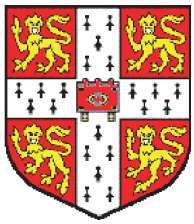
Computational Analysis of Protein Function within Complete Genomes

Anton James Enright

Wolfson College



A dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

European Molecular Biology Laboratory,
European Bioinformatics Institute,
Wellcome Trust Genome Campus,
Hinxton, Cambridge, CB10 1SD,
United Kingdom.

Email: enright@ebi.ac.uk

March 7, 2002

To My Parents
and
Kerstin

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

This thesis does not exceed the specified length limit of 300 pages as defined by the Biology Degree Committee.

This thesis has been typeset in 12pt font using LaTeX2ε according to the specifications defined by the Board of Graduate Studies and the Biology Degree Committee.

Computational Analysis of Protein Function within Complete Genomes

Summary

Anton James Enright
Wolfson College

March 7, 2002

Since the advent of complete genome sequencing, vast amounts of nucleotide and amino acid sequence data have been produced. These data need to be effectively analysed and verified so that they may be used for biological discovery. A significant proportion of predicted protein sequences from these complete genomes have poorly characterised or unknown functional annotations. This thesis describes a number of approaches which detail the computational analysis of amino acid sequences for the prediction and analysis of protein function within complete genomes. The first chapter is a short introduction to computational genome analysis while the second and third chapters describe how groups of related protein sequences (termed *protein families*) may be characterised using sequence clustering algorithms. Two novel and complementary sequence clustering algorithms will be presented together with details of how protein family information can be used to detect and describe the functions of proteins in complete genomes. Further research is described which uses this protein family information to analyse the molecular evolution of proteins within complete genomes. Recent developments in genome analysis have shown that the computational prediction of protein function in complete genomes is not limited to pure sequence homology methods. So called *genome context* methods use other information from complete genome sequences to predict protein function. Examples of this include gene location or neighbourhood analysis and the analysis of the phyletic distribution of protein sequences. In chapter four a novel method for predicting whether two proteins are functionally associated or physically interact is described. This method is based on the detection of gene-fusion events within complete genomes. During this research many novel tools and methods for genome analysis, data-visualisation, data-mining and high-performance biological computing were developed. Many of these tools represent interesting research projects in their own right, and formed the basis from which the research carried out in this thesis was conducted. Within the final chapter a selection of these novel algorithms developed during this research is described.

Preface

This thesis describes work carried out at the European Bioinformatics Institute (EBI) in Cambridge, UK, between October 1998 and February 2002. The EBI is an outstation of the European Molecular Biology Laboratory (EMBL). I was the recipient of an EMBL predoctoral fellowship to work with Dr. Christos Ouzounis in the Computational Genomics Group.

There are many individuals who have provided assistance and encouragement throughout this work. I would first like to thank Professor Kenneth Wolfe and Professor David McConnell at the Smurfit Institute of Genetics in Trinity College Dublin, where I had previously been an undergraduate. While at Trinity I was given the best scientific education one could hope for, spanning classical genetics, mathematics and bioinformatics. I was encouraged at Trinity to accept an internship with Digital Equipment Corporation in my final year. It was at Digital that I discovered how bad my programming skills actually were, and through the patient efforts of Dr. Eamonn O'Toole managed to rectify the situation somewhat.

There are many people at the European Bioinformatics Institute who through their friendship and support have made this work an enjoyable and rewarding experience. Firstly, my supervisor Dr. Christos Ouzounis, whose patient support and scientific knowledge have greatly aided this research. The other members of the computational genomics group have helped me over the years in countless ways and have also had the dubious pleasure of discovering all of the bugs in software I have written. James Cuff and Michele Clamp helped me get on my feet as a bioinformatician by patiently providing advice and assistance on an almost daily basis. I'd also like to thank Steven Searle, who has helped me debug countless programs, and whose sage-like wisdom of all things computational still terrifies me. Stijn Van Dongen, the inventor of the MCL algorithm, deserves special mention for writing such a beautiful algorithm, and for helping me understand how it works. Finally, and not least, the members of the EBI 'A-Team', especially Kerstin Dose, who keep the EBI running and who have sorted out countless problems for me.

Contents

Summary iii

Preface iv

1 Introduction 1
- 1.1 Protein Sequence Alignment and Similarity Detection . . . 3
  - 1.1.1 The Needleman-Wunsch Algorithm . . . 4
  - 1.1.2 The Smith-Waterman Algorithm . . . 8
  - 1.1.3 The FASTA Algorithm . . . 9
  - 1.1.4 The BLAST Algorithm . . . 12
  - 1.1.5 Statistics for Sequence Alignment Searching . . . 14
  - 1.1.6 Profile Searching: PSI-BLAST and HMMER . . . 22
- 1.2 Computational Biology Data Sources . . . 26
  - 1.2.1 Nucleotide Sequence Databases . . . 26
  - 1.2.2 Protein Sequence Databases . . . 29
  - 1.2.3 Protein Domain and Family Databases (Pfam & InterPro) . . . 31

2 Algorithms for Automatic Protein Sequence Clustering 33
- 2.1 Genomic Protein Family Analysis . . . 35
- 2.2 Protein Sequence Clustering Techniques . . . 37
  - 2.2.1 Multi-Domain Proteins . . . 38
  - 2.2.2 Orthology, Paralogy and Evolution . . . 40
  - 2.2.3 Automation and Scaling . . . 42
  - 2.2.4 Previous Methods . . . 44
- 2.3 GeneRAGE . . . 49
  - 2.3.1 Initial Steps . . . 49
  - 2.3.2 Symmetrification of the Matrix . . . 51
  - 2.3.3 Detection of Multi-Domain Proteins . . . 52

2.3.4 Clustering the Similarity Matrix . . . 53
2.3.5 Validation and Testing . . . 55
2.3.6 Algorithm Implementation . . . 58
2.3.7 Conclusions . . . 58
2.4 Tribe-MCL . . . 60
2.4.1 Introduction . . . 60
2.4.2 Markov Clustering of Sequence Similarities . . . 62
2.4.3 Application of the MCL Algorithm to Biological Graphs 67
2.4.4 Validation of the Algorithm . . . 70
2.4.5 Large-Scale Family Detection . . . 72
2.4.6 Algorithm Implementation and Availability . . . 75
2.4.7 Conclusions . . . 75

3 Analysis of Protein Families 77
3.1 Exploration of Protein Families using GeneRAGE . . . 78
3.1.1 Detection of Novel Archaeal Domains . . . 78
3.1.2 Transcription Associated Protein Family Analysis . . . 84
3.1.3 Detection of SR-Domain Proteins . . . 89
3.2 Large-Scale Detection of Protein Families using Tribe-MCL . . 93
3.2.1 Family Annotation of the Draft Human Genome . . . 93
3.3 Tribes: A Database of Protein Families . . . 99
3.3.1 A Complete Genomes Database . . . 100
3.3.2 Construction of the Tribes Database . . . 103
3.3.3 Protein Family Analysis using the Tribes Database . . 106

4 Genomic Analysis of Protein Interaction 125
4.1 Computational Detection of Protein-Protein Interactions . . . 126
4.1.1 Structural Biology Approaches . . . 126
4.1.2 Sequence and Genomic Context Approaches . . . 127
4.2 Prediction of Protein Function and Protein-Protein interaction using Gene Fusion events . . . 131
4.2.1 An Algorithm for the Detection of Gene Fusion Events 134
4.2.2 Application and Validation of the Algorithm using Four Complete Genomes . . . 137
4.3 Exhaustive Detection of Protein-Protein Interactions . . . 143

4.3.1 Computational Protocol for Exhaustive Detection of Gene Fusion Events . . . 145
4.3.2 Results . . . 148
4.3.3 Data Storage & Availability . . . 159
4.3.4 Discussion . . . 159

5 Methods for Genome Analysis and Annotation 164
5.1 CAST: Detection of Compositionally Biased Regions in Protein Sequences . . . 165
5.1.1 Introduction . . . 165
5.1.2 Algorithm Concept . . . 166
5.1.3 Detection of Low-Complexity Residues . . . 168
5.1.4 Filtering Procedure . . . 169
5.1.5 Implementation . . . 169
5.2 BioLayout: A Visualisation Algorithm for Protein Sequence Similarities . . . 173
5.2.1 Graph Layout Theory . . . 173
5.2.2 Graph Layout Algorithms . . . 174
5.2.3 The BioLayout Algorithm . . . 175
5.2.4 Implementation and Usage . . . 177
5.3 TextQuest: Automatic Classification of Medline Abstracts . . 183
5.3.1 Introduction . . . 183
5.3.2 Text-Mining Approaches . . . 184
5.3.3 The TextQuest Algorithm . . . 185
5.3.4 Document Clustering Results . . . 188
5.3.5 Implementation . . . 191
5.4 Consensus Annotation of Protein Families . . . 192
5.4.1 Formulation of the Problem . . . 192
5.4.2 Longest Common Substring (LCS) Detection . . . 194
5.4.3 An LCS Algorithm for Automatic Consensus Annotation . . . 194
5.5 High-Throughput Tools for Sequence Analysis . . . 196
5.5.1 htBLAST.pl and Parse.pl . . . 196
5.5.2 Parallel.pl . . . 198
5.5.3 lsfBlast.pl . . . 200

Conclusions 202

A Papers Published During this Work 204

B CGD and Tribes Database Schema **206**

Bibliography **220**

List of Tables

1.1 Algorithms for Sequence Alignment 4
1.2 Computational Biology Databases 27

2.1 Comparison of Different Clustering Methods 48
2.2 Promiscuous Domains is SwissProt 73

3.1 Complete Archaeal Genomes 79
3.2 Novel Domains Discovered using GeneRAGE 81
3.3 Distribution of TAP Families across Domains 86
3.4 Largest Predicted Human Families 97
3.5 Tribes: Family Statistics 106
3.6 Tribes: Family Counts 107
3.7 Tribes: Family Distribution 108
3.8 Tribes: Domain Specific Families 111
3.9 Universal Protein Families in Tribes 118
3.10 Novel Universal Protein Families 119

4.1 Summary of Detected Gene Fusions 139
4.2 The Consensus Annotation Problem 144

5.1 BioLayout Input Format 177
5.2 TextQuest Term Reduction 189
5.3 TextQuest Experiment Terms 189
5.4 The Consensus Annotation Problem 193
5.5 High-Throughput Sequence Analysis Tools 197

List of Figures

1.1 The Needleman-Wunsch Algorithm 7
1.2 Needleman-Wunsch Dynamic Programming Pseudocode 9
1.3 Smith-Waterman Dynamic Programming Pseudocode 10
1.4 Local and Global Alignment 11
1.5 Overview of the BLAST Algorithm 15
1.6 The Extreme Value Distribution 18
1.7 Example Annotated BLAST Output 19
1.8 The BLOSUM62 Matrix 21
1.9 Profile HMM Diagram 24

2.1 Stammbaum des Menschen 34
2.2 Multi-Domain Proteins: Alignment Schematic 39
2.3 Multi-Domain Proteins: Graph Schematic 40
2.4 Gene Duplication and Speciation 41
2.5 Growth in Public Sequence Databases 43
2.6 GeneRAGE Algorithm Flowchart 50
2.7 GeneRAGE Multi-Domain Detection 54
2.8 Protein Family Distribution in *M. jannaschii* 56
2.9 Organisation of the Bacterial *aro* Gene Cluster 57
2.10 Flowchart of the Tribe-MCL Algorithm. 63
2.11 Tribe-MCL Markov Matrix Data Representation 68
2.12 SwissProt Inter-Family Relationships 74

3.1 Archaeal FTR Domain Alignment (distal lobe) 82
3.2 Alignment of an Archaeal Domain of Unknown Function . . . 83
3.3 TAP Family Distribution 87
3.4 RS Domain Detection Strategy 91
3.5 Human Genome Family Annotation Pipeline 95
3.6 Ensembl Protein Family Size Distribution 96
3.7 TFIIB Protein Family Sequence Alignment 98
3.8 Tribes Database Pipeline 104
3.9 Tribes Family Size Distributions 109

3.10 Phylogenetic Distribution of Tribes Protein Families 112
3.11 Distribution of Strain and Species Specific Proteins 114
3.12 Tribes Entry Page 120
3.13 Tribes Search Page 121
3.14 Example Tribes Search Page 121
3.15 Tribes Family View Page 122
3.16 Tribes Species Specificity Page 123
3.17 Tribes Genome View and Protein View pages 124

4.1 Phylogenetic Profile Analysis 128
4.2 Gene Co-localisation and Gene Fusion Analysis 130
4.3 3-D Structure of an *E. coli* Fusion Protein 132
4.4 Alignment of TrpC and TrpF Fusion Proteins 133
4.5 Flowchart of the DifFuse Algorithm 136
4.6 Sensitivity and Specificity of the DifFuse Algorithm 137
4.7 Gene Co-Localisation and Gene Fusion Analysis 140
4.8 Protocol for the Exhaustive Detection of Gene Fusion Events . 146
4.9 3-D Structures of Composite Proteins 1 149
4.10 3-D Structures of Composite Proteins 2 150
4.11 Z-Score Distribution for Component Proteins 151
4.12 Correlation of Gene Expression between Component Pairs . . 154
4.13 Absolute and Relative numbers of Component and Composite Proteins 156
4.14 Phylogenetic Reconstruction based on Fusion Events 160
4.15 Screenshot of the AllFuse Web-Server 162
4.16 Fusion Alignment from the AllFuse Web-Server 163

5.1 Flowchart of the CAST Algorithm 170
5.2 Screenshot of the CAST Web-Server 172
5.3 BioLayout Pseudocode 178
5.4 The BioLayout Graphical User Interface 179
5.5 PostScript Output from BioLayout 181
5.6 BioLayout 3-D Example 182

B.1 ER Diagram for CGD and Tribes 208
B.2 MySQL table definition for the Genomes table 208
B.3 MySQL table definition for the Proteins table 209
B.4 MySQL table definition for the Swiss_Proteins table 209
B.5 MySQL table definition for the Feature table 209
B.6 MySQL table definition for the Family_Members table 210
B.7 MySQL table definition for the Families table 210

Chapter 1

Introduction

Bioinformatics is by definition a multi-disciplinary science. It is this aspect that makes the field immensely appealing. Conducting research at the interface between molecular biology, mathematics and information technology is a challenging undertaking. Enormous rewards are possible however, if these fields are successfully used to analyse fundamental biological questions.

The discovery that the primary amino acid sequence of a protein can determine both its biochemical function and three-dimensional structure (Anfinsen and Corley, 1969; Anfinsen, 1973), was instrumental in the birth of computational biology. The primary sequence of a protein can hence be used to predict its structural and functional properties. Furthermore, the observation that biological sequences evolve at measurable and relatively constant rates, permitted the analysis of molecular evolution at the sequence level (Zuckerkandl and Pauling, 1962; Zuckerkandl et al., 1965). Peptide and nucleotide sequencing (Brown et al., 1955; Sanger and Coulson, 1975; Sanger, 1981) technologies have advanced rapidly, and have allowed the complete genome sequencing of free living organisms (Fleischmann et al., 1995). These breakthroughs have placed us in a position which allows computational analysis of gene and protein function on a genomic scale.

The work described in this thesis can be described as *functional genomics*. The goal of this field is to determine the function of genes and proteins in complete genomes. The term 'function' must be used with caution. A single gene may have metabolic, regulatory and structural functions. The function of a gene can not in most cases be described completely by any one of these roles, but is the sum of its separate sub-functions. Issues such as this are rapidly being addressed by efforts such as the *Gene Ontology* project

(Ashburner et al., 2000).

This research concentrates on the development of novel methods for the prediction and description of protein function within complete genomes. More specifically, we concentrate on using the primary amino acid sequences of proteins from complete genomes to achieve this goal. While it may be desirable to compare proteins at a structural level, the quantity of publicly available structural information needed for large-scale analyses of this kind, is unfortunately not yet available. We determine functional relationships between proteins using two methods. The first method involves the classification of proteins into groups with a common evolutionary history. These groups are generally known as *protein families*. In many cases such proteins will also exhibit conserved functional roles (Dayhoff, 1976). The second area involves the cross comparison of complete genomes in order to detect evolutionary events which may indicate that proteins interact, or are functionally associated. This type of analysis uses the context of a gene in a complete genome in order to discover functional information, and has been termed *contextual genomics* (Enright and Ouzounis, 2001c).

This chapter is intended to serve as a gentle introduction to the field of sequence alignment, which has created the tools and expertise necessary for much of the work described in this thesis. Because sequence analysis methods are greatly dependent on the quality of quantity of biological sequence data available, this introduction will also describe key sources for biological data. Each subsequent chapter will contain a more detailed introduction to the research described therein. We hope that the work described within this thesis will prove interesting and useful for biological discovery.

1.1 Protein Sequence Alignment and Similarity Detection

The primary data for most analyses described within this thesis is protein similarity information. This information is derived by comparing the amino acid sequences of proteins and can be used to infer functional and evolutionary links between them. In order to compare two protein or nucleotide sequences, one needs to determine the locations of where insertions or deletions may have occurred in each sequence. This problem is commonly known as *sequence alignment*. Alignments are generally represented in a manner similar to this:

```
Sequence 1: ATTTTGCCCTTTATGCCGT-ATTAT
            |||| |||||  ||||||| |  ||
Sequence 2: ATTT-GCCCT--ATGCCGTTACGAT
```

In this case two DNA sequences are aligned with matches/mismatches (shown by the presence or absence of a '| ') and gaps (shown by '- ') which represent insertions or deletions (commonly called *indels*) in either sequence. There may however be many possible alignments of two sequences, for example here our previous example is compared with two other possible alignments.

```
Sequence 1: ATTTTGCCCTTTATGCCGT-ATTAT
            |||| |||||  ||||||| |  ||
Sequence 2: ATTT-GCCCT--ATGCCGTTACGAT

19 matches, 2 mismatches, 4 gaps

Sequence 1: ATTTTGCCCTTTATGCCGT-A-TTAT
            | ||||||| | ||||||| |   ||
Sequence 2: A-TTTGCCC-T-ATGCCGTTAC-GAT

19 matches, 1 mismatch, 6 gaps.

Sequence 1: ATTTTGCCCTTTATGCCGTATTA---T
            ||||  ||||  ||||||  |||   |
Sequence 2: ATTTG-CCCT--ATGCCG--TTACGAT

18 matches, 1 mismatch, 8 gaps
```

Algorithm	Alignment Type	Scoring Matrix	Gap Penalty	Time Required	References
Needleman-Wunsch	global similarity	arbitrary	penalty/gap k	$\Theta(n^2)$	(Needleman and Wunsch, 1970)
Sellers	(global) distance	unity	penalty/residue rk	$\Theta(n^2)$	(Sellers, 1974)
Smith-Waterman	local similarity	$S_{i,j} < 0.0$	affine $q + rk$	$\Theta(n^2)$	(Smith and Waterman, 1981; Gotoh, 1982)
FASTA	approx. local similarity	$S_{i,j} < 0.0$	limited gap size $q + rk$	$\Theta(n^2)/K$	(Lipman and Pearson, 1985; Pearson and Lipman, 1988)
BLASTP	maximum segment score	$S_{i,j} < 0.0$	multiple segments	$\Theta(n^2)/K$	(Altschul et al., 1990)

Table 1.1: Algorithms for comparing protein and nucleotide sequences.

The problem of accurate sequence alignment amounts to finding an alignment of two sequences with the maximum number of matches while minimising the number of gaps and mismatches. The first of our three alignment examples above has the maximum number of matches, and a smaller number of gaps and mismatches than the others, and is hence the better alignment. Sequence alignment of this kind is an important problem in biological sequence analysis, and has been tackled by a number of varied approaches (shown in Table 1.1).

1.1.1 The Needleman-Wunsch Algorithm

The first commonly used approach is that of Needleman and Wunsch (Needleman and Wunsch, 1970). This approach uses an alignment scoring system which assigns a penalty to gaps linearly according to their length. For any alignment its score S can be calculated by:

$$S = x - \sum w_k z_k$$

In this scoring system the score is calculated by counting the total number of matches x and subtracting from this value the product of the number of k-length gaps (z_k) and an arbitrary gap penalty w_k for a gap of length k. The linear gap penalty w_k can be defined as $w_k = a + bk$ where a and b are arbitrary penalties for gap opening and gap extension. An optimal alignment

between two sequences is hence calculated by finding the alignment between two sequences which maximises the score S.

$$S = Max(x - \sum w_k z_k)$$

An alternative way of looking at this problem is to calculate the minimum number of edits that need to be made to two sequences in order to produce an alignment. The key question is of course, how does one computationally and efficiently detect this optimal alignment between two sequences ? The easiest way to represent all possible alignments between two arbitrary protein or nucleotide sequences is as a path graph. Edges on this graph represent matches, mismatches and gaps, while vertices indicate an alignment state ending at that point. An example alignment path graph is shown in box 1 of Figure 1.1. This path graph shows all possible alignment paths between the sequences `GAATTC` (top) and `GGATC` (left). Diagonal alignment paths represent match and mismatch alignments, while horizontal and vertical paths represent a gap in either the top or left sequence. Any alignment state in the graph has been reached through a series of matches, mismatches or gaps.

Dynamic programming methods lend themselves perfectly to this type of optimisation. From a dynamic programming perspective each state in the graph can be scored as follows, using the Needleman-Wunsch scoring scheme:

$$M_{i,j} = Max \begin{cases} M_{i-1,j-1} + S_{i,j} \\ M_{i,j-1} - w \\ M_{i-1,j} - w \end{cases}$$

Each state (node) in the path graph $M_{i,j}$ is assigned a score based on the scores of its incoming states which are connected in the path graph. The value $M_{i-1,j-1}$ represents a diagonal match/mismatch state while the $M_{i,j-1}$ and $M_{i-1,j-1}$ values represent the insertion of gaps in either sequence. Because not all paths are equally desirable, we add the value $S_{i,j}$ to match-mismatch paths, this value is a score for a match between the amino acids or nucleotides represented by i and j. Conversely paths which introduce a gap, are penalised by subtracting the value w. Given these rules it is relatively straightforward to use dynamic programming to assign a score to each state within the graph, based on the highest scoring alignment path ending at that point.

In order to illustrate the dynamic programming Needleman-Wunsch approach we shall use very simple rules. Match paths will be scored by adding

1 to the alignment score, while paths ending in a gap or mismatch will receive nothing.

$$M_{i,j} = Max \begin{cases} M_{i-1,j-1} + S_{i,j}(1 \text{ or } 0) \\ M_{i,j-1} \\ M_{i-1,j} \end{cases}$$

The dynamic programming solution is shown in Figures 1.1 & 1.2 and can be described as follows:

1. The first row ($M_{0,j}$) and first column ($M_{i,0}$) of the graph automatically have zero values assigned (Figure 1.1 box 1). We start at $M_{1,1}$ and choose the path which maximises the value at that point. The two gapped paths (horizontal and vertical) will maintain the score at zero, representing starting the alignment with a gap in either sequence. The match state will give a score of 1, representing the alignment starting with a G/G match. The score is obtained by taking the incoming score $M_{0,0} = 0$ and adding to it the score for a G/G match ($S_{1,1} = 1$), hence the score obtained is $0 + 1 = 1$. In this case we shall choose this match path as it gives the highest score, and assign the score 1 to $M_{i,j}$ (Figure 1.1 box 2).

2. We continue the first dynamic programming round by assigning values in a similar fashion to each state in $M_{1,j}$ and $M_{i,1}$. In the case shown in Figure 1.1 (box 3), the highest scores are obtained by taking the gapped path that conserves or increases the current score of 1 from $M_{1,1}$.

3. The second dynamic programming step adds values to the states $M_{2,j}$ and $M_{i,2}$. In this case the value of $M_{2,2}$ is maintained at 1, as the diagonal path represents a mismatch (score 0) and the horizontal and vertical paths represent gaps (score 0). The value at $M_{2,3}$ increases to 2 as the diagonal A/A match path increases the score at this point by 1 (Figure 1.1 box 4).

4. We continue in this fashion and calculate a score at each alignment state according to the highest scoring path ending at that point (Figure 1.1 box 5).

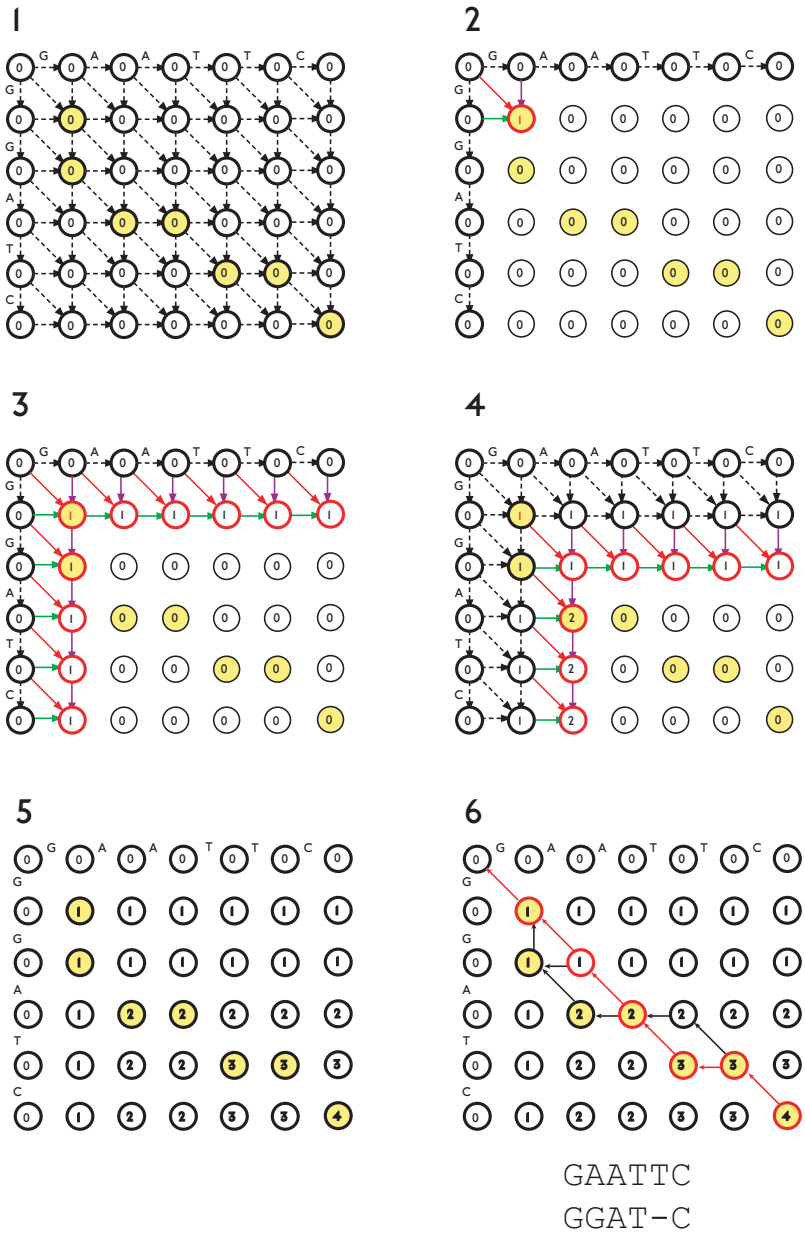


Figure 1.1: Dynamic programming path graphs for the Needleman-Wunsch algorithm. Each sequence is shown at the left and the top of each path graph. Six separate steps are shown. This figure is fully explained in the accompanying text. Yellow circles represent match states and black arrows show possible paths through the graph. The red arrows illustrate one possible alignment path through the graph. The resulting optimal alignment is also illustrated.

5. Because the Needleman-Wunsch method is a global alignment algorithm, the maximal alignment score is shown in the bottom right of the path graph (Figure 1.1 box 6). In this case the alignment score is 4. In order to generate the alignment that gives this score, we trace back from the bottom right (end of the alignment) taking at each step the path that gave the score at each point. In cases where multiple paths can give the same score at any point, we chose any of these paths. Because of this property, multiple optimal alignments may be obtained. For simplicity, most implementations will take only one alignment. In the case shown in Figure 1.1 (box 6) we have traced back from the end of the alignment according to the red path, which gives the final alignment:

```
GAATTC
|  ||  |
GGAT-C
```

This simple example does not apply penalties of any sort for gaps, in practice the Needleman-Wunsch algorithm uses a fixed penalty for gaps. A more detailed example of the Needleman-Wunsch algorithm with fixed gap penalties is shown in Figure 1.4.

1.1.2 The Smith-Waterman Algorithm

The Needleman-Wunsch algorithm produces a global alignment between two sequences. In other words, the alignment must span every residue of each of the two sequences being aligned. Frequently biological sequences will exhibit similarity across regions of their length but not their full length. One example of this is when two sequences with multiple domains share a common domain. For this reason algorithms have been written which can discover optimal local alignments between two sequences. One of the first local alignment algorithms developed was the Smith-Waterman algorithm (Smith and Waterman, 1981). For local alignment detection we need to consider alignments that can begin and end at any of the MN positions in the alignment matrix. Where M and N represent the lengths of each of the sequences being aligned. Since every possible position in the matrix can be the start of an alignment, scores in the matrix are never allowed to fall below zero. Because

begin $M_{0,0} \leftarrow 0$
  for $j \leftarrow 1$ to N **do**
    $M_{0,j} \leftarrow M_{i,j-1} + \sigma(\binom{-}{b_j})$
  end
  for $i \leftarrow 1$ to M **do**
    $M_{i,0} \leftarrow M_{i,j-1} + \sigma(\binom{a_i}{-})$
    for $j \leftarrow 1$ to N **do**
      $M_{i,j} \leftarrow max[M_{i-1,j-1} + \sigma(\binom{a_i}{b_j}), M_{i,j-1} + \sigma(\binom{a_i}{-}), M_{i-1,j} + \sigma(\binom{-}{b_j})]$
    end
  end
  write "Global Similarity Score is" $M_{M,N}$
end

Figure 1.2: Pseudocode for the dynamic programming implementation of the Needleman-Wunsch algorithm.

every point in the matrix can also be viewed as the end of an alignment, we also need to store the highest score from each step. The added complexity of the local alignment method is in fact very simple to implement. The Smith-Waterman pseudocode is shown in Figure 1.3. The only differences are the storage of the highest score (*best*) and the added criteria that no matrix entry can have a negative score. The highest scoring alignment is built by starting at the position in the matrix with the highest score, and tracing back the alignment path from which this score arose until a zero value is reached. A comparison of the global Needleman-Wunsch method and local Smith-Waterman method is shown in Figure 1.4. In practice, local and global alignment algorithms of this type are relatively similar in terms of computational complexity (Table 1.1).

1.1.3 The FASTA Algorithm

Although the Needleman-Wunsch and Smith-Waterman algorithms are rigorous methods for detecting optimal local and global alignments between sequences, they are quite computationally intensive. In order to compare a query sequence against a large database of sequences, a significant amount of computation time may be required, even with high-performance computers. For this reason, many faster heuristic alignment methods have been developed. These methods unlike the Needleman-Wunsch and Smith-Waterman

begin $best \leftarrow 0$
  for $j \leftarrow 1$ to N **do**
    $M_{0,j} \leftarrow 0$
  end
  for $i \leftarrow 1$ to M **do**
    $M_{i,0} \leftarrow 0$
    for $j \leftarrow 1$ to N **do**
      $M_{i,j} \leftarrow max[0, M_{i-1,j-1} + \sigma(\binom{a_i}{b_j}), M_{i,j-1} + \sigma(\binom{a_i}{-}), M_{i-1,j} + \sigma(\binom{-}{b_j})]$
      $best \leftarrow max(M_{i,j}, best)$
    end
  end
  write "Local Similarity Score is" *best*
end

Figure 1.3: Pseudocode for the dynamic programming implementation of the Smith-Waterman algorithm.

methods, do not examine the complete space of all possible alignments, but instead use heuristics to quickly identify potentially high scoring alignments and hence limit the search space considerably. The first such method developed was the FASTA algorithm (Pearson, 1990). FASTA concentrates on identifying regions containing high concentrations of pairs (ktup=2) or single aligned residues (ktup=1). These regions are then explored for high scoring alignments. The action of the FASTA algorithm can be summarised as follows:

- Step 1) Identify regions shared by two sequences with the highest density of identities (ktup=1) or pairs of identities (ktup=2).
- Step 2) Rescan the ten regions with the highest density of identities using a substitution matrix. Trim the ends of the region to include only those residues contributing to the highest score. Each region is a partial alignment without gaps.
- Step 3) If there are several initial regions with scores greater than a given cut-off value, check whether the trimmed initial regions can be joined to form an approximate alignment with gaps. Calculate a similarity score that is the sum of the joined initial regions minus a

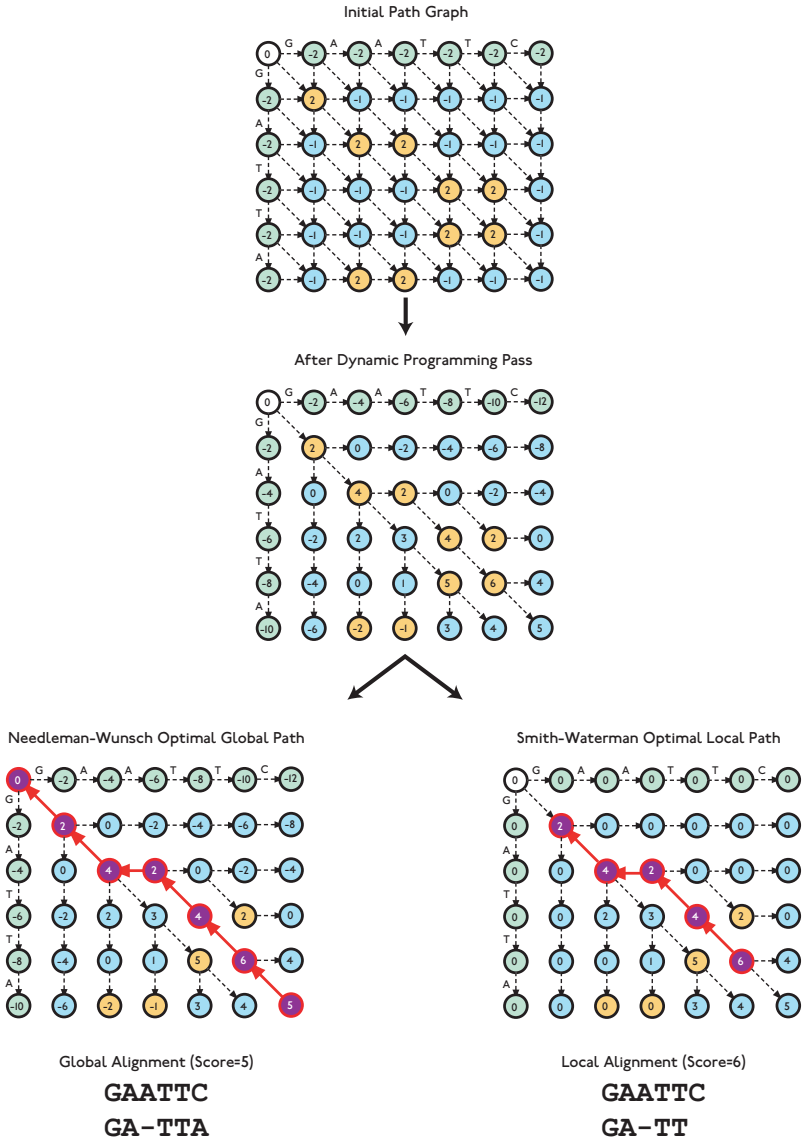


Figure 1.4: Dynamic programming path graphs for Needleman-Wunsch and Smith-Waterman alignments between two sequences using fixed gap penalties. Each sequence is shown at the left and the top of each path graph. The match states are shown in yellow. Optimal paths and alignments for global and local implementations are shown with red arrows. The final path graph shown at the bottom left of the figure represents the Needleman-Wunsch global alignment for these two sequences. The final path graph on the bottom right illustrates the equivalent Smith-Waterman local alignment.

penalty (usually 20) for each gap. The score of the single best initial region found in Step 2 is also reported.

