B

Creative Commons Attribution-NoDerivatives 4.0 International



<https://creativecommons.org/licenses/by-nd/4.0/>

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-NoDerivatives 4.0 International Public License (“Public License”). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

- a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.
- b. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.
- c. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

- d. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.
- e. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.
- f. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.
- g. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.
- h. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.
- i. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.
- j. **You** means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:
 - A. reproduce and Share the Licensed Material, in whole or in part; and
 - B. produce and reproduce, but not Share, Adapted Material.
2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.
3. Term. The term of this Public License is specified in Section 6(a).
4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.
 - A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.
 - B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.
6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.
2. Patent and trademark rights are not licensed under this Public License.
3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material, You must:
 - A. retain the following if it is supplied by the Licensor with the Licensed Material:
 - i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);
 - ii. a copyright notice;
 - iii. a notice that refers to this Public License;
 - iv. a notice that refers to the disclaimer of warranties;
 - v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
 - B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and
 - C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

For the avoidance of doubt, You do not have permission under this Public License to Share Adapted Material.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

- a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database, provided You do not Share Adapted Material;
- b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and
- c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

- a. **Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.**
- b. **To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.**
- c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

- a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.
- b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:
 1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
 2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.
- c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.
- d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

- a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.
- b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

- a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.
- b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.
- c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.
- d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.