- Step 4) For sequences with scores greater than a threshold, construct an optimal local alignment of the query sequence and the library sequence, considering only those residues that lie in a band centred on the best initial region found in Step 2. For protein searches with ktup=2 a 16 residue band is used by default. A 32 residue band is used with ktup=1.

- Step 5) After the first 60,000 scores have been calculated, normalise the raw similarity scores using estimates for the statistical parameters of the extreme value distribution. The default strategy regresses the similarity score against $ln(library - sequencelength)$ and calculates the average variance in similarity scores. Z-scores (normalised scores with mean 0 and variance 1) are calculated, and the calculation is repeated with library sequences with Z-scores greater than 5.0 and less than -5.0 removed. These Z-scores are then used to rank the library sequences.

- Step 6) Finally the Smith-Waterman algorithm (without limitation on gap size) is used to display alignments.

1.1.4 The BLAST Algorithm

Another heuristic alignment algorithm is the BLAST algorithm (Basic Local Alignment Search Tool) (Altschul et al., 1990; Altschul et al., 1997). This algorithm is significantly faster than FASTA and is the most commonly used algorithm for sequence homology detection. The initial phase of the algorithm is to build a list of *words* within a query sequence, which represent sequential residue combinations of a specific size (usually 3 for protein sequences). The central idea of the BLAST algorithm is that a statistically significant alignment of two sequences is likely to contain a high-scoring pair of aligned words. A discrete finite automaton is constructed to rapidly detect matches of these words in a database of query sequences. Word matches are then extended into ungapped local alignments, and alignments that score above a specific threshold are extended using slow gapped alignment extensions. Older versions of BLAST could not produce gapped alignments (Altschul et al., 1990), but instead multiple ungapped alignments. Newer versions of BLAST however are much improved and can rapidly and accurately produce gapped local alignments (Altschul et al., 1997). In order to

understand the operation of the BLAST algorithm, it is worth focusing on the initial word generation step. Here is an example of word generation for the sequence `qlnfsagw` showing all words of size 2:

```
q l n f s a g w

q l
  l n
    n f
      f s
        s a
          a g
            g w
```

This word list can be extended to include words with residues of similar chemical properties, as defined in a substitution matrix. The words are scored by aligning them back to the query sequence using the scoring system from the substitution matrix. Words that score higher than a specific threshold are kept. The example below extends the initial list of words and uses a threshold of 8 for word scoring and the PAM120 substitution matrix.

```
Expanded Word List
ql: ql, qm, hl, zl
ln: ln, lb
nf: nf, af, ny, df, qf, ef, gf, hf, kf, sf, tf, bf, zf
fs: fs, fa, fn, fd, fg, fp, ft, fb, ys
sa: no words score 8 or more, including the initial word sa
ag: ag
gw: gw, aw, rw, nw, dw, qw, ew, hw, iw, kw,
mw, pw, sw, tw, vw,bw, zw, xw
```

The action of BLAST is illustrated in Figure 1.5 and can be summarised as follows:

- Step 1) For each three amino acids in the query sequence, identify all of the substitutions of each word that have a similarity score greater than a threshold score T.

- Step 2) Build a discrete finite automaton (DFA) to recognise the list of identical and substituted three letter words.
- Step 3) Use the DFA to identify all matching words in sequences in the database (Figure 1.5 box 1). The refined BLAST algorithm is based upon the observation that a high scoring pair (HSP) of interest is much longer than a single word pair, and may therefore entail multiple hits on the same diagonal and within a relatively short distance of one another. Hence, if two non-overlapping matches are found closer than a threshold distance A (Figure 1.5 box 2), each match is extended both forwards and backwards without allowing gaps. Extended matches above a specific threshold are stored. These matched ungapped alignment segments above the threshold are called high scoring pairs (HSPs). If a high scoring pair exhibits a score above yet another threshold, then gapped extensions are triggered. These extensions start from a seed pair of aligned residues within the HSP (Figure 1.5 box 3). The seed is chosen by locating an 11 residue alignment with the highest score along the HSP and taking its central residue as the seed. From the seed residue, gapped extensions travel across the path graph originating from the seed residue, and the highest scoring gapped local alignment is stored if it is deemed to be statistically significant (Figure 1.5 box 4).
- Step 4) Report all of the significant alignments detected.

1.1.5 Statistics for Sequence Alignment Searching

Finding optimal or sub-optimal alignments is only one part of the sequence alignment problem. Another problem that needs to be addressed is the scoring of alignments to separate biologically meaningful alignments from ones that have arisen purely by chance. Accurate statistical analysis of local sequence similarity scores can improve the detection of distantly related proteins (Karlin and Altschul, 1990; Mott, 1992). Most of the major similarity searching algorithms (such as BLAST, FASTA and Smith-Waterman) have statistical estimates built into their scoring systems. Perhaps the most widely used statistical estimate for large-scale sequence comparison is the expectation value (*E-value*) used by the BLAST algorithm. In this section we will

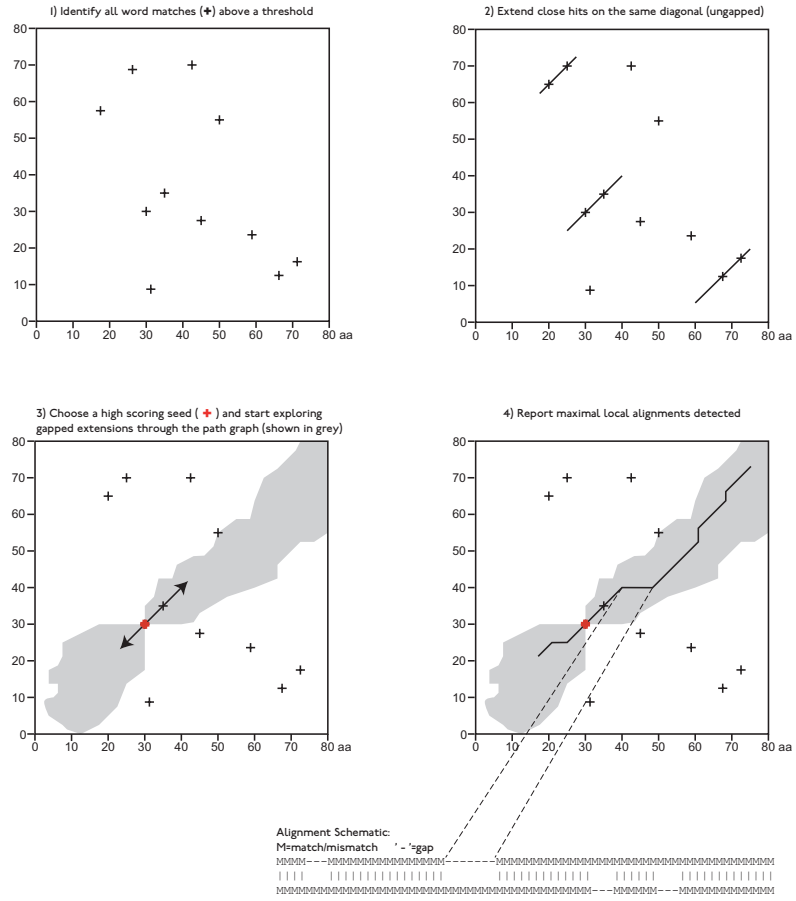


Figure 1.5: Overview of the BLAST algorithm. Each box represents two sequences (one on each axis) according to length. Box 1 shows the relative positions on each sequence of detected word matches between both sequences. In box 2, BLAST has identified high-scoring matches which occur on the same diagonal and relatively close together. Box 3, shows the generation of an ungapped seed alignment from one of these diagonal match pairs. Finally in the box 4, a high scoring gapped alignment is extended from the seed, to produce a sequence alignment. BLAST is significantly faster than methods such as Smith-Waterman because the computationally slow gapped extension steps are restricted to those regions shown in grey, and not the whole graph. An alignment schematic from the result in box 4 is shown (M =match; '-' =gap).

discuss the statistical analysis of sequence alignments, for alignments without gaps.

Ungapped Alignments

The sum of a large number of independent identically distributed (IID) random variables will tend to a normal distribution. This is not the case however, for sequence alignment scores. In this case we are dealing with the maximum values of a large number of IID random variables, which will tend to an extreme value distribution (Gumbel, 1958). An illustration of the Extreme Value Distribution (EVD) is shown in Figure 1.6.

Karlin and Altschul undertook one of the first detailed statistical analyses of local similarity scores for sequence alignments, without gaps and using an arbitrary amino acid substitution matrix (Karlin and Altschul, 1990). This statistical analysis assumes that the distribution of sequence similarity scores obtained by searching a query sequence against a database of sequences should follow the extreme value distribution (shown in Figure 1.6). Karlin and Altschul derived an appropriate distribution analytically, using results which have been previously described (Dembo and Karlin, 1991), with two parameters called λ and K. These parameters can be thought of simply as natural scales for the scoring system and the search space size respectively. Values for λ and K are derived from the substitution matrix and the amino acid composition of the query sequence. One can use these values to normalise a similarity score S obtained from an alignment with a arbitrary scoring matrix as follows:

$$S' = \lambda S - \ln Kmn \tag{1.1}$$

Where S' is the normalised similarity score for the alignment, m is the length of the query sequence and n is the total length of the library sequence. The probability of obtaining a normalised similarity S' score above a threshold x can then be calculated as:

$$P(S' \geq x) = 1 - exp(-e^{-x}) \tag{1.2}$$

Equations 1.1 and 1.2 can be merged to give:

$$P(S \geq x) = 1 - exp(-Kmne^{-\lambda x}) \tag{1.3}$$

An expectation value (E-value) can be calculated from this probability P. The expectation value is the probability of finding a score $S' \geq X$ for a search of D sequences, and is calculated as follows:

$$E(S' \geq X) = PD \tag{1.4}$$

The λ constant is a scale parameter for converting a raw similarity score S into a natural scale and is calculated as the positive root of:

$$\sum_{a,b} q_a q_b e^{\lambda S} = 1 \tag{1.5}$$

The K constant corrects for non-independence of possible match starting points. Deriving K is convoluted and is described elsewhere (Karlin and Altschul, 1990). Both λ and K parameters are calculated analytically for ungapped alignment statistics but must be estimated for gapped alignments. Current versions of BLAST and FASTA report scores in terms of information *bits*. This is a normalised and corrected similarity score and is derived as follows:

$$S_{bit} = \frac{\lambda S_{raw} - lnK}{ln2} \tag{1.6}$$

Expectation values for scores in bits are then calculated as follows:

$$E(S_{bit}) = mn2^{-S_{bit}} = \frac{mn}{2^{S_{bit}}} \tag{1.7}$$

Example output from BLAST is shown in Figure 1.7. Values for λ and K have been calculated analytically by the algorithm, and are shown at the bottom of the output. In this case these values have been calculated as $\lambda = 0.323$ and $K = 0.140$. The raw score for this alignment is obtained by looking up each of the alignment residue pairings in the BLOSUM62 matrix (Figure 1.8) and taking the sum of all scores for the complete alignment. In this example the positive and negative scores for all pairs are shown below the alignment. The total score in this case is 145. This score can be normalised using Equation 1.6, resulting in a normalised score of 70.4 bits.

$$S_{bit} = \frac{(0.323 \times 145) - ln(0.140)}{ln(2)} = \frac{48.80}{0.693} = 70.42 \tag{1.8}$$

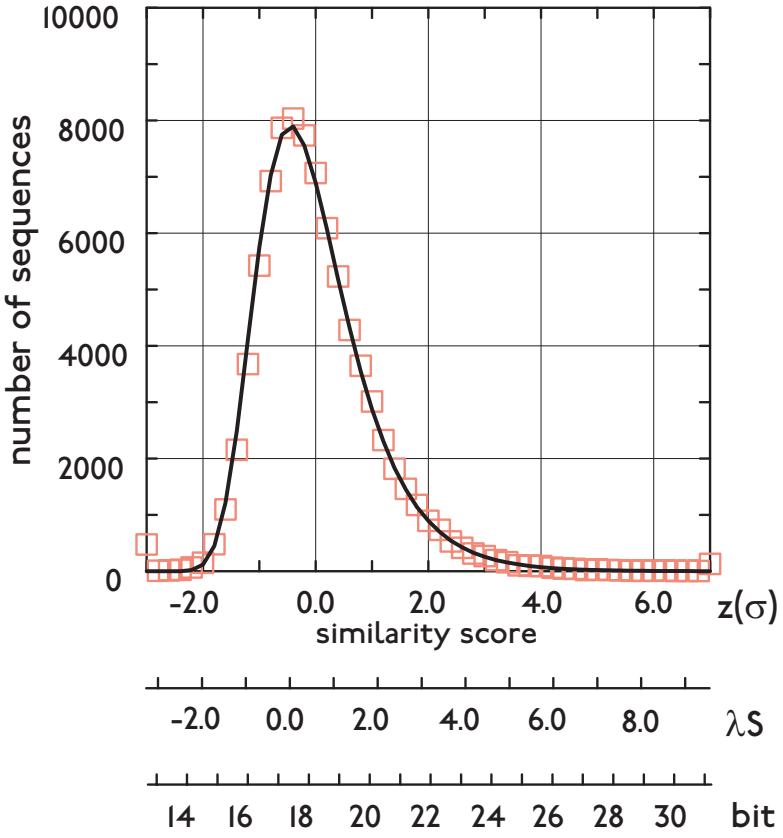


Figure 1.6: The Extreme Value Distribution. The x-axis shows raw and normalised scores for a set of alignments, the y-axis shows the distribution of these alignment scores across all alignments.

```
BLASTP 2.0.8 [Jan-05-1999]


Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs",  Nucleic Acids Res. 25:3389-3402.

Query= MG293 glycerophosphoryl diester phosphodiesterase, putative
{Mycoplasma pneumoniae} (244 letters)

Database: /ebi/cgg/data/genomes/cgd 73,345 sequences; 94,110,160 total letters

 Score = 70.4 bits (145), Expect(3) = 9e-23
 Identities = 31/53 (58%), Positives = 37/53 (69%)

>CPER-X13-001877 glycerophosphodiester phosphodiesterase (Length=238)

Query: 7  LLAHRGYSFIAPENTKLAFDLAFEYCFDGIELDVHLTKDEQLVIIHDETTLRT 59
          + AHRG+S+  PENT LAF  A +    GIELDVH +KD +LVIIHDE   RT
Sbjct: 3  VFAHRGFSYKYPENTLLAFKEALKLDIYGIELDVHKSKDGKLVIIHDEDIKRT 55

          +++++++++-++++-+++-++++--+-+++++++-+++-++++++++---++
Scores:   10495634332756524461340113036454648215621464486511255 = 145


  Database: /ebi/cgg/data/genomes/cgd
  Posted date:  Mar 2, 2002  1:02 AM
  Number of letters in database: 94,110,160
  Number of sequences in database:  273,345

Lambda     K      H
   0.323    0.140    0.406

Matrix: BLOSUM62
Number of Hits to DB: 1st pass: 41778674, 2nd pass: 1051929
Number of Sequences: 1st pass: 273345, 2nd pass: 3593
Number of extensions: 1st pass: 1687673, 2nd pass: 836884
Number of successful extensions: 1st pass: 3593, 2nd pass: 9007
Number of sequences better than 10.0: 126

length of query: 244                 length of database: 94110160
effective length of query: 192       effective length of database: 79896220
effective search space: 15340074240  effective search space used: 15340074240
```

Figure 1.7: Example output from the BLASTp algorithm. Individual scores from aligned residue pairs have been illustrated, and are explained further in the text.

The E-value shown is then calculated using Equation 1.7, where m is the size of the query sequence (in amino acids) and n is the total size of the sequence database searched (also in amino acids). These values are also shown in Figure 1.7. The E-value calculated represents the expectation likelihood that an alignment of these two sequences detected within a database of given size, has occurred by chance.

Scoring Matrices

The statistics described in the previous section rely on scores derived from a substitution matrix. This matrix scores any given pair of aligned residues according to their substitution likelihood in a protein family at a certain evolutionary distance.

Given how important this substitution matrix is for accurate alignment scoring, it is imperative that accurate matrices are constructed. The most simple approach is to count the frequencies of aligned residue pairs in confirmed alignments and to develop maximum likelihood estimates from these data. This approach can however produce misleading results. The reason for this, is that sequences usually occur in families, so alignments are not always independent from one another. Some sequence families have diverged more recently from ancestor sequences than other sequences. For closely related (recently diverged) sequences, one would expect alignments of these sequences to contain many identical residues. The probability P_{ab} for aligned residues that are not identical ($a \neq b$) should be very low, so that scores obtained from the matrix for alignment of a and b will be negative. Conversely for sequences which have diverged a longer time ago, P_{ab} will approach the background frequency of amino acids a and b ($q_a q_b$), and the associated score for an a, b alignment will be close to zero. For this reason substitution matrices are optimised for different levels of divergence between sequences, and care should be taken to choose the appropriate matrix for sequence comparisons.

Two types of substitution matrices are in common use. The *Point Accepted Mutation* (PAM) matrix (Dayhoff, 1978) uses closely related proteins to obtain substitution rates for recently diverged proteins and extrapolates rates for sequences which are more distant. The matrix is constructed by building phylogenetic trees for closely related protein families. Sequences are compared to their closest ancestor in a tree and frequencies of all residue

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

Figure 1.8: The BLOSUM62 substitution matrix (Henikoff and Henikoff, 1991). Scores for every pair of aligned residues are shown. Scores indicate the likelihood of substitutions between two residue types in a protein family at a specific evolutionary distance. Scores are scaled in half-bit units, and may be positive or negative.

pairings are stored. These values are transformed into conditional substitution probabilities for short time intervals. Finally the results are extrapolated to longer time intervals according to the number of *point accepted mutations* (PAM units) that are expected. A PAM value of one indicates that we expect a 1% level of substitution. Matrices containing these substitution probabilities are then created for increasing PAM values. One commonly used matrix is the PAM250 matrix which is scaled by $3/log2$ giving scores in third-bits for aligned residues. This extrapolation from short time intervals to longer intervals fails to capture the real complexity of biological sequence evolution (Henikoff and Henikoff, 1991). Closely related sequences exhibit different patterns of amino acid substitution. Single related amino acid changes are common in closely related sequences, for example isoleucine to leucine (`I` $\Longleftrightarrow$ `L`) or leucine to valine (`L` $\Longleftrightarrow$ `V`). Also, sequences at large evolutionary distances can exhibit all manner of codon changes in their DNA sequences.

The BLOSUM set of substitution matrices are an attempt to overcome this bias in PAM matrices (Henikoff and Henikoff, 1992). The data used to construct a BLOSUM matrix is derived from the BLOCKS database of aligned ungapped regions (or blocks) from protein families (Henikoff and Henikoff, 1991). This set of aligned blocks is clustered by placing blocks in the same cluster if their percentage of identical residues is above a threshold. The frequencies of observing an alignment between residues which come from different clusters is calculated, and normalised to correct for the relative sizes of clusters. This set of frequencies is then used to estimate substitution scores using a log-odds calculation:

$$p_{ab} = \log(\frac{p_{ab}}{q_a q_b})$$

This log-odds matrix elements are scaled and rounded to nearest integer values. The commonly used BLOSUM62 matrix (Figure 1.8) is scaled so that its values are half-bits. This scaling is performed by multiplying log-odds matrix elements by 2/log2.

1.1.6 Profile Searching: PSI-BLAST and HMMER

Although sequence similarity searching algorithms and their associated statistics can easily identify similarity relationships between proteins, there are limits to their sensitivity. Many functionally important protein similarities

only become apparent with comparison of the three-dimensional structures of proteins (Holm and Sander, 1997; Brenner et al., 1998).

Unfortunately the majority of proteins do not yet have three-dimensional structures determined. In the absence of these structural data, methods have been developed that can identify remotely similar proteins based on *profile* searching. Given a set of related proteins, such as those with significant BLAST similarity, it is possible to align these proteins and use observed patterns of amino acid conservation within the family to build a statistical profile for this protein family. Searching with a sequence profile is highly desirable, as it captures information contained in the whole family which may be missed by simply comparing a single sequence from that family to a database (Gribskov et al., 1987). Patterns of conservation identified from the alignment of related sequences can hence aid the recognition of distant similarities (Altschul et al., 1997; Brenner et al., 1998). Many methods exist to build and search profiles from sequence alignments. Traditional pairwise alignment algorithms such as BLAST, FASTA and Smith-Waterman use position independent scoring parameters. Profiles use position-specific scores for amino acids pairings and position-specific scores for gaps. This property of profile methods is well suited to capturing the degree of conservation at specific positions across a multiple alignment, and the rate at which gaps may be introduced. Two of the most commonly used methods for profile sequence searching are PSI-BLAST and HMMER (Altschul et al., 1997; Eddy, 1998). In this section we will briefly discuss these two methods, and their application to biological sequence analysis.

PSI-BLAST

The PSI-BLAST algorithm is based on the standard BLAST algorithm. A query sequence is scanned against a database of sequences and high scoring alignments are detected. A multiple alignment of detected sequences is then used to construct a profile. For each position in the derived alignment pattern, every amino acid is assigned a score. If a residue is highly conserved at a particular position, that residue is assigned a high positive score, residues that are not conserved are assigned high negative scores. At weakly conserved positions, all residues receive scores at or near zero. Position-specific scores can also be assigned to potential insertions and deletions.

This profile is then used as a query to perform another (more sensitive)

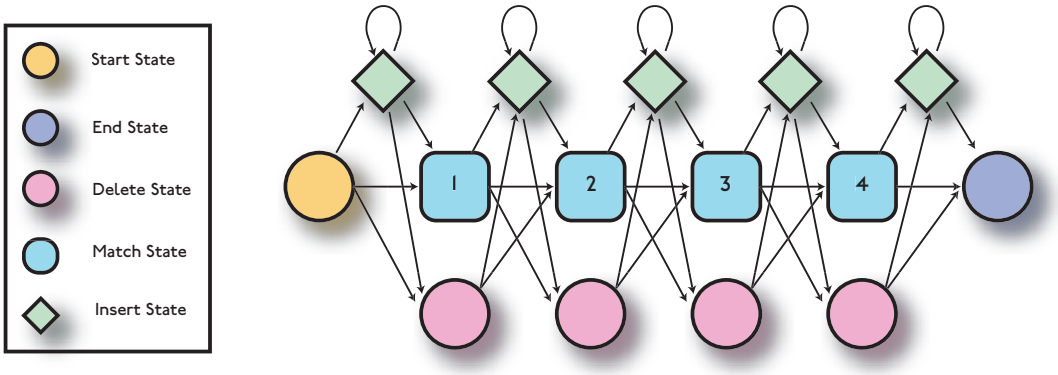


Figure 1.9: Diagram of a Profile Hidden Markov Model (HMM) for an alignment of length four (Krogh et al., 1994). Individual states and transitions between states are explained in the legend.

search of the database. The PSI-BLAST algorithm is iterative which means that subsequent searches which detect related homologues are used to further refine the profile. The iteration is programmed by the user and continues until convergence (i.e. no more significant sequence similarities are detected) or for a specific number of iterations. While this method has been shown to be fast and reliable for distant sequence homology detection (Brenner et al., 1998), care must be taken to control thresholds for the inclusion of sequences in profiles. If a threshold chosen is too permissive, it may allow unrelated protein sequences to be added to a profile and hence the method may never converge.

HMMER

The HMMER algorithm (Eddy, 1998) is similar to PSI-BLAST at a very basic level. The method generates sequence profiles from an alignment of related sequences and uses this profile to accurately and sensitively detect more distantly related sequences that fit this profile. It differs from PSI-BLAST in the method by which profiles are generated. The algorithm uses profile hidden Markov models (profile HMMs) to build a statistical model of an alignment based on the consensus sequence of a family (Krogh et al., 1994). The use of HMMs had previously been successful in the field of speech recognition and is very well suited to the problem of sequence profile generation (Eddy, 1998).

The use of HMMs allows a more systematic approach to estimating model

parameters. A HMM is a type of dynamic statistical profile, constructed by analysing the distribution of amino acids and other features in a training set of proteins. The HMM concept is more complex than simple profiles and can be visualised as a *finite state machine*, a concept frequently used in fields such as computer science.

Finite state machines travel through a series of states and produce output while moving from state to state, or when a particular state is reached. A HMM generates a protein sequence by emitting amino acids as it travels through a series of states. Each state has an associated table of amino acid emission probabilities. Transitions are controlled by transition probabilities associated with moving from state to state. An example profile HMM is shown in Figure 1.9 (Krogh et al., 1994). The term *Hidden Markov Model* arises from the fact that the state sequence is a first-order Markov chain, but only the symbol sequence is directly observed. In other words, the state sequence (e.g. an alignment) is not uniquely determined by the observed symbol sequence, but is inferred probabilistically from it.

The profile HMM shown in Figure 1.9 can be described as follows. For consensus columns in a multiple sequence alignment a *match* state models the distribution of residues that are allowed in that column. The *insert* and *delete* states allow the insertion of one or more residues between one column and the next, or the deletion of the consensus residue. Models such as this must be trained on a multiple sequence alignment to generate probabilities for states and state transitions. Once the model has been trained, it is possible to accurately detect whether other sequences fit this profile. Any sequence can be represented by a path through the model. The probability of any sequence, given the model, is computed by multiplying the emission and transition probabilities along the path. Dynamic programming approaches (such as the *Viterbi* algorithm) are used to calculate the most likely path through the model. Once the most likely path is detected, the probability of this path is used to assess the probability that a given sequence fits the model. The HMMER package provides all of the tools necessary to build a profile HMM from a multiple sequence alignment, and to search sequences against this profile (Eddy, 1998).

1.2 Computational Biology Data Sources

Bioinformatics is a data-driven field, vast quantities of information pertaining to nucleotide, amino acid and other biological data, are constantly being generated by laboratories around the world. The methods described in the first part of this introduction require accurate and reliable sequence information in order to work successfully, and hence it is imperative that biological information is stored in a meaningful and consistent format. A phrase coined by computer scientists 'garbage in, garbage out', is a succinct way of describing the problem. In this section we focus on primary nucleotide and sequence databases, as a complete discussion of biological databases is beyond the scope of this introduction. A section on protein domain family database is however included, as these resources are important for the work described in Chapters 2 & 3. Table 1.2 shows a list of some of the most useful databases and illustrates the variety of biological data that are currently available.

1.2.1 Nucleotide Sequence Databases

EMBL, GenBank and DDBJ

The three primary nucleotide sequence databases are EMBL, GenBank, and DDBJ (Stoesser et al., 2002; Benson et al., 2002; Tateno et al., 2002) (Table 1.2). These databases include sequences submitted directly by individual laboratories and genome sequencing consortia. Sequences are also taken from literature and patent applications. The databases are synchronised on a daily basis, and accession numbers are managed in a consistent manner between these three centres. Given the size of these databases (over 15 billion nucleotides) they are also available in subdivisions. For example, GenBank is currently partitioned into 17 divisions. The European Molecular Biology Laboratory (EMBL[1]) nucleotide sequence database is maintained by the European Bioinformatics Institute (EBI), in Cambridge, UK. The EMBL database can be accesses and searched using the SRS[2] system (Etzold and Argos, 1993), or the entire database can be obtained as a single flat file. The GenBank[3] nucleotide database is maintained by the National Center for Biotechnology Information (NCBI), which is part of the National Institute of

[1] http://www.ebi.ac.uk/embl/
[2] http://srs.ebi.ac.uk/
[3] http://www.ncbi.nlm.nih.gov/Genbank/

Database	Location	Type of Data	Reference
		Nucleotide Databases	
EMBL†	European Molecular Biology Laboratory	Genomic and submitted sequence	(Stoesser et al., 2002)
GenBank†	National Center for Biotechnology Information (NCBI)	Genomic and submitted sequence	(Benson et al., 2002)
DDBJ†	DNA Data Bank of Japan	Genomic and submitted sequence	(Tateno et al., 2002)
		Amino Acid Databases	
SwissProt	European Molecular Biology Laboratory, Swiss Institute for Bioinformatics (SIB)	Curated peptide sequences	(Bairoch and Apweiler, 2000)
PIR	National Biomedical Research Foundation (NBRF), in collaboration with MIPS and JIPID	Curated peptide sequences	(McGarvey et al., 2000)
NRDB90	European Molecular Biology Laboratory	Protein sequences with near-neighbour redundancy removed	(Holm and Sander, 1998)
		Protein Family, Domain and Motif Databases	
Pfam‡	Sanger Centre and PFAM consortium	Annotated protein domain families, alignments and profile HMMs	(Bateman et al., 2000)
ProSite‡	Swiss Institute of Bioinformatics (SIB)	Protein domains and families	(Falquet et al., 2002)
PRINTS‡	University of Manchester	Conserved protein sequence motifs (fingerprints)	(Attwood et al., 1999)
BLOCKS	Fred Hutchinson Cancer Research Center	Conserved protein sequence motifs (blocks)	(Henikoff and Henikoff, 1991)
ProDom‡	Institut National de la Recherche Agronomique	Protein domain families	(Corpet et al., 2000)
SMART‡	European Molecular Biology Laboratory	Protein domain families	(Schultz et al., 2000)
InterPro	European Molecular Biology Laboratory	Merged and linked database of other resources (see ‡below)	(Apweiler et al., 2001)
		Structural Information Databases	
PDB	Research Collaboratory for Structural Bioinformatics	Protein 3-dimensional structures	(Sussman et al., 1998)
CATH	University College London	Expert hierarchical classifications of protein structures	(Orengo et al., 2002)
SCOP	Laboratory for Molecular Biology (LMB), Cambridge	Expert hierarchical classifications of protein structures	(lo Conte et al., 2000)
		Pathway and Interaction Databases	
KEGG	Institute for Chemical Research, Kyoto University	Metabolic and Regulatory Pathways	(Kanehisa and Goto, 2000)
DIP	University of California, Los Angeles (UCLA)	Curated Protein-Protein Interactions	(Xenarios et al., 2002)

† EMBL, GenBank and DDBJ are cross linked
‡ These databases form part of the InterPro database of protein domains and families

Table 1.2: Summary of major biological databases.

Health (NIH) in the United States. It is accessed and searched through the Entrez system at NCBI, or can be downloaded as flat files. The DNA Data Bank of Japan (DDBJ[4]) began as a collaboration with EMBL and GenBank. It is run by the National Institute of Genetics, Japan.

Organism Specific Databases

The primary nucleotide sequence database store information in a general manner for all organisms. For many model organisms, there exist specialised genome databases which have been expertly annotated by researchers with extensive knowledge of the biology of specific organisms. These databases exist for a variety of organisms, but three of these databases (SGD, FlyBase and Ensembl) are exceptional, both in terms of methodology and annotation quality.

The FlyBase[5] resource was one of the earliest model organism databases (Ashburner, 1993). This database endeavours to capture large amounts of biological information regarding the complete genome sequence of the fruit fly *Drosophila melanogaster*. Detailed information has been gradually added to the database which includes: genes, proteins, genetic elements, chromosomal maps, aberrations, literature references and images. The database is constantly being updated and modified. Since the publication of the draft *D. melanogaster* genome (Adams et al., 2000), FlyBase has become an even more powerful resource for functional analysis of fly proteins. The Saccharomyces Genome Database[6] (SGD) is a similar resource concerning the complete genome of bakers yeast (*Saccharomyces cerevisiae*) and related yeast strains (Dwight et al., 2002). This resource was started in 1994 and also has benefited from the publication of the complete *S. cerevisiae* genome (Mewes et al., 1997).

The Ensembl[7] resource (Hubbard et al., 2002) is a joint initiative of the European Bioinformatics Institute and the Wellcome Trust Sanger Institute. The project is a complete pipeline of genome assembly, gene prediction, large-scale annotation and analysis of the draft human genome sequence. Ensembl was started in 1999 and the initial pipeline, databases and world wide web resources were put in place before the completion of the draft genome. The

[4]http://www.ddbj.nig.ac.jp/
[5]http://flybase.bio.indiana.edu/
[6]http://genome-www.stanford.edu/Saccharomyces/
[7]http://www.ensembl.org/

current version of Ensembl (Build 26) contains over 4,389 megabases, 29,181 predicted genes and 34,019 predicted transcripts. Ensembl also contains information pertaining to predicted genes and proteins, cytological markers, single nucleotide polymorphisms (SNPs), protein families, domains and a variety of other information. Ensembl is very different to other resources at some very basic levels. The entire system and data are *Open Source* (i.e. the entire system, not just the data, is freely available). Ensembl also allows users to submit annotations remotely via the *distributed annotation system* (DAS) (Dowell et al., 2001).

1.2.2 Protein Sequence Databases

SwissProt

The two major protein sequence databases (Table 1.2) are the SwissProt and PIR manually curated databases. Scientific curators collaborate with the individuals or institutions submitting sequences in order to accurately annotate the biochemical function of the protein being submitted. Literature links regarding scientific research carried out on a given protein are also stored. SwissProt[8] (Bairoch and Apweiler, 2000) is a collaborative project between the Swiss Institute of Bioinformatics and the European Bioinformatics Institute. The goal of this database is to consistently maintain a high level of annotations for each protein. Annotations include: function, domain structure and post-translational modification information. A minimal level of redundancy and a high level of integration with other databases is another key aim. SwissProt was initiated in 1986 by Amos Bairoch at the Department of Medical Biochemistry at the University of Geneva. This database is one of the best protein sequence databases in terms of the quality of the annotation and its size. The current release of SwissProt (release 40) contains 101,602 curated protein sequences.

TrEMBL is a computationally derived supplement to SwissProt that contains translations of EMBL nucleotide sequence entries that have not yet been integrated into the main SwissProt resource. Annotations for TrEMBL entries are automatically derived and are generally not of the same high quality as SwissProt entries. Both databases can be accessed and searched through the SRS[9] system (Etzold and Argos, 1993).

[8] http://www.ebi.ac.uk/swissprot/
[9] http://srs.ebi.ac.uk/

The Protein Information Resource (PIR)

The Protein Information Resource[10] (PIR) (McGarvey et al., 2000) is a division of the United States National Biomedical Research Foundation (NBRF). It is involved in a collaboration with the Munich Information Center for Protein Sequences (MIPS) and the Japanese International Protein Sequence Database (JIPID). The PIR-PSD (Protein Sequence Database) release 71.03 (February 2002) contains 283,138 entries. PIR was the first public protein sequence database and was founded in 1984 by Margaret Dayhoff.

The PIR database strives to be comprehensive, accurate, consistently annotated and well organised. However, from a curation and annotation point of view, the SwissProt database is probably more desirable. The SwissProt and PIR databases exhibit a large degree of overlap, yet there are still many sequences which can be found exclusively in only one of them.

The Non-Redundant Database (NRDB)

There is a large level of redundancy in many biological sequence databases, including the SwissProt and PIR protein sequence databases (Holm and Sander, 1998). It is desirable when searching sequences against a large database to remove closely related sequences, which provide little extra information and increase the search space significantly. The *Non-redundant Database*[11] (NRDB) is a database containing protein sequences from SwissProt, SwissNew, TrEMBL, Tremblnew, GenBank, PIR, Wormpep and PDB. Near-neighbour redundancy in this database has been removed by detecting very closely related sequences by virtue of highly similar amino-acid compositions, and subsequent sequence alignment. The NRDB90 database merges two sequences into a representative sequence if they exhibit more than 90% identity detected in this fashion. This is very useful because large-scale computational analyses do not need to be carried out on a full sequence database, but on a smaller non-redundant database, without sacrificing information.

[10] http://pir.georgetown.edu/

[11] http://www.ebi.ac.uk/~holm/nrdb90/

1.2.3 Protein Domain and Family Databases (Pfam & InterPro)

A large proportion of this thesis describes the classification of proteins into sequence families. A number of similar projects have been performing this type of analysis for the classification of protein domains and motifs. Many such databases exist, and it would be beyond the scope of this work to discuss all of them. In this section two of these resources (Pfam and InterPro), which we have used extensively during this work, will briefly be described.

Pfam is a database of multiple alignments of protein domains constructed using profile hidden Markov models (Bateman et al., 2000). The database is comprehensive well curated and very useful for determining the presence of domains or motifs within protein sequences. Domain families are constructed by building a seed alignment of related sequences taken from sources such as SwissProt, Prosite and ProDom. Care is taken at this stage to check for errors in both the sequences and the seed alignment. A profile HMM is constructed from a finished seed alignment using the HMMER package (Eddy, 1998). This profile is used to search the SwissProt database for more members of the family, and a multiple sequence alignment for the full family is generated. Once again, these full alignments are also rigorously checked for errors and inconsistencies. These finished full alignments are stored and (in many cases) extensively annotated in the PfamA database of high quality alignments.

This method is clean and accurate for the classification of protein domains into families, but can be manually intensive. In order to enrich the database further and to cover sequences in SwissProt that are not already part of the PfamA database, another automatic protocol is used. In this case the Domainer (Sonnhammer and Kahn, 1994) algorithm is applied to SwissProt BLAST similarity data, and used to automatically detect conserved domains and motifs within proteins. Profiles are once again constructed from these automatically generated alignments, and are used to detect more family members which are built into a final alignment. These results are stored in the PfamB subdivision. Currently the Pfam database consists of 3,360 protein domain families, and is growing rapidly.

Many other family databases exist, providing classifications of full-length proteins, motifs and domains. The Pfam database has perhaps been one of the most successful family databases. Recently, the InterPro initiative

(Apweiler et al., 2001) has linked the major family databases into a single resource linked to the SwissProt and TrEMBL databases. This database contains families from the Pfam, Prosite, Prints, ProDom, SMART and TIGRFams resources. InterPro is an exceptionally useful resource as it allows the strengths of each separate family database to be combined into a single comprehensive resource for information regarding protein sequence motifs and domains. InterPro release 4.0 consists of 4,691 entries representing: 1,068 domains, 3,532 families, 74 repeats, and 15 post-translational modification sites. The database also stores over 2 million links between a large proportion of SwissProt and TrEMBL proteins. Both the Pfam and InterPro databases have been exceptionally useful for the research described in this thesis, and will be further discussed in Chapters 2 & 3.

Chapter 2

Algorithms for Automatic Protein Sequence Clustering

Clustering techniques are used for the classification of objects with discrete features into groups. This type of classification procedure has been used extensively in many fields. Biology is an observational and experimental science, and as such, classification is of paramount importance. The earliest recorded biological clustering was developed by Aristotle, a student of Plato around 300BC. The classification scheme he developed for plants and animals remained mostly unchanged for two thousand years until Carl Von Linnaeus in 1735 published his work *Systema naturae* (Linnaeus, 1735). Linnaeus was a Swedish botanist known as the father of taxonomy. The species classification system he developed (based on 'genus' and 'species') is still very much in use today. An early biological classification from 1874 is shown in Figure 2.1.

From a bioinformatics perspective, clustering has been used extensively in the field of molecular phylogeny for the reconstruction of phylogenetic trees (Nuttall, 1904; Fitch and Margoliash, 1967) from protein and nucleotide sequence data. Recently however, clustering has also been employed for the classification of many other diverse types of information within the field of bioinformatics and computational biology. One example of this, is the clustering of genes into related groups based on the co-expression of their messenger RNAs (Eisen et al., 1998), as determined by microarray experiments (Schena et al., 1995). The development of whole genome microarray (DeRisi et al., 1997) approaches has further tested the limits of various clustering techniques to make sense of these complex data.

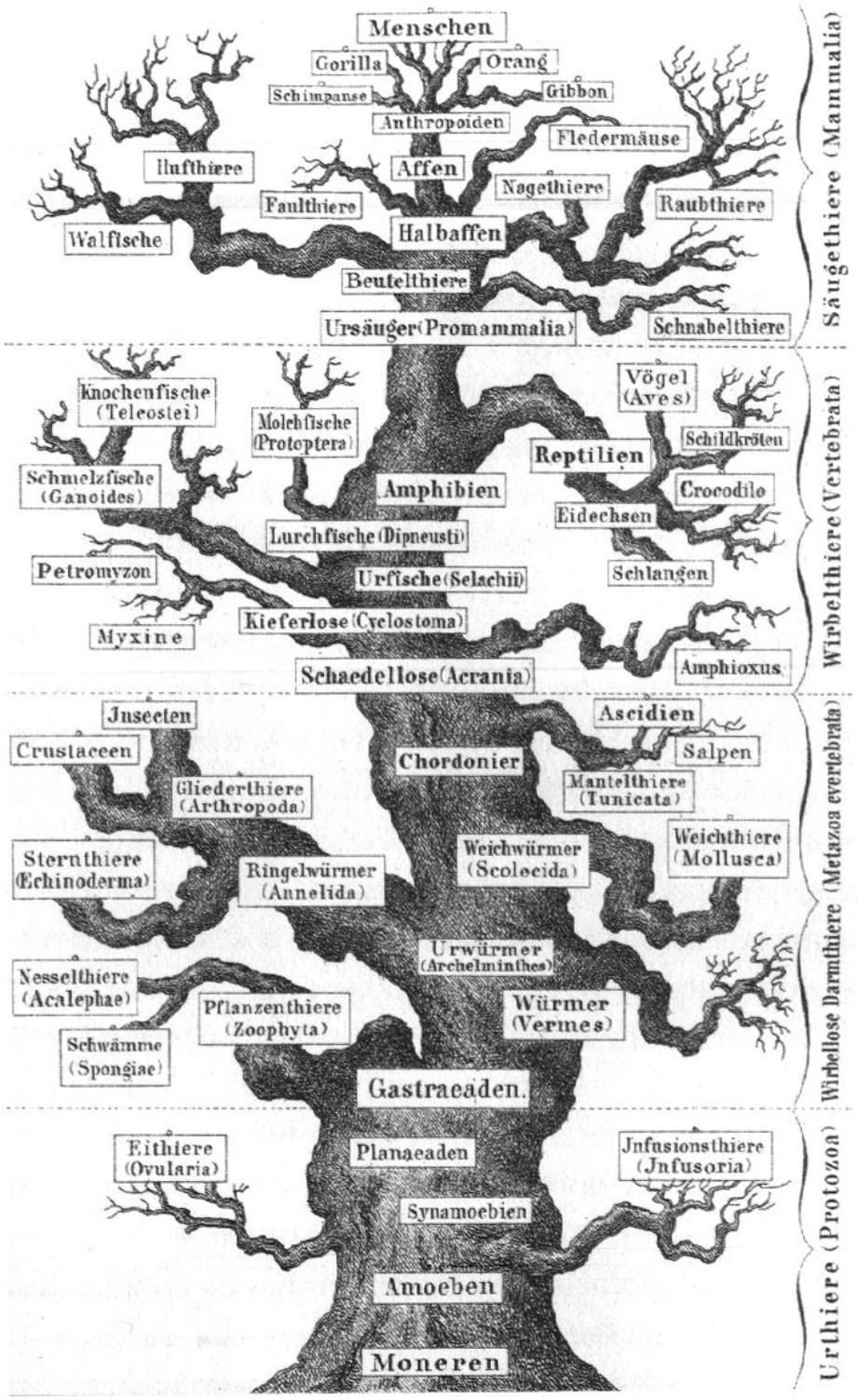


Figure 2.1: Stammbaum des Menschen: 'Family Tree of Mankind' (Haeckel, 1874).

Another important field within bioinformatics is the clustering of proteins, based on their sharing of sequence similarity, into related evolutionary groups which are termed *protein families*. This Chapter will describe the problem of protein sequence clustering, previous approaches to this problem and two novel clustering algorithms developed during this research, for the classification of proteins into functionally related groups.

2.1 Genomic Protein Family Analysis

The development of rapid and accurate global and local sequence alignment tools, such as BLAST, FASTA and Smith-Waterman (Altschul et al., 1990; Pearson, 1990; Smith and Waterman, 1981), which have been described in detail in the previous chapter, have broadened the scope of bioinformatics research enormously. Large scale functional analysis of proteins and nucleotide sequences became possible, and provided new insights into molecular and genomic processes. While these tools became readily available in the early 1990s, there was still relatively little information in protein and nucleotide sequence databases that could be used for accurate genome comparison and analysis.

An enormous growth of sequence data began with the sequencing of the first complete genome of a free living organism, that of *Haemophilus influenzae* (Fleischmann et al., 1995). In the following seven years over seventy complete genomes have been completely sequenced. The rate of growth of sequence information is enormous, as novel sequencing and assembly technologies such as capillary sequencing and whole genome shotgun approaches (Carrilho et al., 1996; Venter et al., 1996) have been developed. These developments have provided valuable genomic data that allows the development of many novel tools and methods for complete genome analysis and comparison. The goal of functional genomics is to determine the function of proteins predicted from these sequencing projects (Tsoka and Ouzounis, 2000b; Eisenberg et al., 2000). To achieve this goal, computational approaches such as sequence clustering can assist the classification of functional genomics targets.

As the flood of sequence information increases rapidly, various attempts have been made for widespread classification of predicted protein sequences into related functional and evolutionary groups. Throughout this chapter the term "*protein family*" will be used to denote groups of proteins related

in terms of both evolution and function (Fitch, 1973; Dayhoff, 1976). Such classifications of proteins into families are of enormous use in many areas of biological research.

At one level, protein families can be used to explore the function of a predicted protein sequence. If a protein of unknown function is deemed to be a member of a protein family, then this unknown protein can be assigned a function based on other proteins within that family, whose functions may be known (Hegyi and Gerstein, 1999). Surprisingly, there are many pairs of clearly orthologous proteins in databases, where one of the pair is annotated with a functional description, but the other is annotated as "unknown" (Galperin and Koonin, 2000). This predictive aspect of protein families is hence very useful, as the common method of functional assignment (taking the highest scoring similarity from a database search) can frequently be misleading or inaccurate in terms of functional assignment (Kyrpides and Ouzounis, 1998).

Complete genome comparisons can also be aided greatly by accurate protein family information (Ouzounis et al., 1996). Families may be specific to certain taxonomic groups or widespread across all domains of life (Ouzounis and Kyrpides, 1996), facts that can provide evolutionary insights into the underlying biology of organisms (Rubin et al., 2000). The set of protein families of a complete genome provides a functional and evolutionary snapshot of its underlying biological mechanisms. Indeed, genome phylogeny can be accurately reconstructed by analysing the proportion of protein families that two genomes share and is remarkably similar to phylogenies constructed from ribosomal RNA sequences (Huynen and Bork, 1998; Enright and Ouzounis, 2001b). Investigation of universal protein families which occur in all complete genome sequences, can also shed light on the functional content of the proposed last universal common ancestor (Kyrpides et al., 1999; Doolittle, 2000).

Many methods exist for the classification of proteins into families, and many attempts have been made to build large comprehensive, searchable databases of protein families. However, no method exists for accurate and exhaustive classification of sequence space, which at present totals over 600,000 protein sequences from many different genomes. At the level of protein domains, which also provide functional and evolutionary insight, progress has been rapid. The Pfam protein family database (described in Chapter 1) is an excellent example of this, it combines a rapid and accurate search strategy

for detecting and classifying protein domains into *domain families*, and a large searchable database of these results (Bateman et al., 1999). Most importantly however, this database is updated regularly and curated, making it of enormous benefit to the scientific community.

This chapter will detail the development of two novel algorithms for the accurate detection of protein *sequence* families, which we hope will also be of benefit to the wider scientific community.

2.2 Protein Sequence Clustering Techniques

To define a protein family, algorithms should take into account all similarity relationships in a given arbitrary set of sequences, a process that is defined as *sequence clustering* (Heger and Holm, 2000). Clustering techniques group objects together based on this sharing of discrete features. In order to cluster a set of objects, a measure is used to assess how similar each object is to the other objects in the set, based on their sharing of features. For protein family detection the similarity measure used is generally sequence similarity determined using a sequence alignment search tool such as BLAST, PSI-BLAST or HMMER. In principle the concept of clustering protein sequences into families from these data is straightforward, and can be described as follows:

1. Take a set A of protein sequences to be clustered.
2. Compare each protein from A to every other protein in the set A.
3. Use the detected similarities to build a distance measure between all proteins in A.
4. Apply a standard clustering technique (Such as K-means, Single Linkage Clustering, etc).
5. Interpret this clustering result as protein families.

In practice such simplistic approaches generally fail to produce biologically relevant families for many reasons. Firstly, many proteins have complex domain structures and may only show sequence similarity across certain regions of their length (Figure 2.2). Secondly, the underlying assumption that homologous proteins are *always* evolutionarily related *and* have conserved function

is not always true. Complex evolutionary events, such as horizontal gene transfer, non-orthologous gene displacement, gene duplication and gene recruitment can make these assumptions invalid (Galperin and Koonin, 2000). Another serious problem is that of data size and complexity, a problem which is faced in almost all aspects of bioinformatics. Clustering methods must be able to deal with large numbers of protein sequences and the computational cost of such methods should not be so high as to make them impractical. In the following sections each of these serious issues will be discussed individually, together with their impact on conventional protein sequence clustering techniques.

2.2.1 Multi-Domain Proteins

Multi-Domain proteins are very common in complete genomes, especially in Eukaryotic genomes. Indeed, given the limited repertoire of domains (Chothia, 1992; Wolf et al., 2000) and folds (Holm and Sander, 1996), domain recombination and domain shuffling events (Doolittle, 1985; Patthy, 1985) appear to be a significant factor in genome evolution (Rossman et al., 1974; Apic et al., 2001a). Multi-domain proteins make protein family analysis very difficult indeed (Smith and Zhang, 1997). The presence of a shared domain within a group of proteins does not necessarily imply that these proteins perform the same biochemical function (Henikoff et al., 1997). Ideally, proteins should be classified into a single family only if they exhibit highly similar domain architectures. A simple example of this problem is shown in Figures 2.2 & 2.3. The first figure shows a multi-domain protein A with two domains. The first domain of this protein (shown in blue) is similar to a related family of proteins which possess only this domain. Similarly the second domain (orange) shows that this region of protein A is similar to a separate family of related proteins. These similarity relationships can be depicted more clearly as a graph (Figure 2.3). The two unrelated sets of orange and blue proteins can be connected via an intermediate step through protein A. Many clustering techniques build clusters using such intermediate sequence relationships to improve the detection of remotely homologous cluster members (Park et al., 1997). Such methods, when presented with a case such as this, will fail to keep unrelated blue and orange proteins separate. An ideal clustering of this example case is for blue and orange proteins to form separate clusters, while the multi-domain protein A is a member of

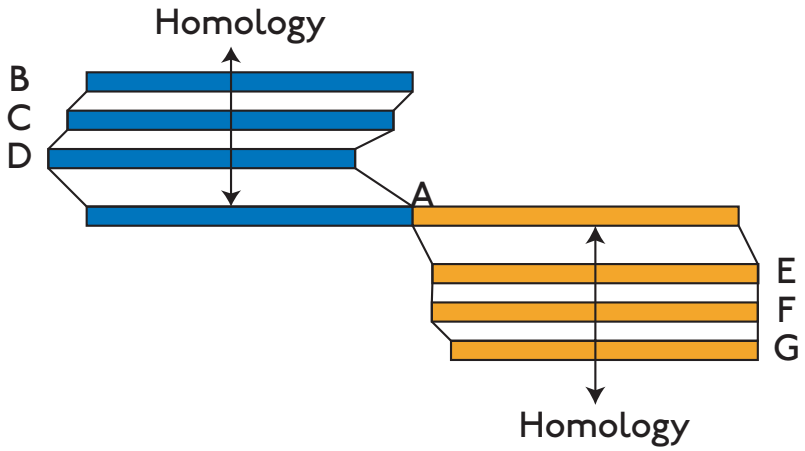


Figure 2.2: Sequence alignment cartoon showing a multi-domain protein (A) and its similarities to two separate groups of related proteins (shown in blue and orange).

both clusters. This represents a more realistic clustering solution in terms of capturing protein function.

Apart from these relatively large, independently folded, protein domains, it has been realised that smaller, quite widespread protein modules exacerbate the problem even further (Doolittle and Bork, 1993). Many proteins sharing these so-called 'promiscuous domains' (e.g. SH2, WD40, DnaJ) (Marcotte et al., 1999a) are known to have very different functions. Proteins assigned to a protein family purely on the basis of a promiscuous domain are unlikely to share a common evolutionary history with other members of that family.

Accurate clustering of protein sequences requires the detection and resolution of these domain conflicts within groups of related sequences (Enright and Ouzounis, 2000). Some of the currently available clustering methods do indeed attempt to detect and resolve domain conflicts, but these methods either require manual intervention (Tatusov et al., 1997) or are very computationally expensive (Gracy and Argos, 1998a; Park and Teichmann, 1998). Given the increasing size of sequence datasets, domain detection methods need to be fully automatic, accurate and rapid in order to provide useful protein family information.

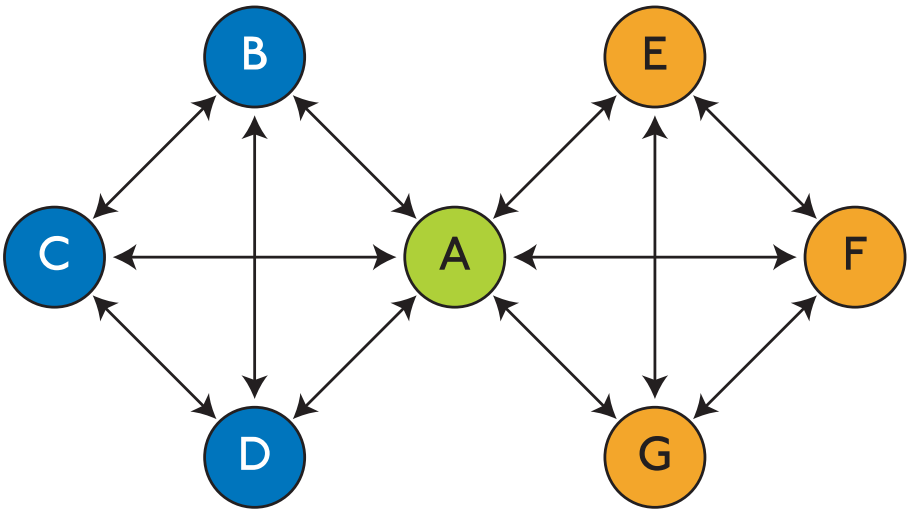


Figure 2.3: A graph representation of sequence similarities between a multi-domain protein (A) and two separate groups of related proteins (shown in blue and orange).

2.2.2 Orthology, Paralogy and Evolution

Comparative genomics has illustrated the complexity of genome evolution. It can no longer be assumed that if two proteins are homologous, then they will possess the same biochemical function. The best example of where this assumption fails, is to examine the relationships between orthologous and paralogous proteins. Orthology and paralogy relationships arise from gene duplication events. Formally orthologues and paralogues shall be defined as follows (Figure 2.4):

> Orthologues are pairs of genes related by speciation, whereas paralogues are pairs of genes related by a gene duplication event.

Gene duplication events appear to be a key process in genome evolution, and allow the creation of genes and families with novel functions (Ohno, 1970). In the previous subsection, domain rearrangements were discussed, this is also a form of gene duplication, although it only involves duplication of part of a gene (usually corresponding to a domain at the amino-acid level).

Whole gene duplication is very common and allows one copy of a duplicated gene to discover new functions through the course of evolution. A duplicate copy of a gene may become decoupled from selective pressures, and may rapidly evolve a new function. This event is rare as most dupli-

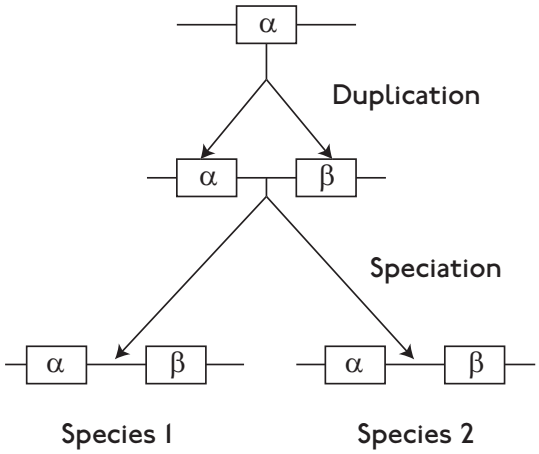


Figure 2.4: Gene duplication and speciation events: This figure shows a gene α, in an ancestral genome which has duplicated. The duplicate copy of the α gene develops a new function and is called β. Subsequent speciation events produce copies of α in different species which are related by vertical evolutionary descent (orthologues) and similarly copies of β. The α and β proteins are related to each other by the duplication event, and are termed paralogues.

cated copies which do not rapidly evolve a useful function will eventually be removed or become pseudogenes (Wolfe and Shields, 1997). Duplicate genes which survive this process have generally acquired new functions which are selectively advantageous, and are hence kept. Well known examples that have been studied include the human globin genes (Dayhoff, 1978) and homeobox genes (McGinnis et al., 1984; Scott and Weiner, 1984). The duplication of an entire genome is also possible and evidence for this has been detected in the complete genome of *Saccharomyces cerevisiae* (Wolfe and Shields, 1997).

Because of the prevalence of gene duplication, it is not always possible to assert that two homologous proteins are functionally equivalent. These genes may be orthologous (related by speciation events), and perform the same function, or they may be paralogous (related by a duplication event). Paralogous proteins may perform exactly the same function, and arose from a gene duplication event which was advantageous because it increased the production of those proteins. In most cases however, duplication events create paralogous proteins which perform similar roles, but in different tissues, or proteins which eventually perform very different functions. These complex relationships between homologous proteins also create problems for protein

sequence clustering. Accurate methods need to be able to assign proteins as being paralogous or orthologous with confidence in order to predict accurate families of proteins with common function.

This problem of detecting the evolutionary ancestry of a gene is made even more difficult by events such as non-orthologous gene displacement, horizontal gene transfer and lineage specific gene loss (Galperin and Koonin, 2000). These events can be summarised as follows:

- Non-orthologous gene displacement (Koonin et al., 1996):
  The displacement of a gene coding for a particular function, by another unrelated (or distantly related) gene with an analogous function

- Lineage specific gene loss (Koonin et al., 2000):
  The complete loss of a gene from a particular phylogenetic lineage. This can occur through deletion of the gene, or by rapid evolution of the gene to a new function, which makes it unrecognisable.

- Horizontal gene transfer: (Syvanen, 1984)
  The direct transfer of a gene or genes from one phylogenetic lineage to another. This can occur between closely related lineages, and also possibly to distant lineages.

Together these evolutionary events can make it difficult to assume two proteins are related in terms of evolution and function. Clustering techniques need to try and capture at least some of this evolutionary complexity in order to accurately classify proteins into functionally distinct families.

2.2.3 Automation and Scaling

Public protein sequence databases are growing at an exponential rate (Iliopoulos et al., 2001b). At the beginning of this research in 1998, there were 326,753 protein sequences deposited in the SwissProt databases (SwissProt & TrEMBL). Currently, there are more than twice that number of sequences deposited (668,701 sequences). Figure 2.5 shows this increase in amino-acid sequence information by year, starting with the first full peptide sequencing of insulin in 1955 by Fred Sanger and colleagues (Brown et al., 1955).

Due to this enormous growth in sequence information, algorithms that rely on protein sequence similarity data (such as clustering algorithms) now need to be fully automatic in order to reliably handle this vast amount of

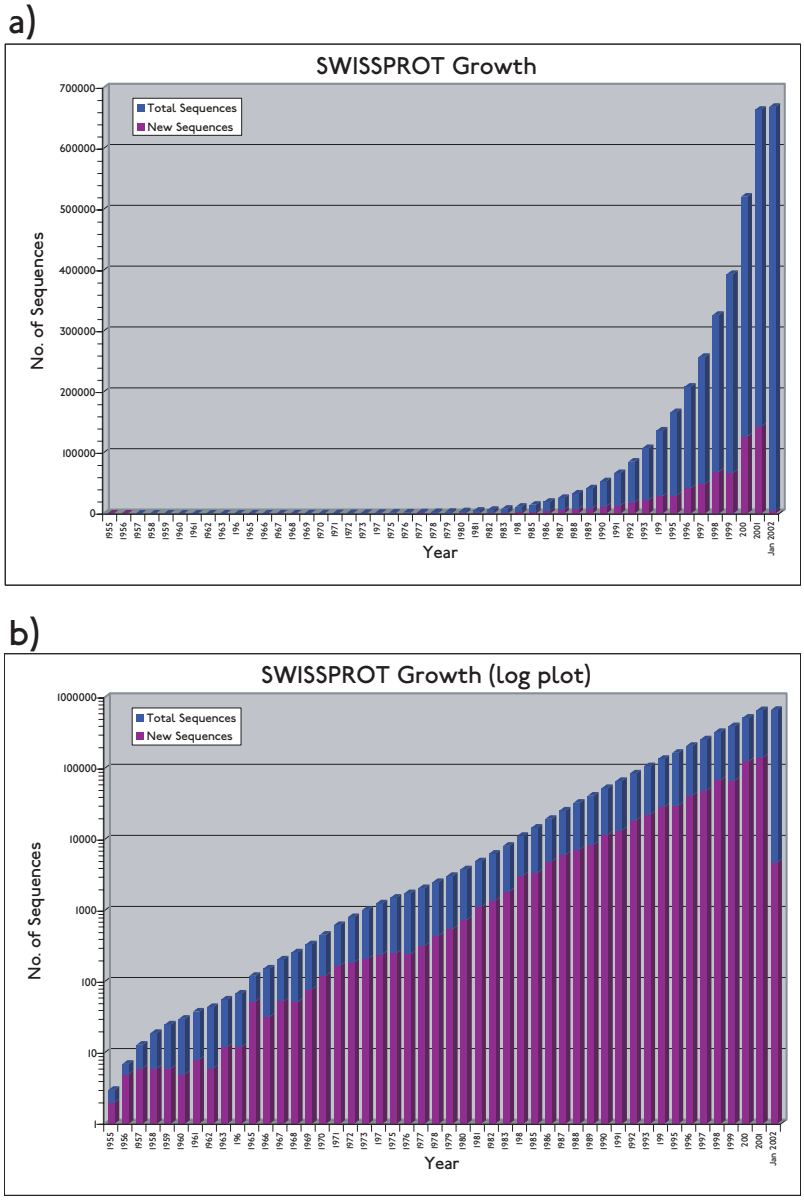


Figure 2.5: Bar charts illustrating the growth of publicly deposited amino acid sequences since the first sequenced protein in 1955. Graph (a) is a standard bar chart showing increase in both new sequences and the total number of sequences, while graph (b) illustrates this growth on a logarithmic scale. Data for the figure were obtained from the combined SwissProt and TrEMBL databases using the SRS query program *getz* (Etzold et al., 1996).

data. Many previous clustering algorithms relied on user intervention to correct clustering problems. This is however, only feasible for very small subsets of these data. Scaling is also very important, algorithms need to be able to process these data in a realistic timeframe and at a manageable computational cost. Design and implementation of algorithms for protein sequence clustering described in this thesis, has been heavily influenced by these two factors.

2.2.4 Previous Methods

In this section a selection of previous methods for protein sequence clustering will be described. Many of these methods attempt to tackle problems associated with multi-domain proteins and complex evolutionary events. However none of these methods manages to tackle the most important problems in a rapid and automatic manner. The methods described here will cover the breadth of different techniques available, many other methods similar to these also exist but will not be described in detail here.

- The Darwin System (Gonnet et al., 1992):

  This analysis involved the first exhaustive match of the entire protein sequence database. Protein sequences are first indexed using a *patricia tree* construction which stores all related sets of amino-acid subsequences for all proteins. This indexing procedure allows the selection of related sets of sequences from sub-trees. The Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) is then applied to related sequences from subtrees until an optimal alignment of subsequences is generated. Without the indexing step this analysis would have been impractical as it would have required an all-against-all matching step with Needleman-Wunsch requiring 10^6 years of CPU time.

  The Initial alignments are then used to generate a new mutation matrix (the widely used *GONNET* matrix), which is then used to further refine alignments which are stored in a hierarchical database according to the PAM (point accepted mutation) distances between proteins.

  While this large-scale clustering of protein sequences paid particular attention to automation and algorithmic scaling, no attention is given

to accurate resolution of evolutionary issues such as orthology and paralogy, or multi-domain protein resolution.

- Clusters of Orthologous Groups (COGs) (Tatusov et al., 1997):

  This approach was one of the first to examine the relationships between orthology and paralogy in protein sequence datasets. The prime motivation of the method is to detect groups of proteins with identical or highly similar functional roles (orthologue clusters), from bacterial genomes. The results from this ongoing analysis of complete genomes are available from the COGs[1] database at the United States National Centre for Biotechnology Information (NCBI).

  The method defines a rather simple notion of orthology, yet appears to be quite effective, at least for bacterial genomes. The orthologue of a protein A in genome α is defined as a protein B in another genome β which is more similar to protein A than any of the other proteins in genome β. This relationship is known as the *best-hit* criterion.

  Complete genome sequences are compared to themselves and to every other genome using BLAST. The best-hit of every protein in every other genome is stored. These hits are then analysed for relationships where three proteins from different genomes are each others *best-hits*. These triangular triplet relationships are then extended by linking together all orthologue triplets which share a common edge. The fully extended sets of orthologue triplets are called *orthologue clusters*.

  Clusters should contain only orthologues using this method, but differential gene-loss in different lineages makes this assumption invalid, as the best-hit criterion can not always guarantee that two proteins are orthologous. For this reason some clusters will contain both orthologues and paralogues. These clusters are inspected manually for such inconsistencies and broken down where required. Inconsistencies produced by multi-domain relationships are also handled in this manner.

  While this method attempts to resolve the orthology/paralogy relationships between proteins, it requires manual intervention to work correctly. Ideally such a method should be fully automatic in order to handle the large volume of protein sequence data that are currently

[1] http:/www.ncbi.nlm.nih.gov/COG/

available. For small scale bacterial analyses the hand-curated COG database is however very useful.

- ProtoMap (Linial et al., 1997):

  The ProtoMap[2] method is a well defined and robust method for classifying proteins into functionally distinct groupings. The method is intrinsically hierarchical and can generate superfamily, family and subfamily clusterings. The method uses a novel approach that eliminates the need for domain decomposition, and clusters proteins sequences based on similarity information obtained from the Smith-Waterman algorithm. While robust and accurate, the method is heavily computationally intensive.

  The method begins by taking an input set of protein sequences to be clustered. These sequences are broken down into 50 residue segments. This is done in an attempt to alleviate the effects of multi-domain relationships between proteins, as peptide fragments are clustered and not full-length proteins. The Smith-Waterman dynamic programming alignment tool then generates a similarity measure between each pair of segments. This similarity is used as a distance metric to embed all similarities into Euclidian space. This is achieved by randomly selecting sets of protein segments, building a distance vector for each segment against all other segments, and embedding each vector into Euclidian space. This embedded space is analysed and a clustering model of the sequences is constructed. The model is intensively analysed to avoid over-fitting noise in the input set. This is achieved using a method similar to the common *delete-half jack-knife* approaches. These data are split into two random subsets and reclustered. At any given level clusters are built only if the reclustering results agree with the initial clustering.

  A database of protein families was constructed using this method by taking 38,000 SwissProt (Bairoch and Apweiler, 2000) proteins and splitting them into over 500,000 segments which are subsequently clustered using the method described above, into a hierarchical database of protein families.

[2] http://www.protomap-old.cs.huji.ac.il/

While this method appears to be accurate and uses a solid statistical and algorithmic framework, it is too computationally expensive to be realistically used for rapid generation of protein families. While the method attempts to avoid problems with multi-domain proteins, by splitting proteins into 50 amino acid segments it is not clear how well this step actually performs with domain repeats and complex domain structures.

- DOMO (Gracy and Argos, 1998b; Gracy and Argos, 1998a):

  The DOMO database also represents protein families but is constructed using some novel approaches for both domain decomposition and sequence clustering.

  An initial grouping of a set of protein sequences is obtained using measures of their amino acid composition. Single residue and dipeptide compositions are calculated for each protein, and proteins with highly similar compositions are selected. When such a set of proteins exhibits more than 65% identity, a single protein is selected to represent all proteins within that set.

  The set of proteins that are deemed to be similar (at least at the level of amino acid composition) are further analysed for local similarities. All sequences are compiled into an amino acid suffix tree which groups proteins with similar amino acid subsequences together. A depth-first search through this tree detects local subsequence similarities between sequences, which are extended using dynamic programming approaches and scored for statistical significance.

  Each protein is then passed through a domain decomposition step that breaks sequences down according to their domain structures. This decomposition step behaves as follows:

  - Detected overlapping pairwise similarities between sequences are clustered into *anchors*.
  - The intersection of these sets of domain anchor points is calculated to determine the correct start and end points of each candidate domain.
  - Candidate overlapping sequence fragments are weighted according to their relative similarities, and a multiple alignment is generated.

Method	Domains	Orthology	Automation	Speed
Darwin	○ ○ ○	● ○ ○	● ● ●	● ● ○
COGs	● ○ ○	● ● ○	● ○ ○	● ● ●
ProtoMap	● ● ○	● ○ ○	● ● ●	● ○ ○
Domo	● ● ●	○ ○ ○	● ● ●	● ○ ○
Domainer	● ● ●	○ ○ ○	● ● ●	● ● ○
Geanfammer	● ● ○	○ ○ ○	● ● ●	● ● ○
Fingerprints	● ● ●	● ○ ○	● ● ○	● ○ ○

Table 2.1: A general comparison of different clustering methods. Methods are ranked based on the following characteristics a) ability to detect protein domains b) ability to detect orthology and paralogy relationships c) degree of automation d) speed of algorithm. These methods are ranked in an arbitrary manner and their relative performance is scored on a scale of zero to four represented by filled or empty circles. Three filled circles represents the highest score possible and three empty circles represents the lowest score.

- These domain alignments are extended if possible towards the N and C termini of each sequence.

Now that sequences have been split according to domain structure, the resulting sets of local alignments between groups of protein domains are assembled into rudimentary multiple alignments. These alignments are refined using a fast hierarchical dynamic programming algorithm which allows alignment gap insertion. Finally a set of sequence profiles for each detected family is generated based on local similarities from the initial search, and multiple alignments.

The Domo system is also an elegant and novel approach to protein sequence clustering, however it also suffers tremendously from the amount of computational effort required to generate local similarities and multiple alignments.

Unfortunately none of these methods effectively tackles all of the main protein sequence clustering problems. A comparison of these methods is shown in Table 2.1. Methods in this table have been rated according to their ability to handle multi-domain proteins, solve orthology/paralogy relationships, level of automation and their relative speed.

2.3 GeneRAGE

Clearly previous methods for protein sequence clustering fail to tackle many of the most important clustering problems in a reliable and automatic fashion. Given the usefulness of protein families for genome comparison and the functional annotation of proteins it was decided to write novel sequence clustering algorithms which attempt to tackle these problems.

We sought to learn from the mistakes and successes of previous clustering methods, and to develop a simple yet elegant algorithm for protein sequence clustering. To this end the GeneRAGE algorithm (Enright and Ouzounis, 2000) was initially developed, primarily for analysis of bacterial genomes. The algorithm pays particular attention to the detection and resolution of protein domains, and also the correction of false-positive and false-negative similarity assignments from the initial sequence similarity data, a step that is generally overlooked. The algorithm will be fully described in the following sections. A graphical flowchart of the algorithm is also shown in Figure 2.6.

2.3.1 Initial Steps

The input for the algorithm is a query database Q containing N protein sequences within which we seek to identify families of related proteins. For our purposes this set of protein sequences is generally obtained from one or more complete genomes, but any arbitrary set of protein sequences may be used. An *all-against-all* sequence similarity search is undertaken to determine significant similarity relationships within the query database Q. In most cases the BLASTp (Altschul et al., 1990) algorithm is used to determine similarity relationships between proteins, below a specified E-value cutoff ($E \leq 1 \times 10^{-10}$). All query sequences are filtered using the CAST algorithm (Promponas et al., 2000) prior to searching, to mask compositionally biased regions in these proteins. The CAST algorithm is described in detail in Section 5.1. The filtering of sequences using the CAST algorithm reduces noise in the sequence similarity search and makes BLAST E-values more reliable for sequence clustering (Altschul et al., 1994).

The BLAST family of algorithms are a good basis for similarity detection as they are relatively fast and sufficiently accurate to provide a solid basis for genome sequence clustering. It is possible, however, to use other search tools for this step. For example, one could use PSI-BLAST or HMMER

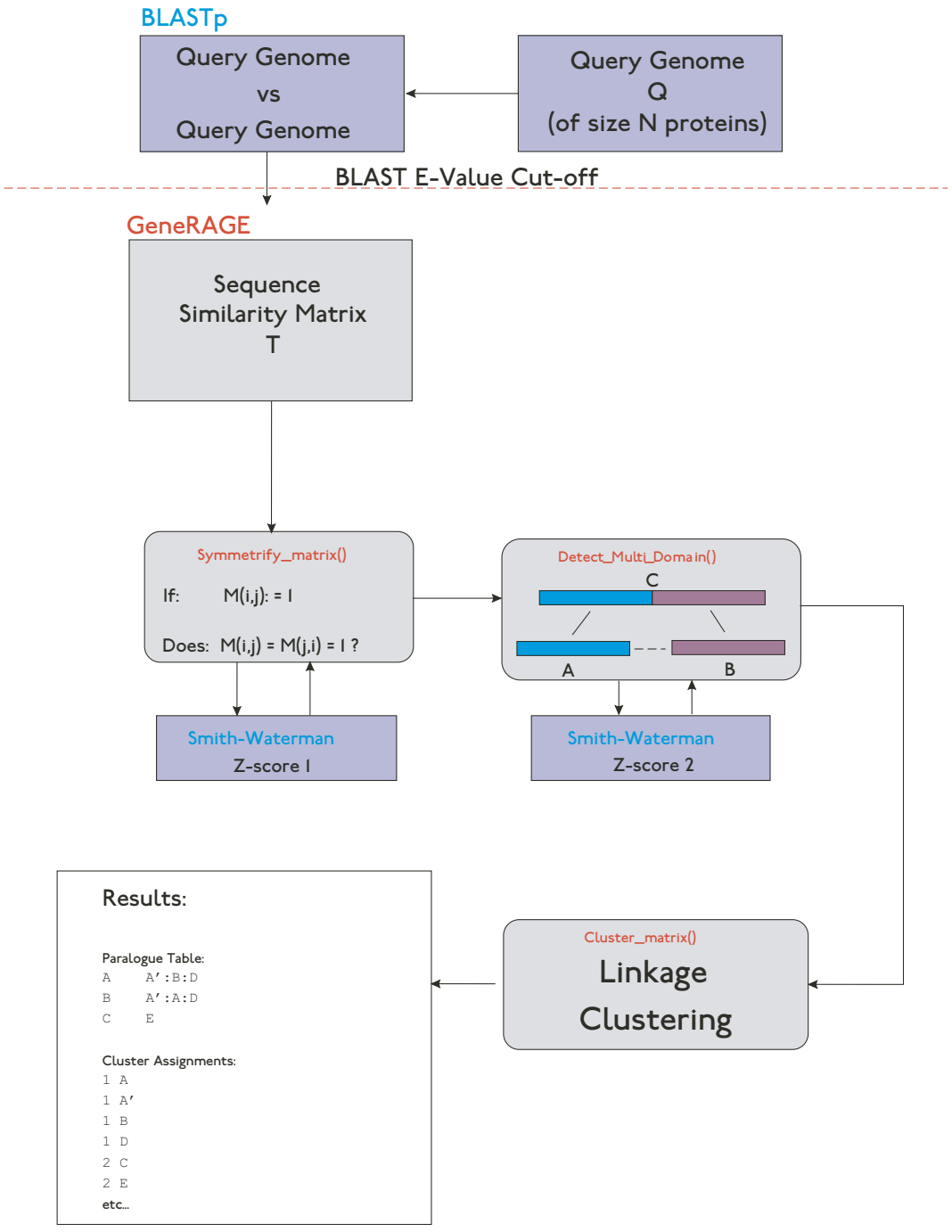


Figure 2.6: Flowchart of the GeneRAGE algorithm.

(Altschul et al., 1997; Eddy, 1995), if more distant homology relationships are required. A bitwise matrix (T) of size $N \times N$ elements is constructed from these pairwise similarity relationships. Each bit in the matrix represents either the presence ('1') or absence ('0') of significant similarity between any two proteins in the database.

2.3.2 Symmetrification of the Matrix

To facilitate clustering, the first step undertaken is the symmetrification of the similarity matrix T. This condition has been previously used in sequence comparison (Rivera et al., 1998), but is frequently overlooked in many other analyses. Accurate symmetrification of the similarity matrix reduces noise in the input data, and allows for a more consistent analysis of these data (Enright and Ouzounis, 2000). The symmetrification procedure is implemented in GeneRAGE as follows:

For every element of the matrix $T_{i,j}$ check:

$$\begin{aligned} \text{If:} \quad &\Rightarrow \quad \forall T_{i,j} : T_{i,j} = T_{j,i} \\ &\Rightarrow \quad \text{Then: Skip} \end{aligned}$$

$$\begin{aligned} \text{Else If:} \quad &\Rightarrow \quad \forall T_{i,j} : T_{i,j} \neq T_{j,i} \\ &\Rightarrow \quad \text{Then: Confirm and Correct} \end{aligned}$$

Any asymmetry detected by this method is generally the result of either a false-positive or false-negative similarity assignment. In order to determine which case is occurring a Smith-Waterman dynamic programming alignment (Smith and Waterman, 1981) with randomisation (Pearson, 1996) is used to determine whether this is a false-positive or a false-negative assignment. If a significant Z-score (e.g. $Z \geq 10$), obtained with a further 100 rounds of randomised alignments, is detected between proteins i and j, then the matrix is corrected by setting:

$$T_{i,j} = T_{j,i} = 1$$

This situation represents a false-negative case from the initial BLAST search step that is rectified at this symmetrification step.

Otherwise, if no significant similarity is detected (e.g. $Z < 10$) the matrix is corrected by setting:

$$T_{i,j} = T_{j,i} = 0$$

This situation represents a false-positive case from the initial BLAST search that is eliminated at this step. When this procedure is complete, matrix T satisfies the symmetrical properties of a sequence similarity matrix:

$$\Rightarrow \text{Now: } \forall T_{i,j} = T_{j,i}$$

2.3.3 Detection of Multi-Domain Proteins

The detection of multi-domain proteins within the query database is important to allow accurate clustering of the matrix (Figures 2.2 & 2.3). Multi-domain proteins are detected by the following simple, yet effective protocol. If two proteins a and b hit a common protein c, does protein a hit protein b? This is shown in Figures 2.6 & 2.7. In other words, does the transitivity criterion hold?

For every element of the matrix $T_{i,j}$ check:

$$\begin{aligned} \text{If:} \quad &\Rightarrow \quad T_{a,c} = T_{c,a} = 1 \\ \text{and} \quad &\Rightarrow \quad T_{b,c} = T_{c,b} = 1 \end{aligned}$$

$$\begin{aligned} &\text{Then:} \\ \text{Check If} \quad &\Rightarrow \quad T_{a,b} = T_{b,a} = 1 \end{aligned}$$

If this is not the case then c may be a multi-domain protein (Figure 2.7). The multi-domain detection algorithm works as follows:

- For each protein c in matrix T, collect the set S_c of all proteins that exhibit significant similarity to protein c, from matrix T.
- For every pair of proteins (a, b) in the set S_c, look up matrix T to check if any similarity exists between them.

$$\begin{aligned} \text{If:} \quad &\Rightarrow \quad T_{a,b} = T_{b,a} = 1 \\ &\Rightarrow \quad \text{Then Skip:} \end{aligned}$$

Else If: $\Rightarrow$ $T_{a,b} = T_{b,a} = 0$

$\Rightarrow$ Then Confirm:

If no similarity has been detected between a and b there are two possibilities (Figure 2.7): *Either* protein c is a multi-domain protein *or* proteins a and b are similar, but the initial BLAST similarity search failed to detect a similarity. These situations can be resolved by performing an additional Smith-Waterman dynamic programming alignment between a and b. If significant sequence similarity is detected (e.g. $Z \geq 10$), this is a false negative case that is corrected by setting $T_{a,b} = T_{b,a} = 1$. In this case $T_{a,c} = T_{c,a} = 1$ and $T_{b,c} = T_{c,b} = 1$ already hold, therefore a, b and c belong to the same family. If no significant similarity is detected (e.g. $Z < 10$), mark protein c as a candidate multi-domain protein, composed of two domains a,b with similarity to a and b respectively (Figure 2.7).

Another important application for this technique is the detection of fusion proteins across genomes (Enright et al., 1999). This will be described more fully in Chapter 4. In these cases, proteins a and b represent component proteins in one genome and protein c represents a multi-domain composite protein in another genome with two domains a and b similar to a and b respectively. The fusion detection algorithm (called DifFuse) is a variant of GeneRAGE, where the second dynamic programming test is performed between entries from two databases.

2.3.4 Clustering the Similarity Matrix

The processed matrix is recursively clustered by beginning a clustering operation for each row of the matrix T. If a protein corresponding to a given row i of matrix T is not already clustered, then a new cluster is created containing sequence i. New sequences are added to this cluster by processing across row i of the matrix and recursively subclustering each protein that is hit by protein i. As the clustering procedure descends through each row of the matrix, increasing numbers of proteins are added to each cluster. At this stage, multi-domain proteins are clustered separately from single-domain proteins. When the initial clustering operation is complete, multi-domain family information from the second step of the algorithm is used to split clusters.

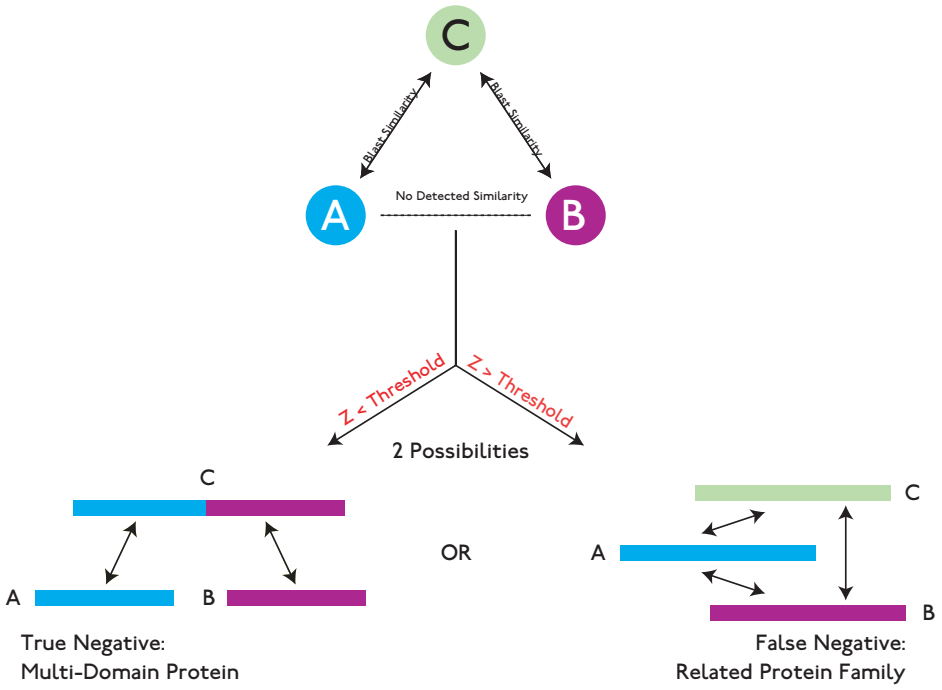


Figure 2.7: GeneRAGE multi-domain detection step: Protein a is similar to protein c, and b is similar to c. If a is not similar to b, then only two possibilities exist. Either protein c is a multi-domain protein and no physical similarity exists between a and b (bottom left). Or alternatively, a true similarity between a and b has not been detected representing a false-negative similarity assignment (bottom right).

Clusters that are deemed to contain two separate families linked by one or more multi-domain proteins are split into their constituent families. Multi-domain proteins can hence be members of more than one cluster. Finally, all clustering information (including multi-domain information) is represented in a clusters table (Figure 2.6). Additionally, for genome comparison studies, a similarity table is created for further analysis (Figure 2.6).

2.3.5 Validation and Testing

To evaluate the performance of the algorithm, the complete genome sequence of the archaeal methanogen *Methanococcus jannaschii* was analysed using GeneRAGE. This particular genome was chosen because of our previous extensive experience with this organism and the availability of updated and highly accurate manual annotations for all genes. The initial genome self-comparison using BLASTp (Altschul et al., 1997) took approximately 20 minutes running in parallel on four workstations (using *htBLAST.pl* which is described in Section 5.5). Symmetrification, multi-domain detection and clustering for all 1,771 proteins, took eight minutes on a two processor R10000 SGI Octane workstation. The analysis was conducted using a BLASTp E-value of $E \leq 1 \times 10^{-6}$ and the CAST filter. Z-score cutoff values of 10 (symmetrification) and 7 (multi-domain detection) were used by GeneRAGE. These values are arbitrarily set by the user, and have been set to the above mentioned values based on experimentation and empirical observations.

Of the original 3,391 hits obtained by BLAST, 1,026 were considered to be false-positives and were removed, while 889 were considered as false-negatives and were added, during the symmetrification step. The total number of hits after the processing of the similarity matrix was 3,254.

Most proteins in *M. jannaschii* (69%) have no paralogues in the genome and hence clustered as individual sequences. Other clusters of varying sizes were formed from related sequences within the genome. The distribution of these cluster sizes is illustrated in Figure 2.8. Multi-domain proteins detected within the genome were clustered correctly according to their individual domain architectures. Clusters containing more than three members were analysed to examine their validity (61 clusters). This analysis was performed by taking each protein in a cluster and examining its corresponding annotation from the *M. jannaschii* functions database (Kyrpides et al., 1996). In addition, multiple alignments (Thompson et al., 1994) were created to assist in

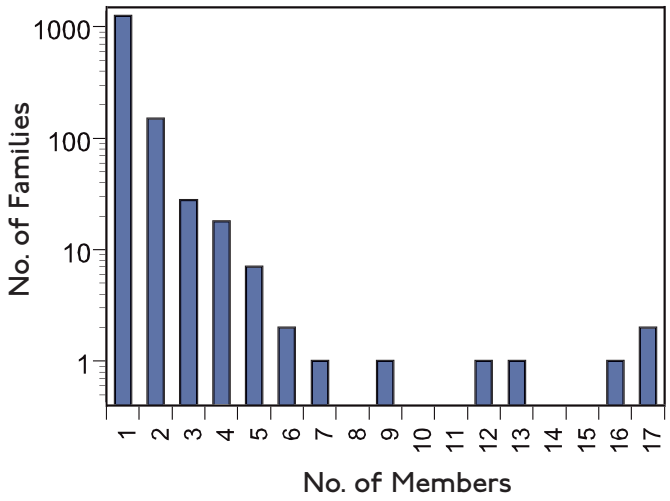


Figure 2.8: Distribution of detected protein families in the *M. jannaschii* genome. The x-axis shows the family size (number of members) and the y-axis shows the distribution of detected families of these sizes.

the evaluation of clusters if needed. Of these 61 clusters, 95% (58/61) had manual annotations that were consistent with that cluster. The other three clusters had consistent, high quality alignments but conflicting annotations. These cases may represent incorrect annotations, functionally diverse families or false-positive cases. Multi-domain proteins detected within the context of this dataset proved to be consistent with the manual annotations (Kyrpides et al., 1996) and further multiple alignment analysis (Thompson et al., 1994). One example is archaeal ATPase proteins (Koonin, 1997), which were successfully resolved into three distinct domains. Further examples of successful multi-domain detection include ABC transporter proteins, hydrogenases and dehydrogenases. Even domains as short as CBS (Bateman, 1997) or TPR (Kyrpides and Woese, 1998) were detected and assigned to consistent clusters within the *M. jannaschii* genome.

To further examine and validate the multi-domain detection algorithm, a complex test set containing multi-domain proteins was used. This set contained complex multi-domain relationships that are not present in the *M. jannaschii* genome described above. The test set consisted of 13 peptides from the aromatic amino acid biosynthesis (*aro*) operon from four different genomes (*S. cerevisiae*, *E. coli*, *H. influenzae* and *M. jannaschii*). In yeast, all proteins from this operon have fused into a single pentafunctional pep-

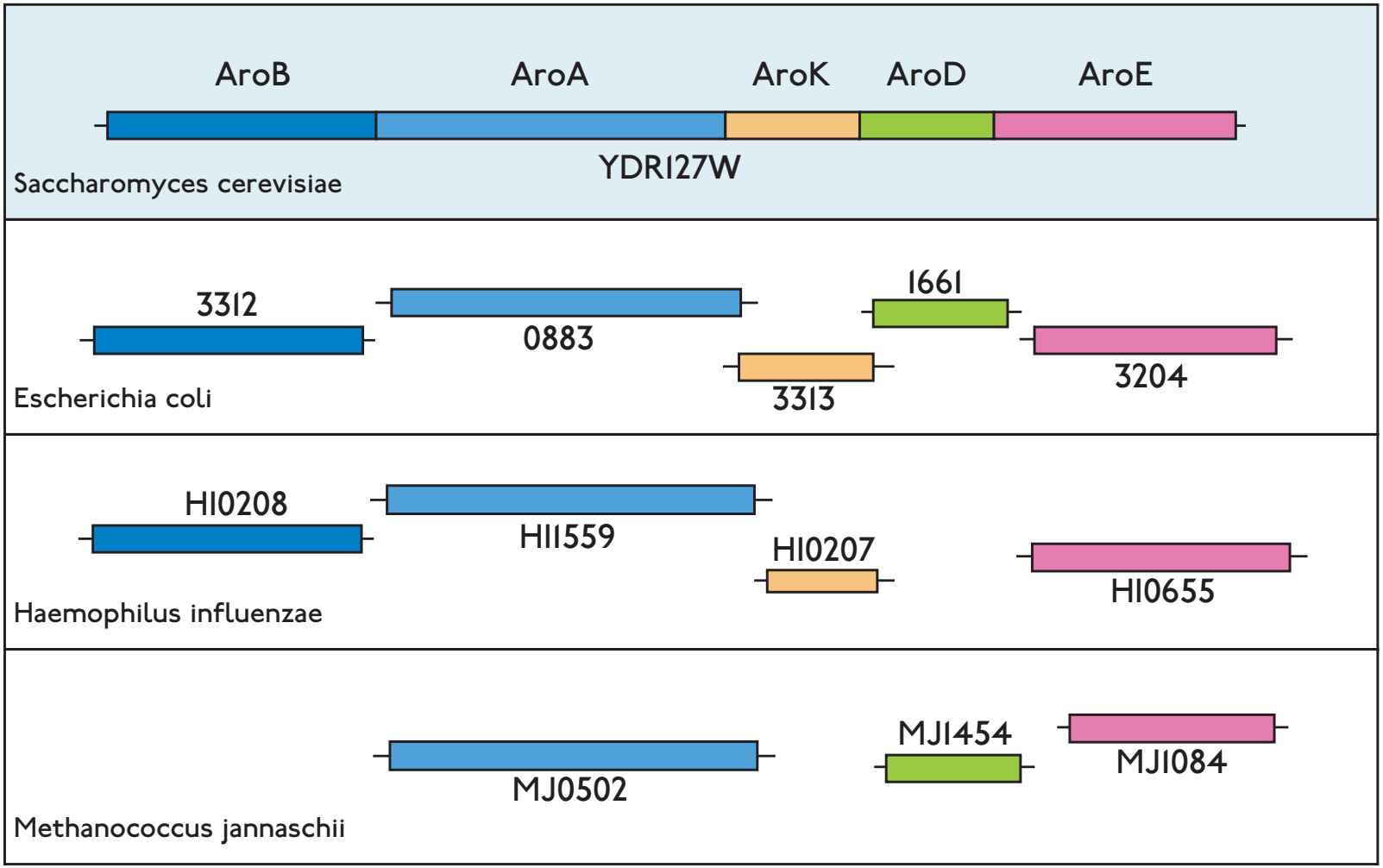


Figure 2.9: Organisation of the Bacterial *aro* Gene Cluster in four different organisms. The entire operon is encoded by a single fused gene in *S. cerevisiae*, and by single genes in the three other organisms.

tide, while in bacterial genomes, the operon is composed of multiple separate peptides. Figure 2.9 illustrates the arrangement of these proteins in four different genomes. Conventional clustering techniques generally fail to cluster individual proteins from each genome into the correct clusters. We are not aware of any sequence clustering technique that can perform the above task automatically at this level of precision. GeneRAGE successfully detected the presence of multi-domain proteins in this test set and divided clusters accordingly. Each cluster generated hence represented a single functional unit or family. The same result can be reproducibly obtained from the clustering of their corresponding complete genome sequences.

2.3.6 Algorithm Implementation

The GeneRAGE algorithm is written in ANSI C and was developed on a Sun Ultra 10 Workstation. The code has been ported to the following operating systems: Solaris, Compaq Tru64 UNIX, SGI IRIX, AIX and Linux. An SMP parallel implementation of the code is also available (based on the POSIX Pthread standard), and has been tested in Linux, SGI IRIX and Compaq Tru64 multiprocessor environments. The minimum hardware requirements for the clustering of small genomes ($<$ 6000 proteins) is 32MB RAM and sufficient disk space to store the sequence database and search results.

2.3.7 Conclusions

The GeneRAGE method represents a fast and efficient method for clustering protein sequences according to similarity. The algorithm has been designed for the clustering of protein sequences within and between complete bacterial genomes. We believe however, that the algorithm also has wide ranging uses for the clustering of protein sequences in general.

The key abstractions made by the algorithm are the representation and symmetrification of sequence similarity information in a binary matrix, and the subsequent detection of multi-domain proteins (Figures 2.6 and 2.7). The symmetrification step is an important abstraction as it not only detect false-positive/ false-negative similarity assignments, but also provides a consistent similarity construct for further processing and analysis. The storage of similarity information as binary relationships in the matrix makes the algorithm more efficient and less memory intensive. This allows the analysis of large datasets (up to 50,000 sequences).

Given the fully automatic implementation of this algorithm and its precision, we believe that the algorithm represents a significant improvement over currently available clustering techniques. The multi-domain detection step, although based on a simple abstraction is a significant improvement over data-driven methods. Because multi-domain proteins are detected due to inconsistencies in the similarity matrix, this step not only detects multi-domain proteins, but also detects and corrects further false-negative similarity relationships. The precision of this step has been demonstrated in many cases. The detection of multi-domain proteins is however, highly dependent on the cut-off scores specified. Multi-domain proteins containing two domains that are not similar are relatively easy to detect. The difficulty lies in the detection of multi-domain proteins that contain two or more very similar domains. Future research may endeavour to provide ways of dynamically modifying the cut-off values used in the multidomain detection step, allowing more efficient detection of these cases.

The algorithm was designed for the clustering of peptides from bacterial genomes, and these validation results indicate that clustering of this sort is both fast and reliable. The multi-domain detection step is the slowest part of the algorithm due to its iterative use of the Smith-Waterman algorithm. When GeneRAGE is applied to proteins from complete Eukaryotic genomes, such as *Caenorhabditis elegans*, the method may become very computationally intensive. This is due to the complex domain architecture of eukaryotic proteins, which require multiple recursive rounds of Smith-Waterman correction in the multi-domain detection step of the algorithm. For this reason, we have developed another algorithm for sequence clustering and protein family detection in Eukaryotic genomes. This algorithm (Tribe-MCL) will be described in detail in the following sections of this chapter.

The GeneRAGE package is freely available for protein sequence clustering[3]. Although the algorithm is still a relatively new arrival in the field, a number of research groups have been actively using the algorithm for protein family analysis and research. Recently a number of these analyses have been published (Ward, 2001; Janssen et al., 2001; Boucher et al., 2001), and we hope that the method continues to prove useful for protein sequence classification.

[3]http://www.ebi.ac.uk/research/cgg/services/rage/

2.4 Tribe-MCL

In the previous section we described the GeneRAGE algorithm for the accurate detection of protein families in bacterial genomes. This algorithm has proved very successful for protein family analysis, however the algorithm suffers, both in terms of computational complexity and accuracy, due to the complexity of eukaryotic domain architectures, and the highly paralogous nature of these genomes. For this reason we have developed a novel and complementary approach called Tribe-MCL (Enright et al., 2002) for rapid and accurate clustering of protein sequences from very large eukaryotic datasets, into families. Experiments using the BioLayout algorithm (see Section 5.2) illustrated the power of graph-based visualisation of protein sequence similarities, to overcome problems associated with complex domain structures of proteins. To this end we have reapproached the problem of sequence clustering using a graph-based representation of sequence similarities.

The method relies on the Markov Cluster (MCL) Algorithm (van Dongen, 2000b) for the assignment of proteins into families based on precomputed sequence similarity information. This novel approach does not suffer from the problems that normally hinder other protein sequence clustering algorithms, such as the presence of multi-domain proteins, promiscuous domains and fragmented proteins. The method has been rigorously tested and validated on a number of very large databases, including SwissProt, InterPro, SCOP and the draft human genome. Our results indicate that the method is ideally suited to the rapid and accurate detection of protein families on a large scale. The method has been used to detect and categorise protein families within the draft human genome and the resulting families have been integrated into the Ensembl[4] database (Hubbard et al., 2002).

2.4.1 Introduction

Despite significant progress in the field of sequence clustering, new challenges have emerged due to the availability of large eukaryotic genomes, in terms of their size and complexity (Birney et al., 2001). In particular, eukaryotic protein families constitute a bottleneck for most methods. Many eukaryotic proteins contain large numbers of protein domains (Hegyi and Bork, 1997; Apic et al., 2001b), each of which needs to be detected and resolved by an effi-

[4]http://www.ensembl.org/

cient clustering algorithm. Iterative automatic domain detection algorithms such as GeneRAGE suffer from an excessive and unpredictable number of additional sequence comparison steps, which renders them somewhat impractical when using modest computational resources. Another approach is to detect proteins with very similar domain architectures (Apic et al., 2001a), rather than attempting to detect each domain individually. The assumption is that proteins with near-identical sets of domains may have very similar biochemical roles (Hegyi and Gerstein, 1999; Ouzounis and Karp, 2000).

The GeneRAGE algorithm was developed and tested on protein families within relatively small datasets, such as prokaryotic genomes (Janssen et al., 2001). Given such datasets, the algorithm effectively and accurately identifies protein families and also correctly detects multi-domain proteins (Coulson et al., 2001). When the algorithm is applied to larger datasets, such as those obtained from eukaryotic organisms, some of the previously mentioned problems become apparent. The detection of protein domains using GeneRAGE becomes hampered to a large extent by promiscuous domains, peptide fragments (representing incomplete database entries) and proteins of complex domain structure. Domains such as the *response regulator* domain from two-component systems (Stock et al., 2000) cause proteins with vastly differing functions (such as heat shock factors and phytochromes) (Chang and Meyerowitz, 2001) to be assigned incorrectly to the same family (Yeh et al., 1997).

Given the difficulty of detecting such domains accurately and the ever-increasing amount of eukaryotic data available, we have approached this problem using an elegant mathematical approach based on probability and graph flow theory. Sequence similarity search algorithms have previously benefited from such approaches, for example hidden Markov model (HMM) based search algorithms provide very sensitive detection of distant protein sequence similarity (Eddy, 1998). An ideal method, in this case, would require sequence similarity relationships as input and be able to rapidly detect clusters solely using this information, without being led astray by the complex modular domain structure of eukaryotic proteins. Traditionally, most methods deal with similarity relationships in a pairwise manner, while graph theory allows the classification of proteins into families based on a global treatment of all relationships in similarity space simultaneously. To this end, we have developed the Tribe-MCL algorithm as an efficient and reliable method for protein sequence clustering (Enright et al., 2002). Tribe-MCL

is based on the Markov Cluster (MCL) algorithm, previously developed for graph clustering using flow simulation (van Dongen, 2000a). This approach for protein sequence clustering is extremely fast and appears to be highly accurate. It avoids most of the problems mentioned above and has already been successfully utilised for the clustering of large datasets and the classification and annotation of proteins from the draft human genome (Lander et al., 2001; Hubbard et al., 2002).

2.4.2 Markov Clustering of Sequence Similarities

This section will discuss the process of protein similarity graph clustering using flow simulation with the Tribe-MCL algorithm. A flowchart of the algorithm is shown in Figure 2.10.

Data representation

Sequence similarity relationships within a given protein dataset can be represented as a square matrix, whose elements contain similarity metrics for any pair of proteins in the dataset. These elements can be binary numbers (Enright and Ouzounis, 2000) or real numbers, such as E-values from BLAST (Altschul et al., 1997). Alternatively, this matrix can be considered as a weighted graph, whose nodes (vertices) represent proteins and connections (edges) represent similarity relationships (Figure 2.11a). The BioLayout algorithm (Enright and Ouzounis, 2001a), which is described in Section 5.2, has illustrated that such graphs are an elegant and concise way of representing sequence similarity relationships. Furthermore, these graphs are amenable to graph clustering algorithms, developed in the fields of mathematics and computer science. Such algorithms include single-linkage clustering and k-means (Michalski et al., 1998), with which we have extensively experimented, before choosing the Markov Cluster (MCL) algorithm, because of its relevance, elegance and efficiency.

The MCL Algorithm

The Markov Cluster (MCL) algorithm is an algorithm designed specifically for the settings of simple graphs and weighted graphs (van Dongen, 2000b). It has previously been used in the field of computational graph clustering (van Dongen, 2000a; van Dongen, 2000c; van Dongen, 2000d). Given that it

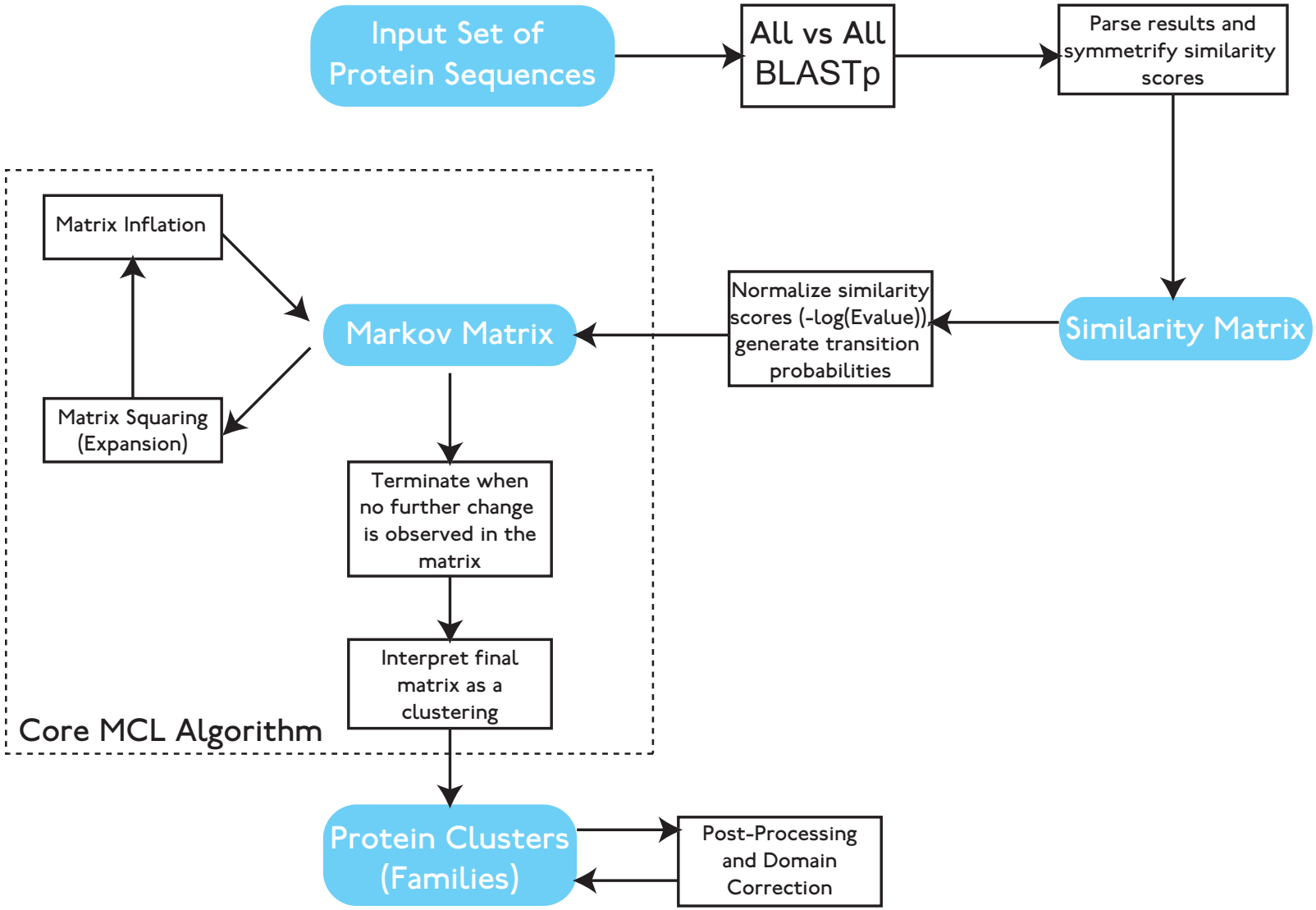


Figure 2.10: Flowchart of the Tribe-MCL algorithm.

is possible to represent biological sequence similarity relationships in terms of these graphs (Enright and Ouzounis, 2000; Enright and Ouzounis, 2001a), it is possible to use an algorithm such as MCL for biological sequence clustering.

Natural clusters in a graph are characterised by the presence of many edges between the members of that cluster, and one expects that the number of 'higher-length' (longer) paths between two arbitrary nodes in the cluster is high. In particular, this number should be high, relative to node pairs lying in different natural clusters. A different angle on this is that random walks on the graph will infrequently go from one natural cluster to another, based on graph transition probability estimates.

The MCL algorithm finds cluster structure in graphs by a mathematical bootstrapping procedure. The process deterministically computes (the probabilities of) random walks through the sequence similarity graph, and uses two operators transforming one set of probabilities into another. It does so using the language of stochastic matrices (also called Markov matrices) which capture the mathematical concept of random walks on a graph.

The MCL algorithm simulates random walks within a graph by alternation of two operators called expansion and inflation. Expansion coincides with taking the power of a stochastic matrix using the normal matrix product (i.e. matrix squaring). Inflation corresponds with taking the Hadamard power of a matrix (taking powers entrywise), followed by a scaling step, such that the resulting matrix is stochastic again, i.e. the matrix elements (on each column) correspond to probability values.

Definition of the Inflation operator

A column stochastic matrix is a non-negative matrix with the property that each of its columns sums to 1. Given such a matrix $M \in R^{k\times k}, M \geq 0$, and a real number r greater than one, the column stochastic matrix resulting from inflating each of the columns of M with power coefficient r is written $\Gamma_r M$, and Γ_r is called the inflation operator with power coefficient r. Formally, the action of $\Gamma_r : R^{k\times k} \longrightarrow R^{k\times k}$ is defined by:

$$(\Gamma_r M)_{pq} = (M_{pq})^r / \sum_{i=1}^{k} (M_{iq})^r$$

Each column j of a stochastic matrix M corresponds with node j of the stochastic graph associated with M. Row entry i in column j (i.e. the

matrix entry M_{ij}) corresponds with the probability of going from node j to node i. It is observed that for values of r greater than 1, inflation changes the probabilities associated with the collection of random walks departing from one particular node (corresponding with a matrix column) by favouring more probable walks over less probable walks.

Definition of Expansion

Expansion corresponds to computing random walks of 'higher length' (i.e. random walks with many steps). It associates new probabilities with all pairs of nodes, where one node is the point of departure and the other is the destination. Since higher length paths are more common within clusters than between different clusters, the probabilities associated with node pairs lying in the same cluster will, in general, be relatively large as there are many ways of going from one to the other. Inflation will then have the effect of boosting the probabilities of intra-cluster walks and will demote inter-cluster walks. This is achieved without any *a priori* knowledge of cluster structure, but simply the result of cluster structure being present. This property of the algorithm lends itself well to the problem of biological sequence comparison described later in Section 2.4.3.

Convergence and Clustering

Eventually, iterating expansion and inflation results in the separation of the graph into different segments. There are no longer any paths between these segments and the collection of resulting segments is simply interpreted as a clustering. The inflation operator can be altered using the parameter r. Increasing this parameter has the effect of making the inflation operator stronger, and this increases the granularity or 'tightness' of clusters.

Cast in the language of stochastic flow, we can state that expansion causes flow to dissipate within clusters whereas inflation eliminates flow between different clusters. Expansion and inflation represent different tidal forces which are alternated until an equilibrium state is reached.

An equilibrium state takes the form of a so-called doubly idempotent matrix, i.e. a matrix that does not change with further expansion or inflation steps. The graph associated with such a matrix consists of different connected directed components. Each component is interpreted as a cluster, and has a star-like form, with one attractor in the centre and arcs going from

all nodes of that component to the attractor. In theory, attractor systems with more than one attractor may occur (these do not change the cluster interpretation). Also, nodes may exist that are connected to different stars, which is interpreted as cluster overlap, or in other words, nodes may belong to multiple clusters (van Dongen, 2000a).

With respect to convergence, it can be proven that the process simulated by the algorithm converges quadratically around the equilibrium states. In practice, the algorithm starts to converge noticeably after three to ten iterations. Global convergence is very hard to prove; it is conjectured that the process always converges if the input graph is symmetric (van Dongen, 2000a; van Dongen, 2000b; van Dongen, 2000c). This conjecture is supported by results concerning the matrix iterands. For symmetric input graphs, it is true that all iterands have real spectrum (the set of eigenvalues), and that all iterands resulting from expansion have non-negative spectrum and are diagonally symmetric to a positive semi-definite matrix. It can be shown that these matrices have a structural property which associates a directed acyclic graph (DAG) with each of them. It turns out that inflation strengthens (in a quantitative sense) this structural property and will never change the associated DAG, whereas expansion is in fact able to change the associated DAG. This is a more mathematical view on the 'tidal forces' analogy mentioned earlier. DAGs generalise the star graphs associated with MCL limits, and the spectral properties of MCL iterands and MCL limits can be related via the inflation operator. These results imply that the equilibrium states can be viewed as a set of extreme points on the set of matrices that are diagonally similar to a positive semi-definite matrix. This establishes a close relationship between the MCL iterands, MCL limits and cluster (DAG) structure in graphs (van Dongen, 2000d).

The MCL algorithm also associates return probabilities (or loops) with each node in the initial input graph. The flow paradigm underlying MCL naturally requires this, and it can be motivated in terms of the spectral and structural properties mentioned earlier. As for the weights that are chosen, experience shows that a 'neutral' value works well. In the implementation used, 'neutral' is chosen as a weight (in principle different for each node) that will not change when the inflation operator is applied to the stochastic column associated with the node. It is possible to choose larger weights (Figure 2.11), and this will increase cluster granularity. The effect is secondary however to that of varying the inflation parameter, and the algorithm is not very sensitive

to changes in the loop weights.

A very important asset of the algorithm is its 'bootstrapping' nature, retrieving cluster structure via the imprint made by this structure on the flow process. Further key benefits of the algorithm are: a) It is not misled by edges linking different clusters; b) It is very fast and very scalable; c) It has a natural parameter for influencing cluster granularity; d) The mathematics associated with the algorithm show that there is an intrinsic relationship between the process it simulates and cluster structure in the input graph (van Dongen, 2000d), and e) Its formulation is simple and elegant.

From the definition of the MCL algorithm it can be seen that it is based on a very different paradigm than any linkage-based algorithm. One possible view of this is that MCL, although based on similarities between pairs, recombines these similarities (via expansion) and is thus affected by similarities on the level of sets (as generalising pairs). Alternating expansion with inflation turns out to be an appropriate way of exploiting this recombination property.

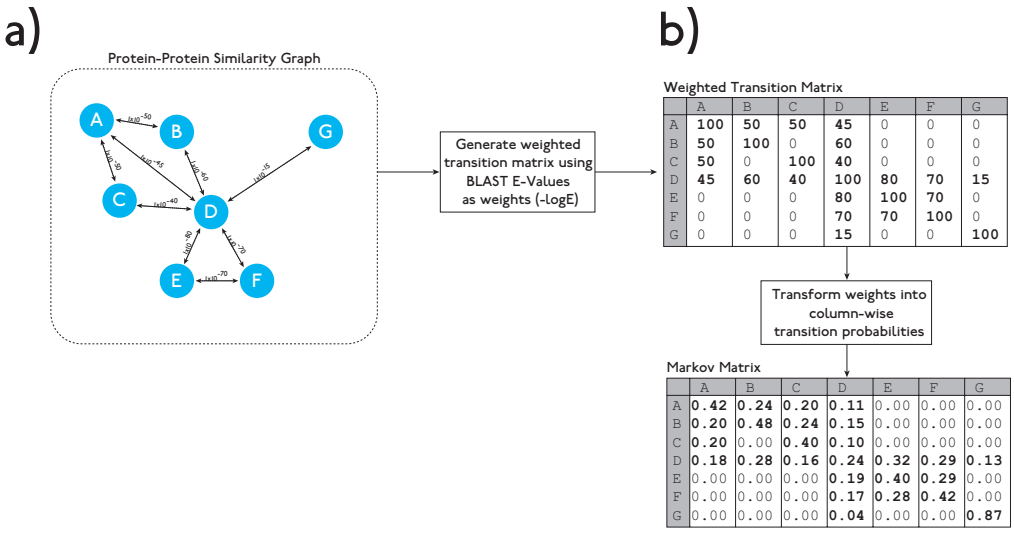
The structure of the MCL algorithm is illustrated in Figure 2.10. The algorithm sets out by computing the graph of random walks of an input graph, yielding a stochastic matrix. It then alternates the expansion operator that squares a matrix using the usual matrix product with the inflation operator. Inflation is performed by raising each matrix entry to a given power and rescaling the matrix so that it becomes stochastic again. Alternation continues until an equilibrium state is reached in the form of a so-called doubly idempotent matrix.

2.4.3 Application of the MCL Algorithm to Biological Graphs

The section above describes the MCL algorithm in a general fashion. In this section, we describe how the algorithm relates to the clustering of proteins into protein families. Biological graphs may be represented as follows (Figure 2.11a):

1. Nodes of the graph represent a set of proteins that we would like to assign to families.

2. Edges within the graph represent similarities between these proteins.



Weighted Transition Matrix	A	B	C	D	E	F	G
A	100	50	50	45	0	0	0
B	50	100	0	60	0	0	0
C	50	0	100	40	0	0	0
D	45	60	40	100	80	70	15
E	0	0	0	80	100	70	0
F	0	0	0	70	70	100	0
G	0	0	0	15	0	0	100

Markov Matrix	A	B	C	D	E	F	G
A	0.42	0.24	0.20	0.11	0.00	0.00	0.00
B	0.20	0.48	0.24	0.15	0.00	0.00	0.00
C	0.20	0.00	0.40	0.10	0.00	0.00	0.00
D	0.18	0.28	0.16	0.24	0.32	0.29	0.13
E	0.00	0.00	0.00	0.19	0.40	0.29	0.00
F	0.00	0.00	0.00	0.17	0.28	0.42	0.00
G	0.00	0.00	0.00	0.04	0.00	0.00	0.87

Figure 2.11: (a) Example of a protein-protein similarity graph for seven proteins (A-G), blue circles represent proteins (nodes) and lines (edges) represent detected BLASTp similarities with E-Values (also shown). (b) Weighted transition matrix and associated column stochastic Markov matrix for the seven proteins shown in (a).

3. Edges are weighted according to a sequence similarity score obtained from an algorithm such as BLAST.

In order to build a protein similarity graph, a FASTA file containing all sequences that are to be clustered into families is assembled. Peptides within this file are filtered using the CAST algorithm, then compared against each other using BLASTp (Altschul et al., 1997). All similarities and associated scores are used to build the protein-protein similarity graph. A Markov matrix (Figure 2.11b) is constructed, representing transition probabilities from any protein in this graph to any other connected protein. Each column of the matrix represents a given protein, and each entry in a column represents a transition probability between this protein and another protein. Diagonal elements are set arbitrarily to a *neutral* value as described previously. The entries in the Markov matrix are probabilities generated from weighted sequence similarity scores (e.g. from BLAST). A weight is assigned to each edge of a protein similarity graph by taking the average pairwise $-log_{10}$(E-Value) (Altschul et al., 1997), resulting in a symmetric matrix. This simple weighting scheme produces reliable results but other more complex schemes can also

be used (e.g. length-based weighting). This Markov matrix is supplied to the MCL algorithm. Initial expansion of the Markov matrix simulates random walks, which allow one to measure 'flow' in the graph. Areas of high flow indicate that a large number of random walks go through this area. The MCL algorithm uses iterative rounds of expansion and inflation (explained earlier) to promote flow through the graph where it is strong, and remove flow where it is weak. This process terminates when equilibrium has been reached, i.e. further rounds of expansion and inflation leave the matrix unaltered.

In a biological sense, we expect that members of a protein family will be more similar to each other than to proteins in another family. Experiments using the Bio-Layout graph visualisation algorithm (Enright and Ouzounis, 2001a), described in Section 5.2. have shown this to be true for most protein similarity graphs. Because of this property of biological graphs, flow within protein families is strong, i.e. a random walk starting at any given protein in a family is more likely to linger within this family than to cross to another family. Flow between protein families will be weaker than flow within a family as there are relatively few (if any) paths that cross two distinct protein families. Intra-family paths represent either sequence similarity relationships due to multi-domain proteins or mere false positive similarity detections. These properties of biological similarity graphs make them ideally suited to the MCL algorithm. The iterative rounds of inflation and expansion remove this weak flow across protein families, and promote the stronger flow within protein families. This boot-strapping procedure allows protein families hidden in the graph to become visible by gradually stripping the graph down to its basic components as detected by stochastic flow.

Many of the problems that normally hinder protein sequence clustering are eliminated by the Markov Clustering approach. Proteins possessing a promiscuous domain, that is present in many functionally unrelated proteins, are normally very difficult to cluster correctly. Promiscuous domains will connect a member of a given protein family to members of that family and possibly to a large number of other (possibly unrelated) protein families. Because these inter-family connections are still far fewer than intra-family connections, the algorithm gradually eliminates these inter-family similarities and detects protein families accurately. The algorithm requires no *a priori* knowledge of protein domains, and clusters proteins into families purely based on observed relationships through the entire similarity graph. However, proteins containing different domains or sets of domains will have

very different sequence similarity patterns, and hence we expect the MCL algorithm to cluster proteins with different domain structures into distinct families. We have extensively validated the performance of the algorithm in terms of speed and accuracy. We have also assessed the performance of the algorithm in terms of the quality of protein family descriptions, based on database annotations.

2.4.4 Validation of the Algorithm

In order to test the effectiveness of protein family detection using Tribe-MCL, we have performed extensive validation using the InterPro protein domain database (Apweiler et al., 2001) and the Structural Classification of Proteins (SCOP) database (lo Conte et al., 2000). These databases contain extensive information relating to protein domains and structures. Ideally, clusters detected by the Tribe-MCL algorithm should have similar domain architectures, including sequence patterns and protein folds, based on InterPro and SCOP, respectively.

InterPro Validation

The InterPro database (Apweiler et al., 2001) is a collection of protein domains and functional signatures from multiple databases such as PRINTS (Attwood et al., 1999), Pfam (Bateman et al., 2000) and ProSite (Falquet et al., 2002). This well-curated database contains a vast amount of information relating to protein domains and sequence motifs and is described in Chapter 1. It is possible to obtain InterPro information for many entries in the SwissProt database (Bairoch and Apweiler, 2000). In order to validate our clustering algorithm, we took SwissProt (release 39) and clustered it into 8,332 families using the Tribe-MCL algorithm. This analysis took approximately five minutes to complete on a Sun Ultra 10 workstation. Protein family and domain information for each SwissProt protein was extracted (if available) from the InterPro database.

Of the 8,332 families, 1,821 contain four or more members with corresponding InterPro annotations. Families that do not contain four or more annotated members are discarded. For each of these 1,821 families, we determine the domain structure of annotated members of that family, according to InterPro domain classifications, and retain the most frequently occurring domain combination. This analysis is performed in order to determine which

families exhibit robust domain combinations in contrast to less well-defined protein families which may display disparate (or even conflicting) domain architectures. Interestingly, 1,583 families (out of 1,821 or 87%) display full correspondence of domain structure across all annotated members. When individual proteins are considered, we count the proportion of proteins with identified InterPro domain combinations identical to the most frequently occurring domain combination of the cluster they belong to. The number of proteins with this property is 14,188 out of a total of 14,409 proteins considered (98%); this value can be considered as an estimate of the classification precision according to InterPro. This result illustrates that although the algorithm has no fixed concept of protein domains, the resulting families have a very consistent domain structure, indicating accurate and meaningful clustering. Although the second set was also clustered using Tribe-MCL, no validation was possible due to the limited availability of InterPro annotations for members of these families.

SCOP Validation

The Structural Classification of Proteins (SCOP) database (lo Conte et al., 2000) is a collection of well-characterised proteins for which three-dimensional structures are available in the Protein Data Bank (Sussman et al., 1998). These proteins have been expertly classified into families based on their folding patterns and a variety of other information. Given that family information for these PDB proteins is well understood and accurately represented in SCOP, it was decided to cluster all proteins in the PDB (18,248 entries) into protein families using the Tribe-MCL algorithm at multiple inflation values (corresponding to different cluster granularities). This analysis detected 1,167 families (at inflation value 1.1). With increasing inflation values of 2, 3, 4 and 5, the number of families is 1,395, 1,606, 1,672 and 1,761 respectively. For each set of clusters (i.e. families), we determine the most frequently occurring SCOP classifications, as above. We also count the number of distinct clusters containing identical SCOP annotations in the same way. We then calculate the total number of proteins in clusters with SCOP classifications consistent with the cluster SCOP assignment. For higher inflation values (i.e. tighter clustering), this precision estimate is highest: 87% for inflation value 5 decreasing to 79% for the lowest inflation value of 1.1. These results further indicate that the clustering obtained by Tribe-MCL is accurately and

consistently assigning proteins into families, despite the fact that this classification relies on structural similarities, which are not always detectable at the sequence level using algorithms such as BLAST.

2.4.5 Large-Scale Family Detection

An Analysis of the effects of Promiscuous Domains

As mentioned above, Tribe-MCL does not require any explicit knowledge of protein domains to detect protein families. This feature can be used for the analysis of domains that are present in many families, such as promiscuous domains. We have decided to analyse the presence of these domains in the SwissProt database using Tribe-MCL, and to estimate how frequently they occur, based on the presence of InterPro (Apweiler et al., 2001) entries in the corresponding SwissProt (Bairoch and Apweiler, 2000) sequence. In other words, for each protein entry in SwissProt containing a detectable domain with InterPro, we count how many different protein families we have detected that contain this domain. A list of the promiscuous domains in SwissProt is described in Table 2.2. It is surprising that the largest set of proteins that are interconnected through promiscuous domains comprises of 237 protein families (21,727 sequences in total), corresponding to 22.3% of all SwissProt entries. These inter-related families are shown together with their functional assignments in Figure 2.12. Although the spectacular complexity of these interconnected families and the range of their functional properties has been suspected before (Doolittle, 1995), this is first time that we obtain a glimpse of this effect at this scale using clustering and visualisation. This effect arises from a number of reasons. Firstly, the family detection is accurate but the corresponding domain is falsely identified by InterPro (e.g. crystallin, shared by 141 families - Table 2.2). Secondly, the presence of these domains in unrelated families represents a meaningful biological phenomenon (e.g. RNA-binding region RNP-1, shared by 110 families - Table 2.2). Thirdly, the granularity of family definition is sometimes too high, resulting in distantly related families containing similar motifs (e.g. immunoglobulin and major histocompatibility complex domain, shared by 107 families - Table 2.2).

InterPro ID	No. of families	Domain Name
IPR001064	141	Crystallin
IPR000504	110	RNA-binding region RNP-1 (RNA recognition motif)
IPR003006	107	Immunoglobulin and major histocompatibility complex domain
IPR000531	97	TonB-dependent receptor protein
IPR003015	96	Myc-type, helix-loop-helix dimerization domain
IPR001680	76	G-protein beta WD-40 repeats
IPR000561	73	EGF-like domain
IPR000169	72	Eukaryotic thiol (cysteine) proteases active sites
IPR000255	67	Phosphopantetheine attachment site
IPR001899	65	Gram-positive cocci surface protein anchoring hexapeptide
IPR001450	60	4Fe-4S ferredoxin, iron-sulfur binding domain
IPR000130	54	Neutral zinc metallopeptidases, zinc-binding region
IPR000205	54	NAD binding site
IPR001005	54	Myb DNA binding domain
IPR001440	52	TPR repeat
IPR001356	49	Homeobox domain
IPR000822	45	Zinc finger, C2H2 type
IPR001841	43	RING finger
IPR000005	42	AraC type helix-turn-helix domain
IPR001777	42	Fibronectin type III domain
IPR001452	38	Src homology 3 (SH3) domain
IPR002290	37	Serine/Threonine protein kinase family active site
IPR000886	34	Endoplasmic reticulum targeting sequence
IPR001304	32	C-type lectin domain
IPR000194	31	ATP synthase alpha and beta subunit, N-terminal
IPR002203	29	Protein splicing (intein)
IPR000923	28	Type-1 copper (blue) domain
IPR001092	28	Helix-loop-helix dimerization domain
IPR001789	28	Response regulator receiver domain
IPR001611	27	Leucine-rich repeat
IPR001917	27	Aminotransferases class-II
IPR000063	24	Thioredoxin family
IPR002110	24	Ankyrin-repeat
IPR001220	23	Legume lectins beta
IPR003009	23	Proteins binding FMN and related compounds core region
IPR000524	22	Bacterial regulatory proteins, GntR family
IPR002114	22	Serine phosphorylation site in HPr protein
IPR000014	21	PAS domain
IPR001478	20	PDZ domain (also known as DHR or GLGF)
IPR000792	19	Bacterial regulatory protein, LuxR family
IPR001650	19	Helicase C-terminal domain
IPR002088	19	Protein prenyltransferases alpha subunit repeat
IPR000644	18	CBS domain
IPR002035	18	von Willebrand factor type A domain
IPR000047	17	Lambda and other repressor helix-turn-helix
IPR000086	17	NUDIX hydrolase domain
IPR001623	17	DnaJ N-terminal domain
IPR002223	17	Pancreatic trypsin inhibitor (Kunitz) family
IPR000437	16	Prokaryotic membrane lipoprotein lipid attachment site

Table 2.2: The top 50 promiscuous domains from InterPro occurring in distinct SwissProt protein families identified by Tribe-MCL. Column names: InterPro ID is the InterPro accession number, No of families is the number of families in which the corresponding domain is present and Domain Description corresponds to the InterPro description line.

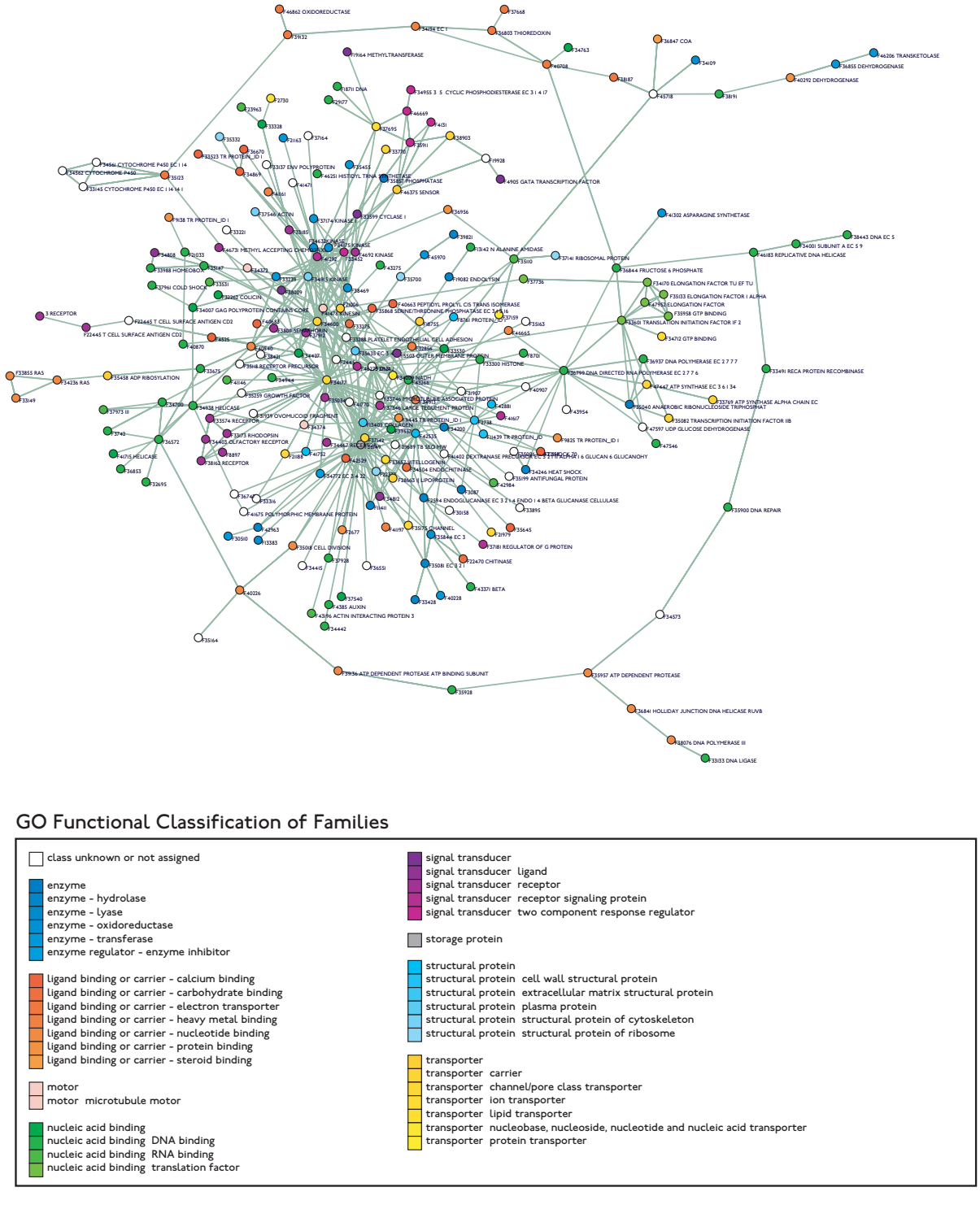


Figure 2.12: Graph representing the largest interconnected group of protein families from the SwissProt protein database (237 protein families, 21,727 sequences in total). Circles represent protein families, with associated family Ids and annotations (where known). Edges show BLAST similarities between families. Circles are coloured according to Gene Ontology (GO) (Ashburner et al., 2000) functional class assignments (where available). This figure was produced using BioLayout which is described in Section 5.2.

2.4.6 Algorithm Implementation and Availability

All source code for the Tribe-MCL package is freely available[5]. The package consists of a number of separate algorithms written in ANSI C. The *markov* program is used to generate binary markov matrix files from parsed similarity tables generated by BLAST or other more sensitive searching algorithms. The *MCL* algorithm is then applied to this matrix for iterative rounds of matrix multiplication, inflation and clustering resulting in a binary cluster matrix. This matrix is then post-processed by the *mcl-clusters* program which interprets this clustering information into protein family assignments.

2.4.7 Conclusions

The validation results for Tribe-MCL are very encouraging. Although the algorithm is an order of magnitude faster than many other methods, the results obtained appear to be robust and highly accurate. The actual implementation of the algorithm allows the efficient and rapid clustering of any arbitrary set of protein sequences, given a list of all pairwise similarities obtained by another method, such as BLAST. Because the method does not operate directly on sequences but on a graph that contains similarity information, it avoids the expensive step of sequence alignment. Instead, global patterns of sequence similarity are detected and used to partition the similarity graph into protein families.

The quality of the clustering is impressive, as validated using available protein domain and structure databases, InterPro and SCOP, respectively. Up to 98% agreement can be obtained in a comparison of the resulting classification using Tribe-MCL and the manually curated InterPro database. Given the speed and quality of the resulting clusters, Tribe-MCL has been used to cluster all human genes (from the Ensembl project) into annotated protein families (described in Section 3.2.1). This task would previously have been prohibitively expensive to achieve in such a short period of time. We hope that the method will become widely used by the community and find some other interesting applications in the field of bioinformatics and computational biology.

The two algorithms described in this chapter are complementary. The GeneRAGE algorithm provides detailed annotated clustering and domain

[5]http://www.ebi.ac.uk/research/cgg/services/tribe/

detection for prokaryotic sequences. For large-scale analyses involving eukaryotic protein sequences, the Tribe-MCL algorithm appears to be ideal. The method allows hundreds of thousands of sequences to be accurately classified in a matter of minutes. We have used both of these algorithms for novel and interesting research into protein sequence function, evolution and large-scale family analysis (described in the following chapter), and we hope that both algorithms will prove useful for biological sequence analysis.

Chapter 3

Analysis of Protein Families

Initial testing and validation of the GeneRAGE and Tribe-MCL methods proved very successful. However, a true estimation of the performance of these methods can only be assessed by applying these methods directly to fundamental biological research areas. Protein families are a useful tool for exploring many aspects of the biology and evolution of complete genomes, and for direct genome comparison. Many possible experiments using these methods are possible. In this chapter we shall detail a selection of research areas where we have successfully applied these two methods to biological problems.

The two methods presented in the previous chapter are different but complementary. The GeneRAGE method represents a tool for detailed detection and analysis of bacterial protein families and protein domains, and is firmly grounded in conventional sequence analysis methodology. Given a clustering result, it is possible to determine the exact process by which clusters were assigned. This is very useful for calibrating the method for detailed protein family analysis. The initial sections in this chapter detail the use of GeneRAGE for detailed protein family analysis, in order to answer very specific biological questions.

The Tribe-MCL method is based on a more abstract concept, and as such is more similar of a 'black box' approach. The method is extremely fast and highly scalable, and lends itself exceptionally well to the task of large-scale sequence clustering. In the final sections of this chapter we detail the use of Tribe-MCL for large-scale sequence clustering and protein family analysis with many hundreds of thousands of proteins.

3.1 Exploration of Protein Families using GeneRAGE

The first three sections of this chapter detail individual research projects where we have applied the GeneRAGE sequence clustering algorithm. The method is ideally suited to detailed protein family analysis of small genomes, and this has guided our choice of experiments. The first section describes sequence clustering and domain detection in archaeal genomes using GeneRAGE. The second section describes the generation of an automatic annotated classification of protein families involved in transcription, and subsequent evolutionary examination of these detected transcription associated protein families. Finally in the third section an experiment is detailed in which we attempt to automatically discover and classify proteins containing the low-complexity SR domain within metazoan genomic sequences. Because of its low-complexity it is generally difficult to detect and classify proteins with these domains, and as such is an excellent test of clustering performance. We hope that these experiments will detail the usefulness of protein family analysis using the GeneRAGE method.

3.1.1 Detection of Novel Archaeal Domains

The Archaea were not recognised as one of the primary domains of life until recently (Woese et al., 1978). Previously it was thought that the two major domains of life were the Eukaryotes and the Prokaryotes. When genome comparisons between bacteria were first performed in the 1970s it became obvious that a group of organisms, thought to be prokaryotes, were very different from other bacteria (Woese and Fox, 1977). These organisms clustered well away from both Eukaryotes and Prokaryotes in phylogenetic analyses. These organisms live in extreme environments and many produce gases such as methane. Due to the distance of these organisms from the two domains of life, it was proposed that they formed another separate domain, the *Archaebacteria*. The name Archaebacteria was later shortened to *Archaea* as these species have little in common with bacterial species. These organisms are difficult to culture, and have a similar appearance to bacteria when studied under a microscope, and hence were difficult to identify as a separate clade in the absence of molecular information. Because the importance of Archaea as a separate domain of life has only recently been discovered, much of their bi-

Species	Number of ORFs
Thermotoga maritima	1,849
Methanococcus jannaschii	1,773
Methanobacterium thermoautotrophicum	1,871
Archaeoglobus fulgidus	2,409
Pyrococcus horikoshii (shinkaj)	2,061
Aeropyrum pernix	2,694

Table 3.1: Complete archaeal genomes used for the experiment. The species name and number of open reading frames (ORFs) is given for each genome used.

ology is poorly characterised. For this reason archaeal genomics has become a popular field for the discovery of novel functional genomics targets.

Given that protein families and protein domains in Archaea are poorly defined, automatic detection and analysis of novel protein families, and protein domains for these species is highly desirable. Previously we have demonstrated the accuracy of the GeneRAGE method for both protein family detection, and domain decomposition for bacterial species. We decided to employ the GeneRAGE algorithm to the detection of novel families and domains within Archaeal species. When this analysis was performed the complete genomes for six archaeal species were available. These species are shown in Table 3.1. In order to detect domains which may be novel, it was decided that the Pfam protein families database would serve as an excellent reference database of known protein domains (Sonnhammer et al., 1998).

The analysis was performed as follows:

- FASTA sequences of predicted proteins from each organism were obtained via FTP from the laboratory which sequenced them.
- Each organism was compared to itself and the other organisms using the BLASTp algorithm (Altschul et al., 1997), with an expectation value threshold of $E \leq 1 \times 10^{-10}$. All sequences were first filtered for compositional bias using the CAST algorithm (Promponas et al., 2000), which is described in Section 5.1.
- Sequence similarities obtained from BLAST were parsed and supplied to the GeneRAGE algorithm, together with their associated FASTA

formatted sequences.

- Clusters (families) predicted by the GeneRAGE algorithm were inspected for the presence of novel Pfam domains, by comparing each cluster member with BLASTp to a FASTA database containing all proteins with domains present in Pfam ($E \leq 1 \times 10^{-10}$). Proteins predicted as multi-domain by the GeneRAGE algorithm were split into constitutive domains for this step of the analysis.

- Families (or domains) which had no detectable similarity to known Pfam domains were selected. Multiple alignments were then automatically constructed for each family or domain using the CLUSTALW multiple sequence alignment algorithm (Thompson et al., 1994).

- Alignments were then provided to the Pfam team at the Wellcome Trust Sanger Institute for automatic Pfam hidden markov model profile analysis.

- Domain alignments with no detectable similarity to known domains, using hidden markov model profile searches (Eddy, 1998), were hand curated and added to the Pfam database.

This analysis resulted in the addition of 294 protein domains to release 5 of the PFAM database (Bateman et al., 2000). The biochemistry of 50 of these archaeal domains (shown in Table 3.2) had already been described to some extent, yet they were not represented in Pfam. One example domain of this kind is Formylmethanofuran tetrahydromethanopterin formyltransferase (Ftr; accession PF01913), a transferase enzyme involved in the formation of methane from carbon dioxide. A multiple sequence alignment of Ftr distal lobe domains is shown in Figure 3.1.

The remaining 224 domains represented novel archaeal domains for which no function had yet been characterised. These domains may represent important targets for research into the underlying biology of the Archaea. A multiple sequence alignment for one of these domains of unknown function (accession PF01862) is shown in Figure 3.2. The addition of these novel domains to the PFAM database will hopefully prove useful for future analyses of this type, and help broaden our understanding of the biology of the Archaea.

PFAM ID	Domain Description
Adenine_deam	Adenine deaminase
Adenylate_cyc_2	Adenylate cyclase
Arch_flagellin	Archaebacterial flagellin
ArgJ	ArgJ family
ATP-synt_F	ATP synthase (F/14-kDa) subunit
BcrAD_BadFG	BadF/BadG/BcrA/BcrD ATPase family
CbiD	CbiD
CbiG	CbiG
CbiM	CbiM
CbiX	CbiX
CitG	CitG family
Desulfoferrodox	Desulfoferrodoxin
Diphthamide_syn	Putative diphthamide synthesis protein
DNA_primase_S	DNA primase small subunit
DS	Deoxyhypusine synthase
eIF5_eIF2B	Domain found in IF2B/IF5
eIF6	eIF-6 family
FTR	"Formylmethanofuran–tetrahydromethanopterin formyltransferase, distal lobe"
FTR_C	"FTR, proximal lobe"
HD	HD domain
Hydantoinase_A	Hydantoinase/oxoprolinase
HypD	Hydrogenase formation hypA family
IMP4	Domain of unknown function
KE2	KE2 family protein
MMR_HSR1	GTPase of unknown function
MoaC	MoaC family
MTD	"methylene-5,6,7,8-tetrahydromethanopterin dehydrogenase"
MtrH	Tetrahydromethanopterin S-methyltransferase MtrH subunit
NTP_transf_2	Nucleotidyltransferase domain
PcrB	PcrB family
Polysacc_synt	Polysaccharide biosynthesis protein
PyrI	"Aspartate carbamoyltransferase regulatory chain, allosteric domain"
PyrI_C	"Aspartate carbamoyltransferase regulatory chain, metal binding domain"
RibD_C	RibD C-terminal domain
Ribosomal_L14e	Ribosomal protein L14
Ribosomal_L37e	Ribosomal protein L37e
Ribosomal_LX	Ribosomal LX protein
SRP19	SRP19 protein
SurE	Survival protein SurE
TFIIE_alpha	TFIIE alpha subunit
Thi4	Thi4 family
ThiC	ThiC family
ThiJ	ThiJ/PfpI family
TraB	TraB family
Translin	Translin family
TRM	"N2,N2-dimethylguanosine tRNA methyltransferase"
tRNA_int_endo	"tRNA intron endonuclease, catalytic C-terminal domain"
tRNA_int_endo_N	"tRNA intron endonuclease, N-terminal domain"
tRNA-synt_1f	tRNA synthetases class I (K)
vATP-synt_AC39	ATP synthase (C/AC39) subunit
vATP-synt_E	ATP synthase (E/31 kDa) subunit

Table 3.2: Novel annotated PFAM domains discovered using GeneRAGE. The first column lists the Pfam domain identifier, and the second column the Pfam domain description.

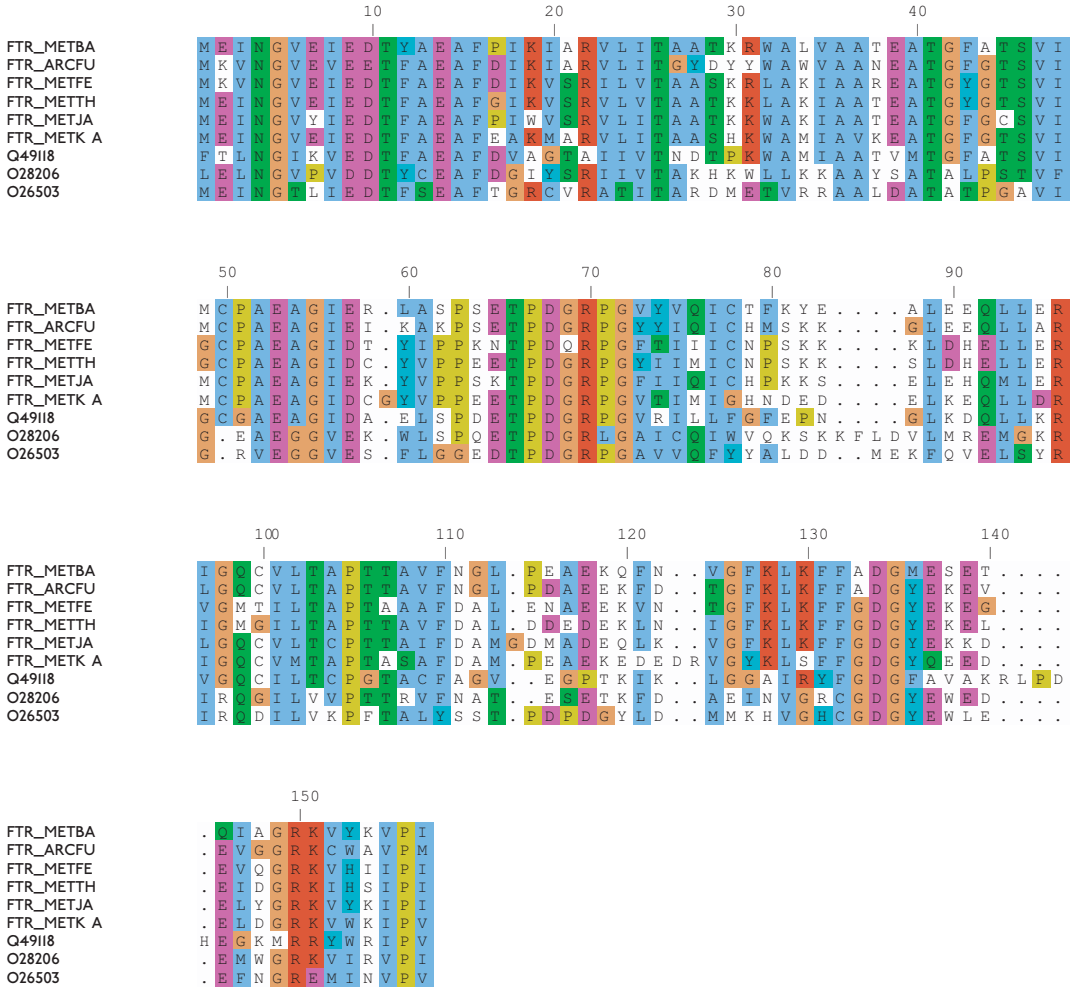


Figure 3.1: Pfam Archaeal FTR domain (accession PF01913) alignment (distal lobe). This alignment was produced using CLUSTALW (Thompson et al., 1994).

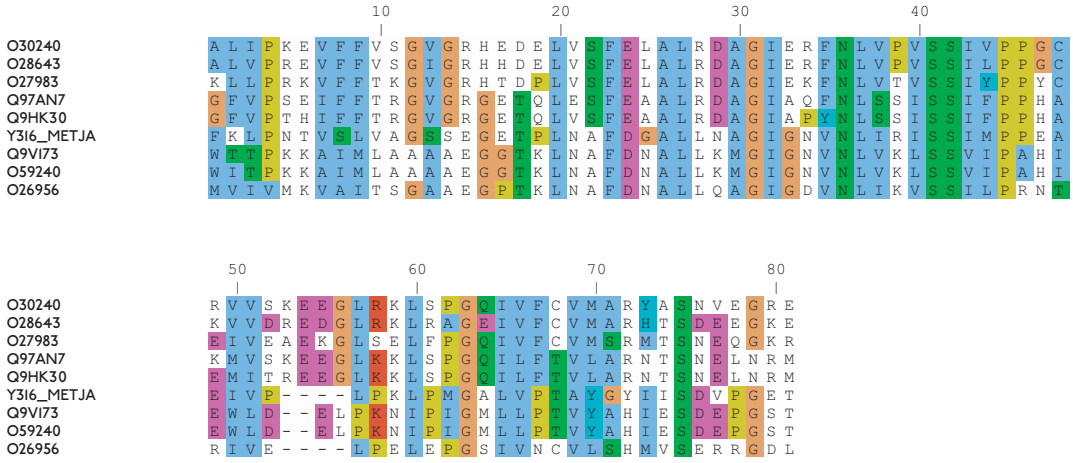


Figure 3.2: Pfam alignment of an archaeal domain (accession PF01862) of unknown function. This alignment was produced using CLUSTALW (Thompson et al., 1994).

3.1.2 Transcription Associated Protein Family Analysis

Protein families provide an effective way of grouping proteins according to shared function (Dayhoff, 1976). Some families will be specific to a single strain of an organism, while some widespread families may be universally conserved across all domains of life. Analysis of the species distribution of families can hence be a useful tool for the investigation of functional diversity across complete genomes. One area of intense interest is the functional diversity of transcriptional machinery, as mechanisms controlling gene regulation appear to be fundamentally different in eukaryotes and prokaryotes (Struhl, 1999).

In order to investigate this diversity computationally, it is desirable to build a comprehensive set of protein families involved in transcription. The species distribution of these families can then be used to explore the functional diversity of proteins involved in transcription. To this end, we sought to use the GeneRAGE sequence clustering algorithm to build a comprehensive database of protein families containing proteins thought involved in transcription. The initial database of transcription associated proteins (TAPs) was constructed in the following manner:

- Transcription associated proteins were extracted from the SwissProt and TrEMBL protein sequence databases via keyword searches (Kyrpides and Ouzounis, 1999) using the Sequence Retrieval System (SRS) (Etzold et al., 1996). Sequences were extracted from these two databases by searching for occurrences of the pattern '`transcription*`' in either the keyword (KW) or description (DE) fields. This search resulted in the detection of 5,894 sequences, which were stored in FASTA format.

- All protein similarity relationships within this set of proteins were then discovered using the BLASTp algorithm with an expectation value (E-value) threshold of $E \leq 1 \times 10^{-7}$. Prior to searching, all query sequences were filtered for regions of low-complexity using the CAST algorithm.

- This set of proteins, and all detected sequence similarity relationships were then supplied to the GeneRAGE algorithm for sequence clustering and domain detection.

This clustering analysis took approximately eight weeks running on a single Sun SPARC workstation and resulted in the detection of 985 distinct protein

families. Because the keyword searching method may result in the detection of many proteins which are not directly involved in transcription, only detected families containing three or more members were kept for subsequent analysis. A total of 744 families were discarded in this manner, leaving 241 protein families containing three or more members. These families contained 4,533 proteins (77% of the original 5,894). This final set of TAP families was annotated, hand-curated and stored in a database[1]. The final database of protein families predicted to be involved in transcription provided a basis for an exploration of the functional diversity of transcription associated proteins (Coulson et al., 2001).

Since each protein had been extracted from either SwissProt or TrEMBL, species and taxonomic information was available for each protein sequence. For each protein family, a full taxonomic classification for each protein within that family was obtained. Using this information it is relatively straightforward to determine for any given family, its distribution across domains and species. The results of this analysis are shown in Table 3.3 and in Figure 3.3.

Analysis of the species distributions of these hand-curated families showed that 90% of these protein families are uniquely present in one of the three primary domains (Figure 3.3a). There are only seven families that are universally present (Figure 3.3b). Two of these families contain the main RNA polymerase subunits, while two families contain the TenA and NifL regulators which are observed only in Fungi from the eukaryotic domain. Another universal family contains SIR2 which is the only universal regulator that is predominantly eukaryotic as it is only present in the bacterium *Streptomyces coelicolor*. The remaining two universal families in this group contain sequences present in eukaryotes but encoded by plastid genomes of bacterial origin (Gray, 1993). The families common between the Archaea and Bacteria are all transcriptional regulators (Kyrpides and Ouzounis, 1999). Apart from the MCM (Zhang et al., 1998) and SmuBP-2 (Mizuta et al., 1993) families, the TAPs shared between the Archaea and Eukarya are basal transcription factors and RNA polymerase subunits. Only the cold shock domain family is shared between eukaryotes and bacteria. In Bacteria, the three major categories (Firmicutes [gram-positive], Proteobacteria and Others [gram-negative]) exhibit fragmentation of their transcription components (Figure 3.3c) with 13% of the families unique to gram-positive and 32% unique to

[1]http://www.ebi.ac.uk/research/transcription/clusters

All Domains (Archaea, Bacteria and Eukarya)

A	B	E	AB	AE	BE	ABE
1.2% (3,20)	28.6% (69,812)	59.3% (143,2607)	2.1% (5,72)	4.2% (10,175)	1.7% (4,395)	2.9% (7,452)

Bacterial Domains (Gram+, Gram- and Others)

G+	G-	O	G+/G-	G+/O	G-/O	G+/G-/O
13.0% (9,41)	31.9% (22,120)	0.0% (0,0)	29.0% (20,316)	1.5% (1,4)	2.9% (2,6)	21.7% (15,325)

Eukaryote Crown Group (Fungi, Metazoa and Plants)

F	M	P	FM	FP	MP	FMP
16.1% (23,106)	44.8% (64,1058)	8.4% (12,83)	20.9% (30,631)	0.0% (0,0)	0.7% (1,316)	9.1% (13,413)

Table 3.3: Percentage of TAP families present in each set of domains. The total number of TAP families is shown in parenthesis, together with the total number of proteins.

gram-negative bacteria. Only 22% of the bacterial-specific TAP families are distributed across the three categories. The eukaryote crown group (Fungi, Metazoa and Plants) exhibits a far higher level of fragmentation with only 9% of families common to all three categories (Figure 3.3d). A total of 45% of the known eukaryotic TAP families are unique to Metazoa, 16% unique to Fungi, 8% unique to Plants and 21% shared between Metazoa and Fungi. Given the extensive sequence information obtained for Fungi or Metazoa (including complete genome sequences), it appears that these unique TAP families are confined to these domains. Less than 1% of the TAP families are common between Plants and either one of the other two categories.

The above analysis represents the most comprehensive computational overview of the transcriptional machinery to date and the results obtained are in accordance with previous computational and experimental work. The main result is that only a minority of transcriptional components are shared between major phylogenetic taxa, which supports the hypothesis that mechanisms of gene activation are intrinsically different in Bacteria and Eukaryotes (Struhl, 1999). Furthermore, because of our poor understanding of archaeal-specific transcriptional control, known archaeal TAP families are

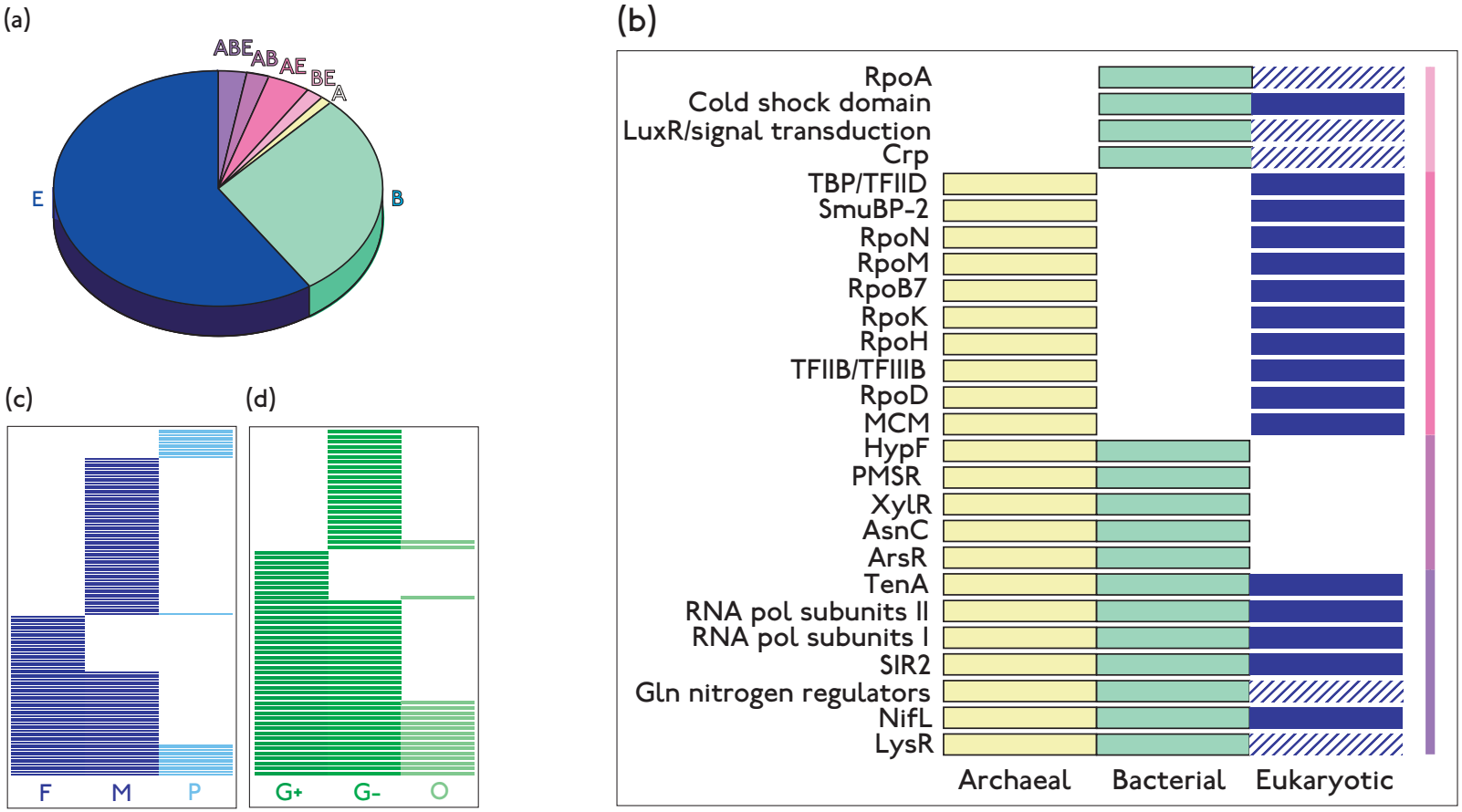


Figure 3.3: TAP family distribution across the three domains of life. (a) Results of the TAP clustering indicating the number of TAP families which are shared across domains (A: Archaea; B: Bacteria; E: Eukarya). (b)TAP families present in two or more primary domains. Hashed rectangles indicate eukaryotic TAPs encoded by chloroplast of cyanelle genomes. The scale bar on the right-hand side indicates the four non-unique groupings shown in (a). (c)TAP family distribution in the Gram-positive Firmicutes (G+), Gram-negative Proteobacteria (G-) and other Gram-negative bacteria that are not Proteobacteria(O). (d)TAP family distribution across the eukaryote crown group (F: Fungi; M: Metazoa; P: Plants).

mostly shared with the other two domains. This complements experimental results showing that Archaea have a basal transcriptional apparatus similar to eukaryotes but their control of gene expression is bacterial-like (Bell et al., 1999). Practical applications of the above observations include the identification of taxon-specific transcription factors as drug targets (Latchman, 1997).

Subsequent to this analysis, another analysis was performed by others (Riechmann et al., 2000), on the transcription factors of *Arabidopsis thaliana* and involved an similar analysis of the distribution of transcriptional machinery across three eukaryotic kingdoms (Fungi, Metazoa and Plants). This analysis showed that approximately 45% of *A. thaliana* transcription factor families are confined to plants. These data support our observations that the majority of TAP families appear to be taxon-specific.

3.1.3 Detection of SR-Domain Proteins

Domains rich in alternating arginine and serine residues (RS domains) are frequently found in metazoan proteins involved in pre-mRNA splicing (Caceres et al., 1997). RS domains present in splicing factors associate with each other and are important for setting up protein-protein interactions required for both constitutive and regulated splicing (Blencowe et al., 1999). The prevalence of the RS domain in splicing factors suggests that it might serve as a useful signature for the identification of new proteins that function in pre-mRNA processing, although it remains to be determined whether RS domains also participate in other cellular functions. The detection and analysis of RS domain proteins can hence be used to identify proteins whose functions are involved in pre-mRNA processing. In order to detect and catalogue RS domain proteins within complete genomes a strategy was developed using targeted database searching, compositional analysis of peptides and sequence clustering using GeneRAGE. The genomes chosen for this analysis were from the following eukaryotes: *Homo sapiens Drosophila melanogaster*, *Caenorhabditis elegans*, and *Saccharomyces cerevisiae*.

To establish search criteria for identifying protein sequences based on the presence of an RS domain, it was necessary to first establish an operational definition of an RS domain. To this end, we compared the sequences of several characterised SR family and SR-related splicing factors that contain short, functionally defined RS domains, in order to determine minimal features shared between these domains. All of the compared proteins contained several distributed SR or RS dipeptides and at least one stretch of two or more tandemly repeated SR/RS dipeptides. These features of known RS domains, were used as the primary criteria for the detection of novel RS domain proteins. Another feature of known RS domain proteins is that there are usually two or more of these dipeptide repeat runs and they usually occur within a region which is compositionally biased towards serine, arginine and other residues. The search strategy we have utilised for the identification and classification of RS domain proteins will be described below and is also shown in Figure 3.4.

The analysis starts by comparing an artificial peptide consisting of 30 repeated 'SR' dipeptides against a FASTA database containing the complete genomes of *Homo sapiens*, *Drosophila melanogaster*, *Caenorhabditis elegans* and *Saccharomyces cerevisiae*. This comparison was undertaken using the

BLASTp sequence similarity searching algorithm (Altschul et al., 1997). The top 500 sequence similarities to database sequences were stored from this comparison. This first pass is intended to select candidate RS domain proteins for more detailed analysis. These 500 candidate RS domain proteins are analysed using regular expression searches. Proteins which possess a perfect match to the pattern `/SRSR/` or `/RSRS/` are kept. This filtering procedure eliminates many sequences that contain repeated runs of arginine and serine, but not repeated arginine-serine dipeptides.

This search and filtering procedure identified 244 unique RS domain protein sequences. Among these, 47 proteins were from *H. sapiens*, 113 from *D. melanogaster*, 78 from *C. elegans*, and 6 from *S. cerevisiae*. The proteins were next clustered into related families of proteins automatically based on their similarity outside of regions containing compositionally biased regions (such as RS dipeptide repeats). This was achieved by filtering each of the 244 sequences using the CAST algorithm, and comparing all 244 sequences against each other with BLAST. The sequence similarity results and filtered sequences were then supplied to the GeneRAGE algorithm for sequence clustering.

The clustering process detected a total of 159 RS domain protein families. Of these families, 44 contained more than one protein and the remaining 115 contained only a single protein. The largest family (cluster 7) contained 36 distinct members, predominantly representing RS domain proteins with one or more RNA recognition motif (RRM). Of the 244 RS domain proteins in the data set, 87 (11 from *H. sapiens*, 48 from *D. melanogaster*, and 28 from *C. elegans*) were represented in more than one family due to overlapping domain relationships.

In order to enrich this set of proteins a full BLASTp search of the non-redundant protein sequence database (NRDB) with each of the 244 protein sequences identified the closest relatives, although not necessarily containing an SR/RS-repeat sequence. This identified numerous human homologues of *D. melanogaster* and *C. elegans* RS domain proteins that were absent from the *H. sapiens* Ensembl data set. The same RS domain criteria and two-step search procedure applied above was therefore used to determine which of the human protein homologues identified from searching the NRDB database contain an RS domain. This identified 127 human RS domain proteins, of which 82 match at least one of the entries from the initial *H. sapiens* data set and 45 match at least one entry from *D. melanogaster* and/or *C. ele-*

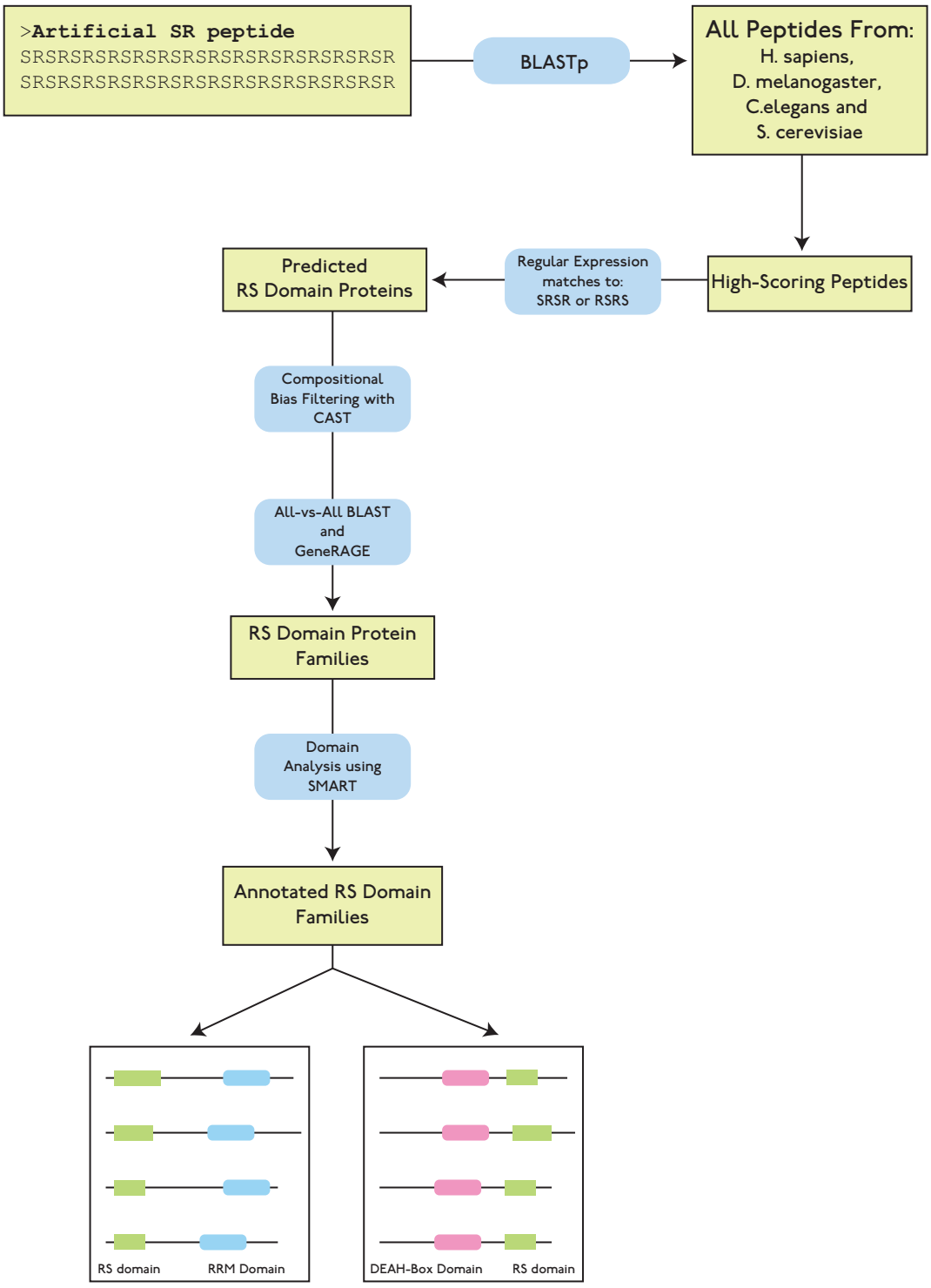


Figure 3.4: Strategy for the detection of RS domain proteins from the complete genomes of *H. sapiens*, *D. melanogaster*, *C. elegans*, and *S. cerevisiae*. The protocol is fully described in the accompanying text.

gans. All classified SR domain proteins were subsequently hand curated, and analysed for known domains using SMART (Schultz et al., 1998). This curation process was performed by our collaborators in the Blencowe laboratory in Toronto. The full set of curated SR domain proteins was stored in a database[2] for further analysis by our collaborators (Boucher et al., 2001).

This survey, besides identifying previously identified SR family and SR-related proteins involved in splicing, has uncovered many new RS domain proteins that are associated functionally with different cellular processes. The results demonstrate a useful strategy for the identification of RS domain proteins, and illustrate the versatility of the GeneRAGE algorithm. The analysis has provided a database of new factors that are candidates for forming interactions involved at different steps in the expression of RNA polymerase II transcripts, as well as in other cellular functions. We hope that the SR family database will provide some useful information for RNA processing research.

[2]http://www.maine.ebi.ac.uk:8000/sr/

3.2 Large-Scale Detection of Protein Families using Tribe-MCL

The Tribe-MCL method proved very successful in our initial validation procedures in the previous chapter. The method is exceptionally fast which makes it ideally suited to large-scale protein family analysis. In the following sections of this chapter we shall describe the application of this method to protein family detection and analysis within very large genomic databases. The first application of Tribe-MCL involved the detection and annotation of protein families within the draft human genome (Lander et al., 2001). Tribe-MCL was relatively untested when this research was carried out, yet it performed accurately, robustly and more importantly was able to process very large amounts of sequence data in a very short period of time.

Due to the success of this initial application, we have applied Tribe-MCL to another even larger undertaking. This project involves the generation of a large protein families resource which contains sequence and family information for all publicly available complete genomes and sequence databases such as SwissProt. The current version of the database contains over 64 complete genomes. This resource represents a huge collection of functional and evolutionary data for protein sequences and is, to our knowledge, the largest of its kind in existence. It is possible to conduct large-scale functional and evolutionary research using this database. In order to illustrate this, a variety of experiments have been carried out, using the database, and have produced some interesting results concerning genome evolution, functional genomics and phylogeny. These initial results will also be fully described within this section.

3.2.1 Family Annotation of the Draft Human Genome

The release of the publicly sequenced draft human genome necessitated that protein family analysis and functional annotation, of these predicted human protein sequences, be performed and made available to researchers (Hubbard et al., 2002). This type of information can be very useful for locating similar genes to a gene of interest or for assigning function to previously unknown genes. To achieve high-quality annotation automatically, functional descriptions for all human genome sequences might be obtained from curated SwissProt proteins, based on detected sequence similarities and the subsequent

family assignment. To this end, Tribe-MCL was applied to a full set of peptides from the Ensembl 0.80 release (29,691 proteins) of the draft human genome along with a vertebrate subset of proteins from the SwissProt and TrEMBL databases (73,347 entries). The protocol used is shown in Figure 3.5, and can be described as follows:

- All protein similarities within these 103,038 proteins were detected using BLAST (Altschul et al., 1997).
- This information (over 15 million protein similarities), was used by the Tribe-MCL algorithm to detect protein families within these three datasets. The Tribe-MCL inflation value parameter was set to 4.0 in this case to allow the detection of highly conserved (closely related) protein families, as these families are more accurate for annotation transfer (Enright et al., 2002). The calculation took approximately 15 minutes on a single CPU of a Compaq ES40 server.
- Finally, the RLCS algorithm (described in Section 5.4) was used to determine a consensus annotation for each detected protein family based on curated SwissProt annotations.

In all, 13,023 protein families were detected, of which 11,481 families (88% of the total) are human-specific. On average, each human protein family contains 2.5 members, while there are only 1,110 single-member families (3% of the total number of families). The family size distribution has an exponential shape, with hundreds of protein families with more than 20 members (Figure 3.6) and 347 families with more than 50 members, indicating a high degree of paralogy. Some well-known families that are detected are zinc-finger containing proteins, olfactory receptors, members of the RAS superfamily of GTPases, myosin, actin, keratin, immunoglobulin, certain ribosomal proteins and multiple kinase types (Table 3.4). The procedure has detected many well-known families and a number of large families in the human genome whose functions are either unknown or predicted. As an example, we show that the TFIIB family of proteins (Tan and Richmond, 1998) has been identified correctly, containing the human, rat and *Xenopus* homologues (Figure 3.7). Despite the fact that the quality of clusters is very high, some of the largest families (with more than 1,000 members) may contain a number of unrelated members. This usually arises from the presence of multiply repeated sequence patterns and not the presence of individual promiscuous

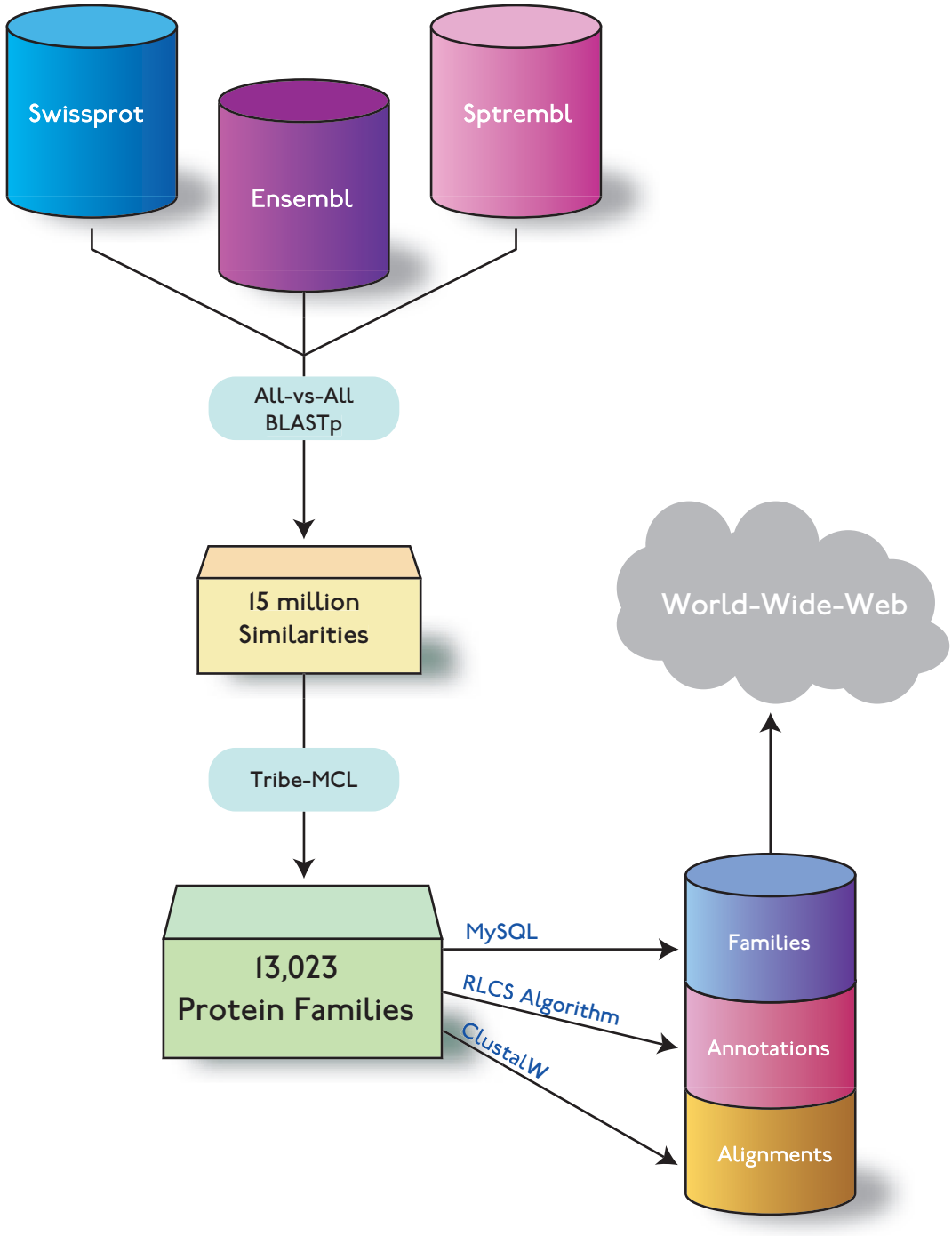


Figure 3.5: Pipeline for protein family detection and annotation for the Ensembl draft human genome. A full description of the protocol is detailed in the accompanying text.

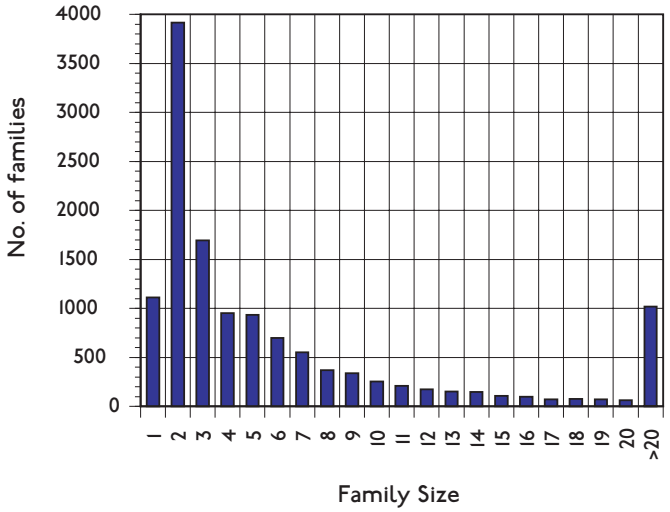


Figure 3.6: Distribution of protein family sizes within the human genome. The x-axis represents family size and the y-axis (blue bars) indicates the number of paralogous protein families.

domains. Future improvements to the Tribe-MCL method will attempt to solve this problem by using multiple levels of protein family classification, and extensive post-processing of the initial clusters.

The largest family identified (in release 080 of Ensembl) is a class of zinc-finger containing transcription factors, while the largest unannotated family contains 355 members (Table 3.4). All detected protein families were subsequently made available as part of the Ensembl 0.80 release. The clustering is fully accessible from the Ensembl website[3] and is continually being updated with new versions of the Ensembl database. It is worth mentioning that the annotations derived from the families detected by Tribe-MCL play a role in the Ensembl gene annotation system. The entire protocol for Ensembl family annotation has now been transferred to the Ensembl group so that family analysis can be performed automatically with each new Ensembl release.

[3]http://www.ensembl.org/

Ensembl Family ID	Consensus Annotation	No. of Members
ENSF00000002017	ZINC FINGER PROTEIN	1743
ENSF00000002558	CLASS II HISTOCOMPATIBILITY ANTIGEN, BETA CHAIN	1497
ENSF00000004397	CYTOCHROME B	1231
ENSF00000004396	CYTOCHROME B	1122
ENSF00000002016	OLFACTORY RECEPTOR	975
ENSF00000002557	CLASS I HISTOCOMPATIBILITY ANTIGEN, ALPHA CHAIN PRECURSOR	814
ENSF00000004395	CYTOCHROME B FRAGMENT	782
ENSF00000004394	CYTOCHROME B	731
ENSF00000004718	CYTOCHROME C OXIDASE POLYPEPTIDE I EC 1.9.3.1	648
ENSF00000002556	HLA CLASS I HISTOCOMPATIBILITY ANTIGEN, B ALPHA CHAIN PRECURSOR	526
ENSF00000006350	NADH UBIQUINONE OXIDOREDUCTASE CHAIN 4 EC 1.6.5.3	456
ENSF00000002015	MYOSIN HEAVY CHAIN,	455
ENSF00000004393	CYTOCHROME B	447
ENSF00000004392	CYTOCHROME B FRAGMENT	435
ENSF00000006349	NADH UBIQUINONE OXIDOREDUCTASE CHAIN 2 EC 1.6.5.3	419
ENSF00000002555	CLASS II HISTOCOMPATIBILITY ANTIGEN, ALPHA CHAIN	398
ENSF00000002013	PROTEIN TYROSINE PHOSPHATASE, NON RECEPTOR TYPE EC 3.1.3.48 PROTEIN TYROSINE PHOSPHATASE	381
ENSF00000002645	HEMOGLOBIN CHAIN	375
ENSF00000002014	RECEPTOR PRECURSOR EC 2.7.1.112	368
ENSF00000002012	UNKNOWN	355
ENSF00000002009	CADHERIN RELATED TUMOR SUPPRESSOR HOMOLOG PRECURSOR FAT PROTEIN HOMOLOG	349
ENSF00000004391	CYTOCHROME B	341
ENSF00000002554	HLA CLASS II HISTOCOMPATIBILITY ANTIGEN, BETA CHAIN PRECURSOR	341
ENSF00000002010	PROTEIN EC 2.7.1.-	341
ENSF00000002644	HEMOGLOBIN ALPHA CHAIN	338
ENSF00000006348	NADH UBIQUINONE OXIDOREDUCTASE CHAIN 2 EC 1.6.5.3	328
ENSF00000002011	EC 3.4.21.-	327
ENSF00000002553	CLASS I HISTOCOMPATIBILITY ANTIGEN, ALPHA CHAIN PRECURSOR	317
ENSF00000002007	IG CHAIN V REGION	314
ENSF00000002008	RECEPTOR	307

Table 3.4: The 28 largest protein families in the draft human genome recorded in Ensembl 0.80 together with their automatically derived consensus annotations and the total number of sequences (from Ensembl, SwissProt and TrEMBL) that they contain.

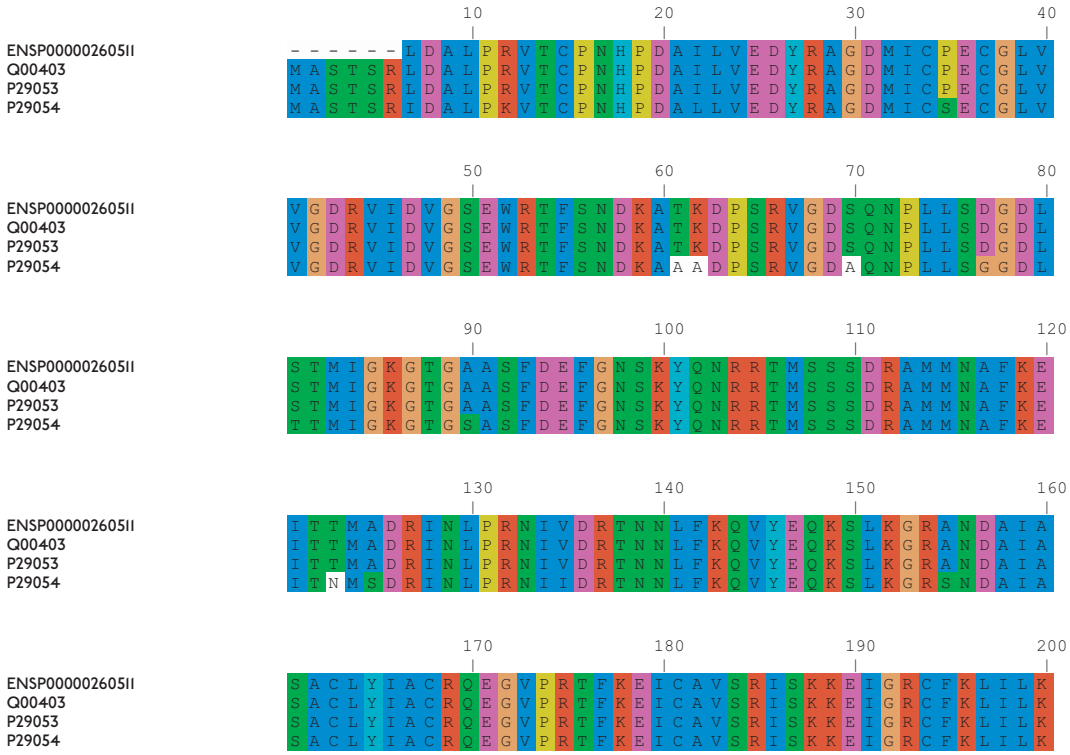


Figure 3.7: Protein sequence alignment of the eukaryotic TFIIB family of proteins detected using Tribe-MCL, including three members from SwissProt (accession numbers given) and the human TFIIB (Tan and Richmond, 1998). This alignment was constructed using CLUSTALW (Thompson et al., 1994).

3.3 Tribes: A Database of Protein Families

Initial validation of the Tribe-MCL algorithm and its application to the draft human genome, proved that the algorithm could be applied to protein sequence datasets containing hundreds of thousands of proteins in a robust manner (Enright et al., 2002). The rate limiting step for the detection of protein families with Tribe-MCL is now the initial similarity detection step using BLAST (Altschul et al., 1997). The availability of powerful computational resources, such as high-performance computing clusters, allows exhaustive all-against-all BLAST analyses such as this to be performed in a matter of hours, rather than weeks or days.

While the source code for the Tribe-MCL algorithm is freely available, many biologists interested in protein family analysis will not have access to such high-performance computing resources. For this reason we have undertaken to generate a comprehensive database of protein families from all publicly sequenced genomes, and databases such as SwissProt (Bairoch and Apweiler, 2000). Publicly available protein domain databases, and small scale protein family databases have proved very popular with the biological community (Sonnhammer et al., 1998; Tatusov et al., 1997). We believe that an exhaustive database of functional protein families would be of enormous benefit to the scientific community. The speed of the Tribe-MCL algorithm allows for this database to be automatically updated (and completely reclustered) every few weeks.

To facilitate the construction of a comprehensive protein family database, a number of issues need to be addressed. The first issue concerns the initial input data for the database. While the SwissProt database is updated and maintained in a regular manner, no such resource exists for publicly sequenced genome data. These data are published by the initial sequencing laboratory, yet there is no public repository for complete genome sequence data. Sequences from complete genomes will eventually be deposited in databases such as the EMBL and SwissProt databases, but this process takes a considerable amount of time. The next important issues are how the analysis will be performed, and how the database will be updated. The final issue involves the intelligent storage of these results in a searchable resource that may be accessed by users. In this section we will discuss each of these issues individually, and describe the construction of release 0.1 of the Tribes database.

3.3.1 A Complete Genomes Database

While resources have been developed to monitor sequencing projects (Bernal et al., 2001), no resource had been developed to store predicted proteins from these genomes in a unified manner. Construction of the Tribes database requires such a resource. For this reason, we have developed the *Complete Genomes Database* (CGD). This database has been developed to provide a single resource containing predicted peptides from all publicly available fully sequenced genomes. Complete genome sequences are being released at an unprecedented rate by the multiple sequencing centres. These centres provide different information for each genome in question and the naming schemes they use for the component proteins are varied, sometimes complex and in many cases uninformative. CGD stores each genome in a fixed ontology within the database and implements a novel and more intuitive naming scheme for proteins. The underlying database has been developed using the MySQL[4] relational database system. Due to the relational schema used, the database is consistent and redundancy is at a minimum. As new genomes are released, predicted peptides from these genomes are added automatically to the database through a simple Perl update script called *addgenome*. This Perl script is designed to process a FASTA file containing a list of peptides from a complete genome and represent it in the fixed ontology for the CGD database. This script requires minimal human intervention from a curator and adds data to two relational tables within the database (genomes and proteins). CGD is a concise, easily manageable central resource of fully sequenced genomes. The complete schema for the CGD database is shown in Appendix B, together with the *add_genome* Perl script required for adding newly sequenced genomes to the database.

Genomes Schema

The genome table contains information regarding each sequenced genome in the CGD database. Unique genome identifiers are generated automatically within the database when a new genome is added using the *add_genome* script. Information regarding a genome is added manually by a curator using the *add_genome* interface. This information includes the full name of the genome being added, its taxonomic classification and information regarding the location and date of its sequencing (Appendix B). Failure to provide any

[4]http://www.mysql.org/

of this information in the controlled format required, prevents the addition of the genome to the database.

Proteins Schema

Once a genome has been accepted into the database, the proteins from the FASTA file are loaded. CGD generates a unique and novel naming scheme for all proteins. This key feature of the database is that it is designed to unify protein identifiers across all complete genomes in a meaningful manner. The unique protein identifier string is automatically generated for each protein. This identifier has the form 4(letter)-3(letter/number)-6(numbers). The first part of the identifier (4 letters) is generated automatically from the full species name of the genome being entered. The first letter of the Genus and the first three letters of the Species are taken for each genome. This clearly identifies proteins from a given genome in an intelligent and obvious manner for users. For example, *Haemophilus influenzae* has a species code of `HINF`.

The second part of the identifier describes the strain. The default value is 'XXX' (meaning no stain given). This is designed for cases where the strain is not obvious or not given. This is used for cases such as the human genome, where a strain identifier is not meaningful. All bacterial and archaeal genomes have been sequenced from recorded strains of an organism. In these cases, the curator must specify a combination of 3 letters/numbers to represent each strain uniquely. For example, *Haemophilus influenzae* strain KW20 has a species code of `HINF-KW2`. The last part of the code (6 numbers) is used to identify each distinct protein from any given genome. The proteins are added to the database from a FASTA file sequentially by the *add_genome* script. The script is designed to strip non-standard amino-acid characters and symbols such as `'*'` and `'\'` from sequences. The protein identifier code is then generated in an automatic manner assigning '-000001' to the first protein processed and so on, until the final protein. The upper limit is 999999 proteins, significantly greater than the number of proteins we expect to be present in any given genome. Examples of these unique identifiers are : `DMEL-XXX-000302` is the 302nd protein in the FASTA file of the *Drosophila melanogaster* genome. The `XXX` symbolises that it is not a strain. `ECOL-MG1-000107`, `ECOL-EDL-000107` and `ECOL-RIM-000107` are the 107th proteins from three different strains of *Escherichia coli*. The strains

represented are EDL933, RIMD0509952 and MG1 respectively. The original identifiers for each protein are also stored within the database ensuring that the new protein identifiers can be mapped to the original identifiers in an automatic manner. This novel identification scheme ensures that every protein within the CGD database has a unique identifier and in addition it also gives the user clear information about the genome and strain from which a protein originates. If the FASTA file contains annotations for its constituent proteins, these are also stored.

Availability of the Database

Since the development and initial testing of the database, it has been brought up to date with the addition of 75 complete genome sequences[5] (February 2002). The current list of indexed CGD genomes is shown in Appendix B. When a genome is sequenced and made available to the public, it is obtained via ftp from the sequencing centre and added to the database by a curator. Draft genomes such as the Ensembl human genome and the complete genome of *D. melanogaster* are updated in the database when updated gene predictions are released. This database has already proved invaluable for the Tribes protein family database, and it should also prove useful to the wider scientific community for similar research.

[5] http://maine.ebi.ac.uk:8000/services/cgd/

3.3.2 Construction of the Tribes Database

The Complete Genome Database (CGD) represents the initial data for the Tribes database of functional protein families. In order to produce and annotate protein families for all peptides in the CGD database, it is useful to include reference peptides whose functional annotations are well characterised and curated. For this reason the SwissProt database of protein sequences and annotations is also taken in addition to the complete set of peptides from CGD. Sequence clustering of the CGD and SwissProt databases combined facilitates the functional annotation of protein families (Enright et al., 2002). The annotation strategy is two-fold. First, any detected protein family containing a large number of SwissProt protein sequences can be annotated with high accuracy based on the detection of a common consensus annotation from SwissProt proteins within that family. If this fails, an attempt is made to annotate any given family based on the annotations of peptides from the CGD database that occur in that family. This section will describe the computational protocol used for Tribe-MCL sequence clustering of the CGD and SwissProt databases, and the automatic annotation and representation of protein families detected by the Tribe-MCL clustering algorithm. The protocol used to generate version 0.1 of the Tribes database may be described as follows, and is shown in Figure 3.8.

- All peptide sequences are obtained from the CGD database and the SwissProt database, and stored in FASTA format. The initial analysis performed for Tribes version 0.8 involved 312,682 peptide sequences in total. Of these sequences 213,158 originate from the CGD database and 99,524 are obtained from SwissProt release 39.

- Each sequence is filtered for low-complexity regions using the CAST algorithm. Sequences is then compared against every other sequence using the BLASTp algorithm with an expectation value (E-value) threshold of $E \leq 1 \times 10^{-5}$. This analysis was performed in parallel on a cluster of 300 Compaq Alpha ES10 (EV5/6) machines using *lsfblast* (described in Section 5.5). These machines were kindly made available for this analysis by the Ensembl group and the Sanger Centre systems group. The analysis generated over 22 million protein-protein similarities, and took a little over 7 hours to complete.

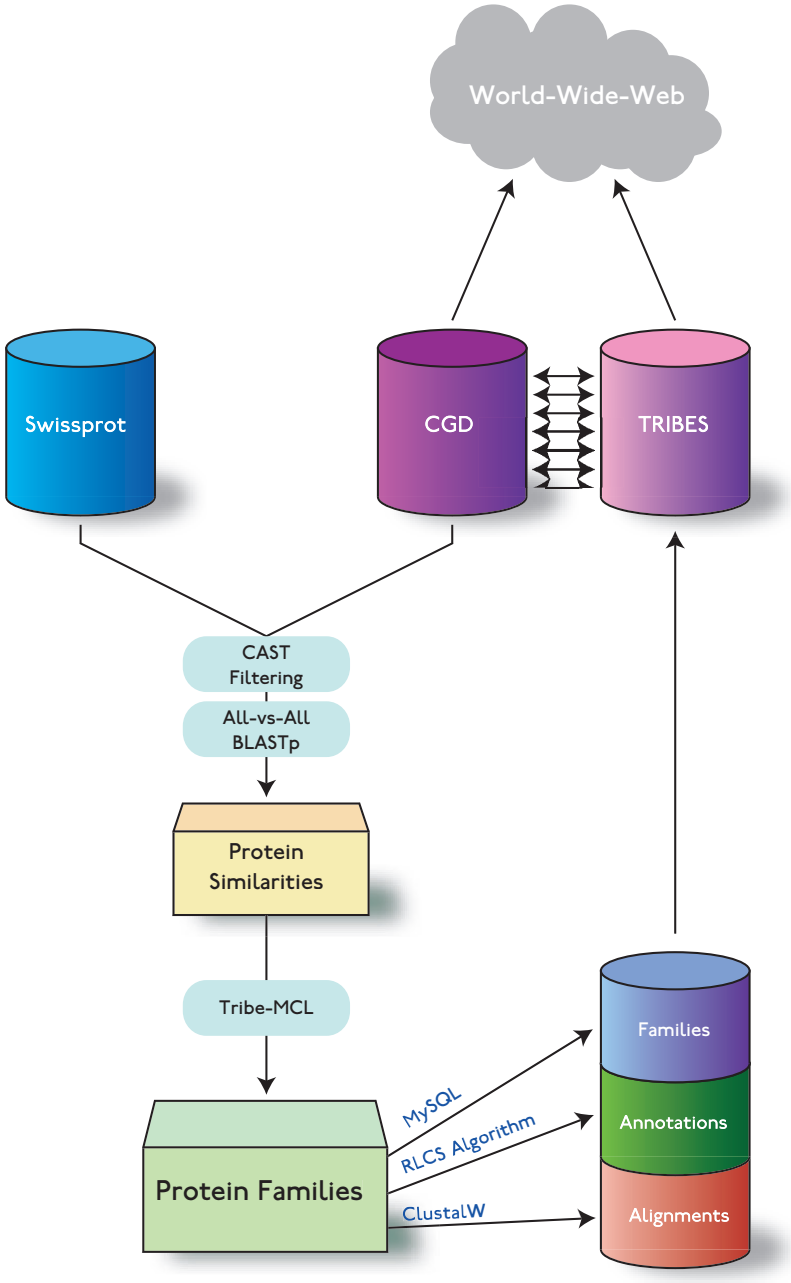


Figure 3.8: Pipeline for protein family detection and annotation for the Tribes database. The protocol is fully described in the accompanying text.

- All protein-protein similarities are loaded into a MySQL relational database of similarities which is linked to the original CGD database.

- A Markov matrix is constructed automatically from the BLAST similarity table by weighting each similarity according to $-log_{10}$(E-value), and then transforming each weight into a column-wise probability value.

- This Markov matrix is then supplied to the MCL algorithm for protein sequence clustering according to stochastic flow patterns detected in the Markov matrix at varying levels of inflation (1.1, 2.0, 3.0 and 4.0). This analysis took approximately one hour on a single Sun Microsystems UltraSparc 10 workstation.

- Each cluster detected by the algorithm is interpreted as a protein family. For each family all annotations are obtained for all sequences within that family from the CGD database. A consensus annotation is generated from all CGD and SwissProt annotations, that is descriptive, and covers the majority of proteins within that family. This consensus annotation is generated automatically using the RLCS algorithm (described in Section 5.4).

- All families, and associated consensus annotations are loaded into a MySQL relational database which is linked to the CGD database using the *addfamilies* Perl script. The schema of this database together with the associated *addfamilies* script is shown in Appendix B.

- An alignment was automatically generated for each family using the CLUSTALW multiple sequence alignment algorithm (Thompson et al., 1994). Multiple sequence alignments in MSF format were subsequently stored in an alignments table in the relational database.

- The final Tribes database is fully linked to the CGD and SwissProt databases (Appendix B). This database may be queried seamlessly across the world wide web using multiple Perl CGI scripts which access the database and return HTML formatted web pages containing relevant information. Screenshots of the Tribes interface are shown at the end of this chapter in Figures 3.12 to 3.17.

Inflation Value	No. Families	No. Singletons	No. Families > 2 members
1.1	48,045	30,139	17,906
2.0	60,322	30,203	30,119
3.0	65,798	32,230	33,567
4.0	70,215	34,974	35,241

Table 3.5: The number of families, 'orphan' families and families with two members or more generated by the Tribe-MCL clustering algorithm with inflation values of 1.1, 2.0, 3.0 and 4.0.

3.3.3 Protein Family Analysis using the Tribes Database

The Tribes database of protein families, constructed as described above, represents a novel resource for functional protein family information. This database should be of enormous benefit for functional and evolutionary analysis of protein families in complete genomes. In this section we will discuss initial research carried out using release 0.1 of the Tribes database. One aspect of this research involves the analysis of how many protein families are present in total and for each genome. Other interesting research areas involve the discovery of protein families which are universal (i.e. contain members from all genomes analysed), and families which are specific to a particular species or strain. The results from these analyses will be described in the following sections.

General Statistics

The total number of families produced varies significantly according to parameters supplied to the Tribe-MCL algorithm. With increasing inflation operator values, we expect Tribe-MCL to produce smaller 'tighter' protein family classifications. At the smallest value of inflation ($I = 1.1$) a total of 46,045 protein families are detected, while at the highest inflation value ($I = 4.0$) 70,215 protein families are detected. This information is shown in Table 3.5. The most significant difference in the number of families produced is observed when the inflation parameter is changed from 1.1 to 2.0. The number of singletons (i.e. families containing a single peptide sequence) increases very slightly, but not significantly as the inflation value increases.

For more detailed statistical analysis, we have concentrated on the broadest most informative protein family classifications ($I = 1.1$ and $I = 2.0$). Some general statistics of Tribes families generated at these two levels of granularity are illustrated in Table 3.6. A significant difference is observed

	I = 1.1	I=2.0
Number of Families	48045	60322
% Families with "UNKNOWN" annotation	21.45%	43.8%
% SP specific families	9.7%	8.64%

Table 3.6: The total number of families is shown together with a breakdown of the total number of families with an 'unknown' consensus annotation and the number of SwissProt specific families generated using the Tribe-MCL clustering algorithm using inflation values of 1.1 and 2.0.

in the success rate for the generation of consensus family annotations for the resultant families. Some 78.55% of the families generated at level 1.1 were successfully annotated while 56.2% of those at level 2.0 achieved a similar result. This result is counter intuitive as one might expect that it is harder to deduce a consensus annotation for larger protein families, such as those detected with inflation value 1.1. It appears that this effect is observed because this level of granularity matches protein superfamily information, which is easier to automatically annotate using the RLCS algorithm. Table 3.7 (columns 4 & 7) reveals that families specific to the eukaryotic domain achieve the greatest success rate for generating consensus family annotations at 85% for those observed at level 1.1 and 70% for those observed at level 2.0 (well above the average across all the families generated). Families specific to the bacterial and archaeal domains exhibit poorer performance than average for the generation of consensus annotations. This effect appears to originate from biases within the SwissProt database, as eukaryotic proteins are present in larger numbers, and may have better (or more descriptive) functional annotations.

The number of protein families specific to SwissProt proteins (i.e. those containing no proteins from the CGD database) is very similar for both levels, with 9.7% of families being SwissProt specific at level 1.1 and 8.64% at level 2.0. The distribution of the number of component proteins for the families generated at both levels is illustrated in Figure 3.9. At both levels, a similar number of orphan families were produced (30,139 at level 1.1 and 30,203 at level 2.0). The spread of data points along the x-axis in Figure 3.9a compared to Figure 3.9b illustrates clearly that families produced at level 1.1 tend to be significantly larger than those produced at level 2.0. The largest family at level 1.1 consists of 2,775 members while at level 2.0 the largest family consists of 1,130 members.

	I = 1.1 Number of Families	I = 1.1 % families	I = 1.1 Number families with "UNKNOWN" annotation	I = 2.0 Number of Families	I = 2.0 % families	I = 2.0 Number families with "UNKNOWN" annotation
Total Families	48045			60322		
A	5462	11.37	1379	6342	10.52	4360
B	18847	39.23	4748	23217	38.48	13178
E	19449	40.49	2975	24985	41.41	7452
V	2054	4.29	47	2250	3.74	49
AB	505	1.06	357	1154	1.92	639
AE	111	0.23	47	161	0.27	48
AV	5	0.01	2	7	0.01	1
BE	615	1.28	300	1161	1.93	427
BV	182	0.38	83	229	0.4	61
EV	57	0.19	20	85	0.14	19
ABE	650	1.36	298	657	1.1	165
ABV	12	0.03	6	12	0.03	2
AEV	6	0.02	2	5	0.02	0
BEV	29	0.07	15	30	0.05	12
ABEV	55	0.12	29	21	0.03	7

Table 3.7: The distribution of families generated (using inflation values of 1.1 and 2.0) across the 15 combinations of domain categories. The number of families, percentage of total families and number of families with an 'unknown' annotation in each combination of domain category at both levels are displayed. (Code: A - Archaea, B - Bacteria, E - Eukaryota, V - Viruses)

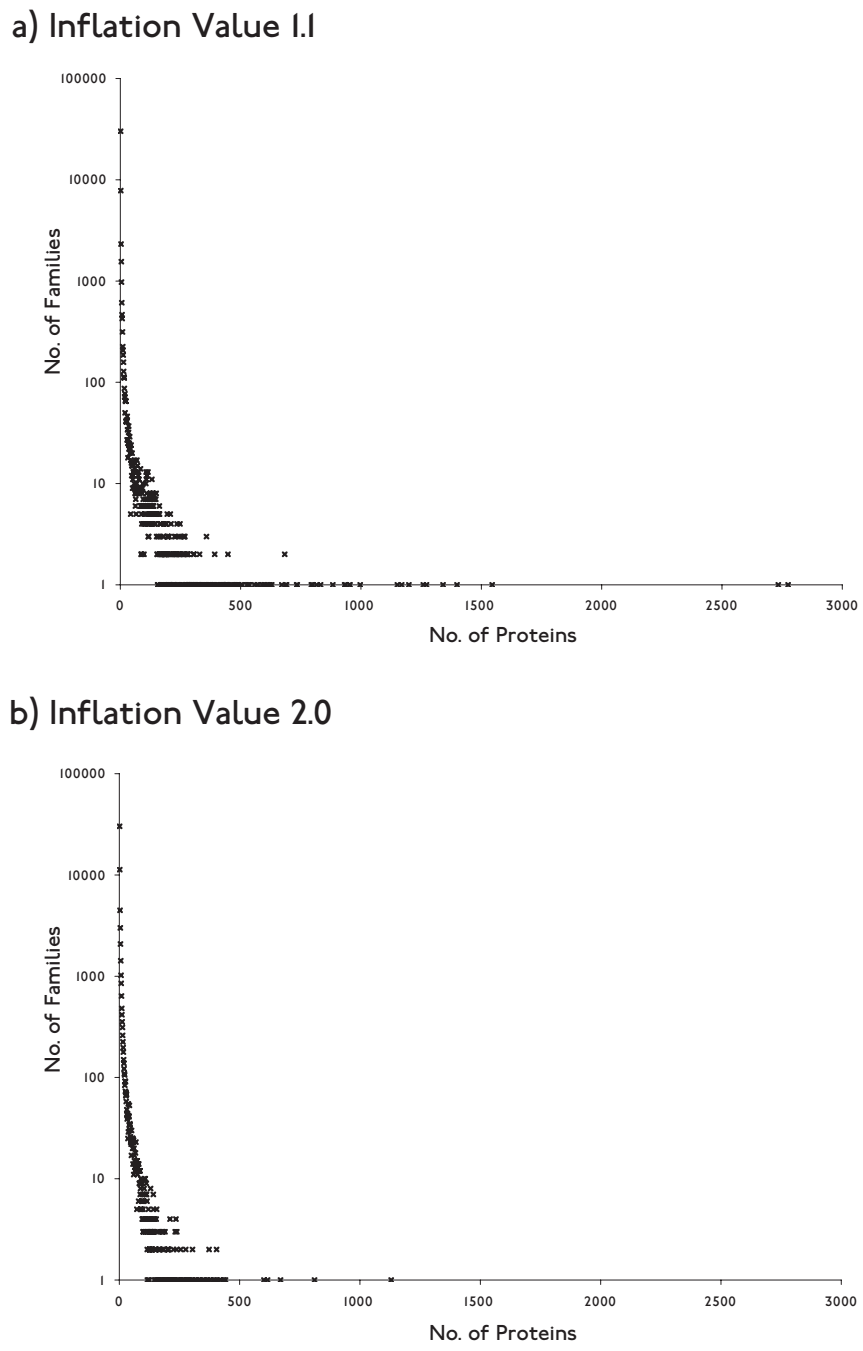


Figure 3.9: The distribution of the number of component member proteins for the families generated at inflation value levels of 1.1 and 2.0. The x-axis shows the total number of proteins in a family, and the number of families of each size are shown on the y-axis.

Phylogenetic Analysis

In order to study the phylogenetic distribution of protein families within the Tribes protein family database, families were classified into 15 mutually exclusive categories according to their observed species distributions. A protein family is hence assigned to a category according to which taxonomic domains are represented by the members of that family. The domains used for this analysis were Archaea (A), Bacteria (B), Eukaryotes (E) and Viruses (V). All viral proteins are SwissProt entries as there are no viral genomes present in CGD.

The breakdown of the families into these 15 categories is almost identical at the two different levels of granularity (Inflation values 1.1 and 2.0), and is shown in columns 3 & 6 of Table 3.7 and graphically in Figure 3.10a & 3.10b. The eukaryotic specific category (i.e. families whose members are only eukaryotic proteins) is the largest (40% of all families at inflation level 1.1 and 41% at level 2.0). The second largest category is the bacterial specific category of families (39% of all families at inflation value 1.1 and 38% at 2.0). The number of families specific to Archaea is significantly smaller (11% and 12% of all families).

The bacterial specific families are especially interesting. Over 90% of phylogenetic diversity occurs in the microbial world, as does a large proportion of metabolic, molecular and ecological diversity (Olsen et al., 1997). Only 0.02% of families at inflation level 2.0 are found to be universal among the 41 species and strains of bacteria within the CGD database. This extremely small percentage highlights the large degree of biological diversity present in the bacterial domain. This category of bacterial specific families could be further broken down in search of families that are universal among pathogenic bacteria. These families may provide targets for broad spectrum antibiotics.

All domain specific families were further analysed to determine the number of families within each domain that are universal across that domain (Table 3.8). For example, an archaeal specific family is considered universal among Archaea if it contains a member from each of the 10 archaeal genomes. The largest number of domain universal families was observed for Eukarya, with Archaea second and Bacteria displaying an extremely small number of universal bacterial families (Table 3.8). These results may be due to sampling biases, as only five complete eukaryotic genomes were present (and mostly represent Metazoa) in the release of CGD used for the analysis.

	I = 1.1	I = 2.0
Total number of archaeal specific families	5462	6341
Number of universal families across the archaeal domain	12	34
Percentage of universal families across the archaeal domain	0.22%	0.50%
Total number of bacterial specific families	18847	23213
Number of universal families across the bacterial domain	6	5
Percentage of universal families across the bacterial domain	0.03%	0.02%
Total number of eukaryotic specific families	19449	24977
Number of universal families across the eukaryotic domain	445	605
Percentage of universal families across the eukaryotic domain	2.28%	2.42%

Table 3.8: The number of families specific to each of the three domains Archaea, Bacteria and Eukaryota. The number (and percentage) of universal families across each of the domains is also illustrated. Results are generated using the Tribe-MCL clustering algorithm at inflation levels of 1.1 and 2.0.

The number of families composed of members representing two domains reveals some interesting results. Similar numbers of families fall into the AB and BE categories at both levels with the AE intersection being approximately five-fold smaller at inflation level 1.1 and seven-fold smaller at 2.0 (Table 3.7). Similar results have previously been obtained (Kyrpides et al., 1999), showing that the archaeal methoanogen *Methanococcus jannaschii*, contains four times as many bacterial-type than eukaryotic-type proteins.

The number of families containing at least one member from each of the three domains of life is extremely small (1.48% at level 1.1 and 1.13% at level 2.0). These results are illustrated in Figure 3.10a & 3.10b. These universal protein families are extremely interesting, and will be analysed further below.

Species specific families and proteins

Species specific proteins are those proteins that are unique to a particular organism. They display no significant similarity to proteins from any other genome and consequently cluster into individual species specific protein families. These proteins are interesting as they possibly possess functions highly specific to the life style, host or habitat of a particular species. For this reason the detection of protein families specific to a single strain or species of organism can provide functional insight into the underlying biology of that organism. The analysis presented here represents 53 complete genomes from the CGD database. This set is incomplete and biased, and hence it is not possible to fully determine whether a family specific to a single genome is

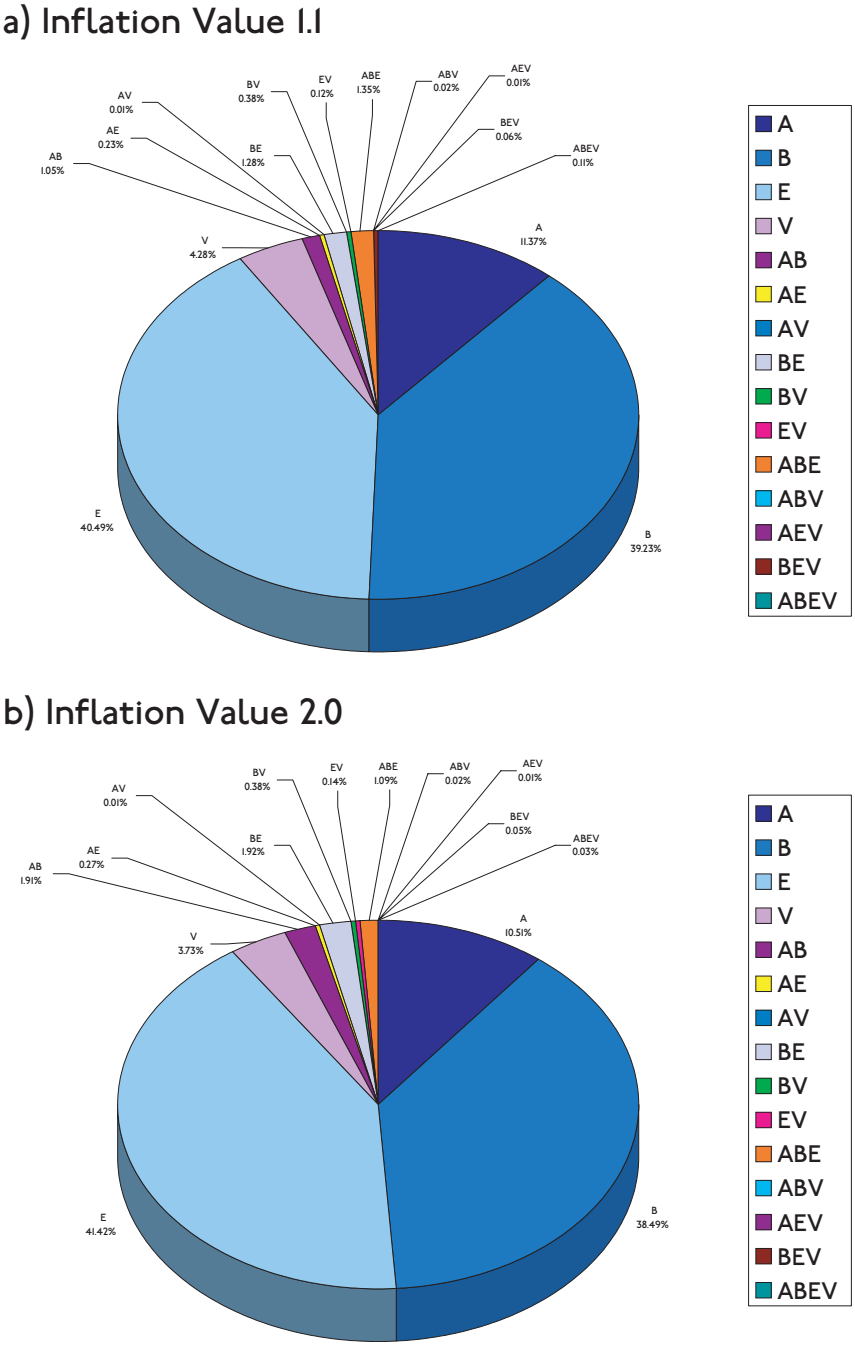


Figure 3.10: Pie-charts illustrating the phylogenetic distribution of protein families across taxonomic domains: a) shows the results obtained from the Tribes database at inflation value $I = 1.1$, b) shows the result obtained at inflation value $I = 2.0$. Domain abbreviations are explained in the text.

truly species specific in all cases. In this analysis we only consider species specific proteins, (i.e. proteins specific to a single species of micro-organism). Proteins and families specific to a single strain of multiple micro-organisms are also interesting (Janssen et al., 2001), but are not considered in this analysis. Genomes present in multiple strains in the CGD database are discarded for this reason (16 genomes). It also possible of course, that a rapidly evolving family may be detected as being species specific, because its sequence similarity to other families may no longer be detectable using BLAST.

The number of detected species specific proteins was calculated for every one of the 40 genomes considered. A Tribes family is detected as species specific if it contains a protein or proteins from only one of the 40 genomes considered, and does not contain SwissProt proteins from any other genome. Some detected species specific families are orphan sequences containing a single member. However, the majority of species specific proteins are found in families containing more than one member. These families usually contain multiple paralogues of a species specific gene, and are most probably the result of large-scale gene duplication within a particular genome.

The analysis was carried out on the families generated at both levels of family granularity (Inflation values of 1.1 and 2.0). The results obtained differ significantly with a greater percentage of proteins being found to be specific to each species at level 2.0. However, the relative pattern of species specific proteins observed between genomes at the two levels is very similar. These results show a large difference in the percentages of proteins being specific to the different genomes (Figure 3.11a & 3.11b). This result is supported by a previous analysis of 31 fully sequenced genomes (Iliopoulos et al., 2001b).

The analysis of species specific proteins is fundamental for our understanding of the underlying biological diversity of organisms. A greater number of species specific proteins are found for each genome at inflation level 2.0. This effect is observed because families produced at level inflation 2.0 have been further broken down into their component subfamilies.

At the larger inflation level ($I = 2.0$), the aerobic hyperthermophilic crenarchaeon *Aeropyrum pernix* has the greatest number of species specific proteins (56.09% of its genome). This supports previous work (Kawarabayasi et al., 1999) which reported that 57.1% of this genome did not show any significant similarities to other sequences. This number may decrease with the sequencing of more archaeal genomes. Similarly, 55.77% of *Borrelia burgdorferi* proteins are detected as species specific. It has been previously reported

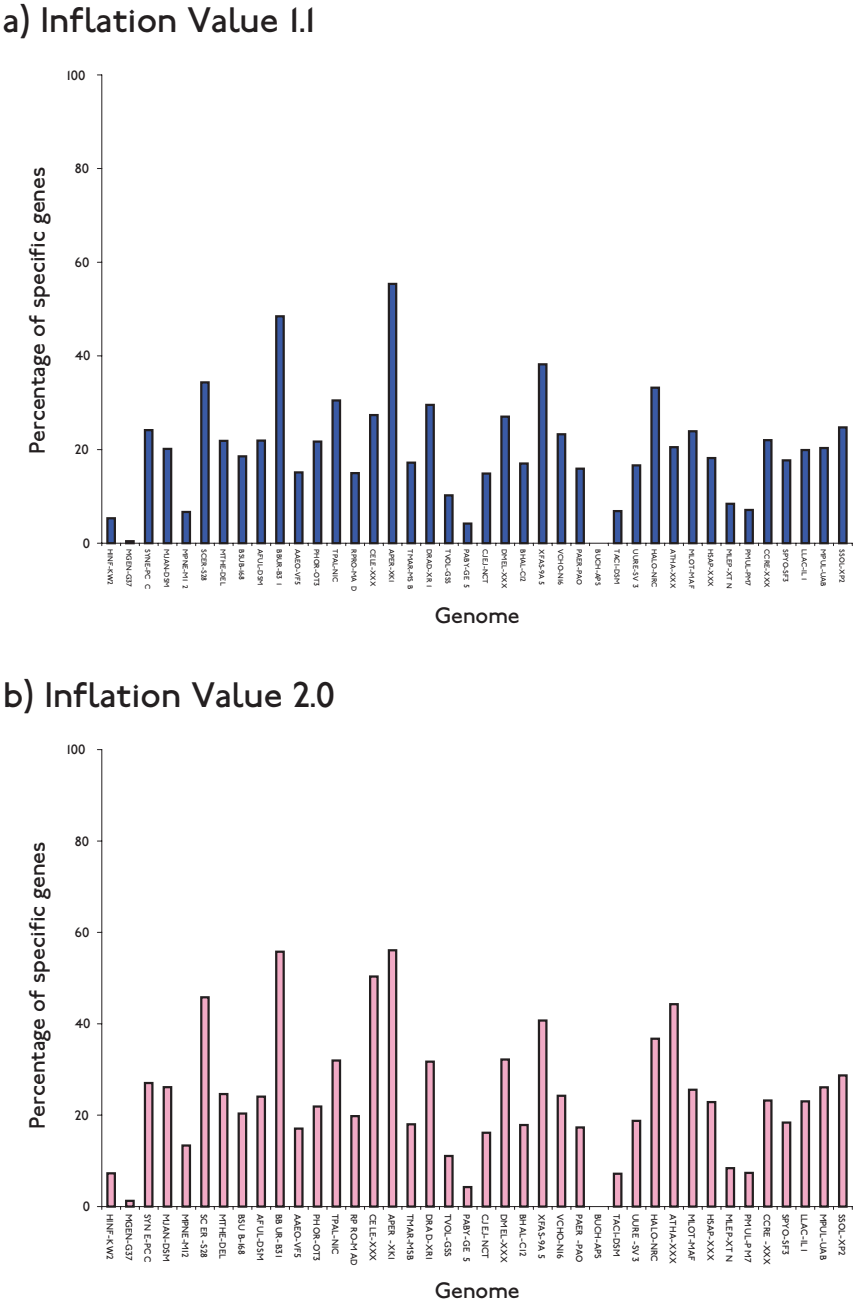


Figure 3.11: The number of species specific proteins for each of 40 genomes determined from the families generated from Tribe-MCL. a) shows the distribution at inflation level 1.1, b) shows the distribution for inflation level 2.0. Genome codes are shown in Appendix B.

that the plasmids in this genome which account for over 600 kbp of its 1230 kbp genome, carry very few genes with homology to genes outside the *Borrelia* Genus (Casjens, 2000).

At the other end of the scale, our results indicate that not one of the 575 predicted proteins of the *Buchnera* genome is unique (at both levels 1.1 and 2.0). This result supports previous studies (Shigenobu et al., 2000), which have suggested that that the *Buchnera* genome is a subset of the *E. coli*. genome.

The minimal genome of *Mycoplasma genitalium*, composed of just 479 proteins is shown to have six unique proteins at level 2.0 and two proteins specific to itself at inflation level 1.1. This result is surprising as previous work indicates that the *M. genitalium* genome is a subset of its close relative *Mycoplasma pneumoniae* (Himmelreich et al., 1997) and that every protein found in *M. genitalium* is found in *M. pneumoniae*, which is composed of 689 proteins.

Of the eukaryotes, four out of five species (*Caenorhabditis elegans*, *Arabidopsis thaliana*, *Drosophila melanogaster* and *Saccharomyces cerevisiae*) display a very high proportion of species specific proteins (Figure 3.11). This is to be expected considering the under-representation of sequenced eukaryotic genomes in this analysis. Interestingly, *Homo sapiens* is shown to have significantly less species specific proteins than the other eukaryotes at 22.88% of its genome at level 2.0. This is possibly due to the fact that it is the only vertebrate eukaryotic sequence analysed. One might expect it to contain a higher number of species specific proteins for this reason. The smaller value observed may be explained by the fact that a large number of proteins from the closely related mouse genome (*Mus musculus*) are present in SwissProt entries.

It is interesting that the majority of species with few unique proteins are bacterial thus indicating that many protein families in their phylogenetic neighbourhood have been detected. Those proteins unique to pathogenic bacteria are of great interest as they may provide suitable targets for antibiotics against a particular pathogen (Galperin and Koonin, 1999). Given the rapid progress in the field of genome sequencing, it is likely that a significant decrease in the number of species specific proteins for each organism will be observed in the coming years.

Universal Protein Families

The number of families that are ubiquitous, i.e. that are present in all 56 genomes, were determined at both levels of inflation. A total of 53 families contain members from all 56 genomes at inflation level 1.1 and 11 families at level 2.0 (shown in Table 3.9). 'Nearly' universal families are those containing a representative from 50 or more genomes. Certain genomes may be absent from the 'nearly' universal families for a number of reasons. The missing protein may have been predicted incorrectly, or not at all, in the missing genome, or perhaps they are absent due to unusual evolutionary phenomena such as non-orthologous displacement of a nearly ubiquitous gene.

The 53 universal families found at level 1.1 contain ribosomal proteins, amino-acyl transferases, translation initiation, elongation factors, metabolic enzymes, RNA polymerase, DNA polymerase III, topoisomerase, SRP, ATP synthase, GTP binding proteins and three families with no consensus annotations. The 11 universal families at level 2.0 are a subset of these families and hence have similar functions. A small number of the universal families represented in this table have no consensus annotation. These families were analysed further to determine if they were potentially new and unexplored ubiquitous families or simply families that had failed to achieve a significant annotation score. This was achieved by analysing the annotation of the SwissProt members of the family. If the SwissProt families contained many 'hypothetical' protein annotations, the family could be a novel universal family and should be subjected to further analysis. These possible novel universal families are shown in Table 3.10.

The number of universal families discovered in this research is extremely small. This highlights the incredible diversity present among living organisms. As expected, a greater number of universal families are found among the broader families generated at level 1.1. These universal families can be classified into three functional superfamilies, energy, information and communication. These three superclasses reflect the involvement of proteins in small-molecule, nucleic acid and protein-protein interactions respectively (Ouzounis et al., 1995). The majority of the universal families fall into the information category with a large number also populating the energy class. Very few universal families are found in the communication class. These results are similar to those found from comparison of the *Methanococcus jannaschii* genome to the NRDB database (Kyrpides et al., 1999). The hy-

pothesis was that metabolic enzymes and translation components are present but replication components, transcription factors and cell division factors are sparse among universal families. This finding is supported by the results presented here.

The novel uncharacterised protein families detected by this study (shown in Table 3.10) are interesting. These families will require a very detailed analysis in order to determine their widespread functional role. For a family to be present and conserved in all currently sequenced genomes, it must play a vital role in the fundamental biology of organisms, as the list of known families in Table 3.10 illustrates. These families clearly represent important targets for high-throughput proteomics, gene expression analysis and gene knockout analysis, so that their function might be determined.

Discussion

The Tribes database represents one of the largest collections of protein sequence families ever assembled. The analysis described in this section is intended to describe the types of analyses possible using these data. Of course, many such experiments are possible using these data. We intend to release version 1.0 of Tribes which will provide this exhaustive collection of sequence information to researchers. The full release will be automatically updated to take into account recently published genome sequences. We have provided a comprehensive web based interface to allow users to search protein family data and annotations and easily obtain important functional and evolutionary information. Screenshots of the current Tribes interface are shown in Figures 3.12 to 3.17. We hope that Tribes will be of benefit to the wider scientific community and aid the functional and evolutionary analysis of protein function in complete genomes.

Tribe Family ID	Consensus Annotation	No. of Members	No. of Genomes
TR-0000040447	THREONYL TRNA SYNTHETASE EC 6 1 1 3	103	56
TR-0000040459	TRANSLATION INITIATION FACTOR IF 2	120	56
TR-0000041608	METHIONYL TRNA SYNTHETASE (74%)	118	56
TR-0000043530	VALYL TRNA SYNTHETASE EC 6 1 1 9 VALINE TRNA LIGASE VALRS (44%)	108	56
TR-0000044833	RIBOSOMAL (90%)	153	56
TR-0000045033	SIGNAL RECOGNITION PARTICLE (62%)	190	56
TR-0000045043	GTP BINDING (70%)	72	56
TR-0000046718	ALANYL TRNA SYNTHETASE (83%)	117	56
TR-0000048632	PHENYLALANYL TRNA SYNTHETASE.... (43%)	105	56
TR-0000052934	DNA TOPOISOMERASE (73%)	153	56
TR-0000060221	ELONGATION FACTOR (64%)	232	56
TR-0000040579	ENOLASE EC 4 2 1 11 2 PHOSPHOGLYCERATE DEHYDRATASE....(49%)	159	55
TR-0000042065	ISOLEUCYL TRNA SYNTHETASE EC 6 1 1 5 ISOLEUCINE TRNA LIGASE ILERS (43%)	108	55
TR-0000042497	PHOSPHOGLYCERATE KINASE EC 2 7 2 3 (68%)	167	55
TR-0000043294	SERINE HYDROXYMETHYLTRANSFERASE EC 2 1 2 1 SERINE METHYLASE SHMT (48%)	133	55
TR-0000043513	RIBOSOMAL PROTEIN (93%)	111	55
TR-0000041228	CYSTEINYL TRNA SYNTHETASE (74%)	107	54
TR-0000044950	O SIALOGLYCOPROTEIN ENDOPEPTIDASE (65%)	85	54
TR-0000046709	SERYL TRNA SYNTHETASE EC 6 1 1 11 SERINE TRNA LIGASE SERRS (42%)	100	54
TR-0000058806	CTP (86%)	83	54
TR-0000042108	REDUCTASE (86%)	130	53
TR-0000043682	DIMETHYLADENOSINE TRANSFERASE (68%)	77	53
TR-0000040937	ENDONUCLEASE III (69%)	74	52
TR-0000041609	RIBONUCLEOSIDE DIPHOSPHATE REDUCTASE CHAIN EC 1 17 4 1....42%)	112	52
TR-0000043287	SYNTHASE (52%)	141	52
TR-0000044519	TRNA PSEUDOURIDINE SYNTHASE EC 4 2 1 70 (41%)	105	52
TR-0000046891	URIDYLATE KINASE (80%)	72	52
TR-0000058399	HISTIDYL TRNA SYNTHETASE (83%)	97	52
TR-0000043703	UNKNOWN / NO CONSENSUS ANNOTATION (100%)	88	51
TR-0000045050	UNDECAPRENYL DIPHOSPHATE SYNTHASE (43%)	94	51
TR-0000045834	THIOREDOXIN (85%)	242	51
TR-0000058559	ATP BINDING (82%)	441	51
TR-0000020863	TRNA PSEUDOURIDINE SYNTHASE A EC 4 2 1 70....(40%)	82	50
TR-0000041699	ADENYLATE KINASE EC 2 7 4 3 ATP AMP TRANSPHOSPHORYLASE (50%)	182	50
TR-0000042344	CHAPERONE PROTEIN DNAK HEAT SHOCK PROTEIN 70....(51%)	205	50
TR-0000045020	RIBOSE PHOSPHATE PYROPHOSPHOKINASE (62%)	95	50
TR-0000058354	TRANSKETOLASE (86%)	96	50
TR-0000058423	DNAJ (52%)	329	50

Table 3.9: The universal and 'nearly' universal families from the Tribes database detected with an inflation value of 2.0. The consensus annotation, number of members, and the number of genomes in each family are shown.

Universal Families at Inflation Level 1.1

Tribe Family ID	Consensus Annotation	No. of Members	No. of Genomes
TR-0000038098	UNKNOWN / NO CONSENSUS ANNOTATION	138	56

Universal Families at Inflation Level 2.0

Tribe Family ID	Consensus Annotation	No. of Members	No. of Genomes
TR-0000046415	UNKNOWN / NO CONSENSUS ANNOTATION	199	53
TR-0000033573	UNKNOWN / NO CONSENSUS ANNOTATION	107	52
TR-0000043703	UNKNOWN / NO CONSENSUS ANNOTATION (100%)	88	51

Table 3.10: Novel universal and 'nearly' universal families from the Tribes database which contain proteins whose functions have not yet been characterised.

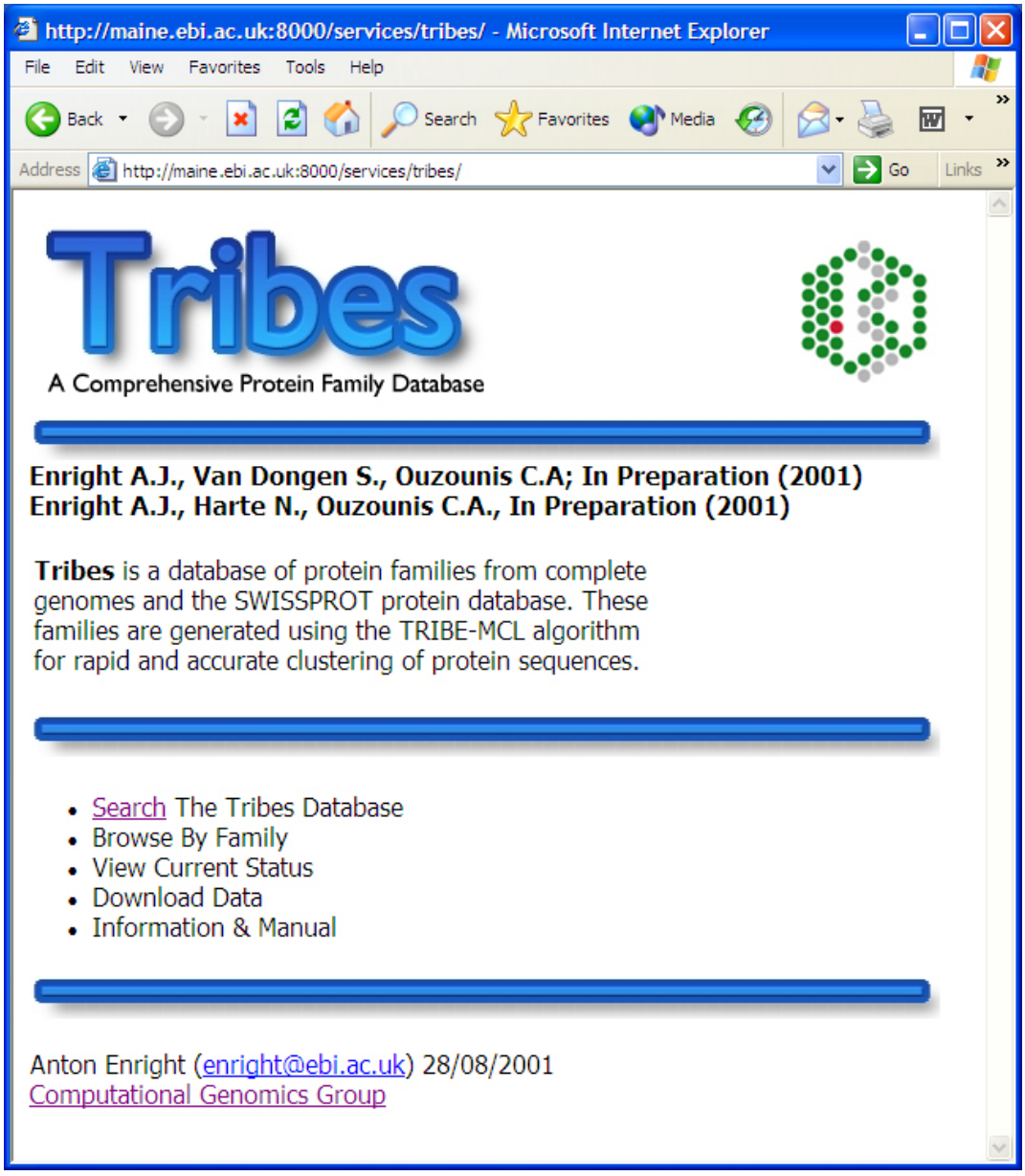


Figure 3.12: Tribes database entry page.

Figure 3.13: Tribes database search page.

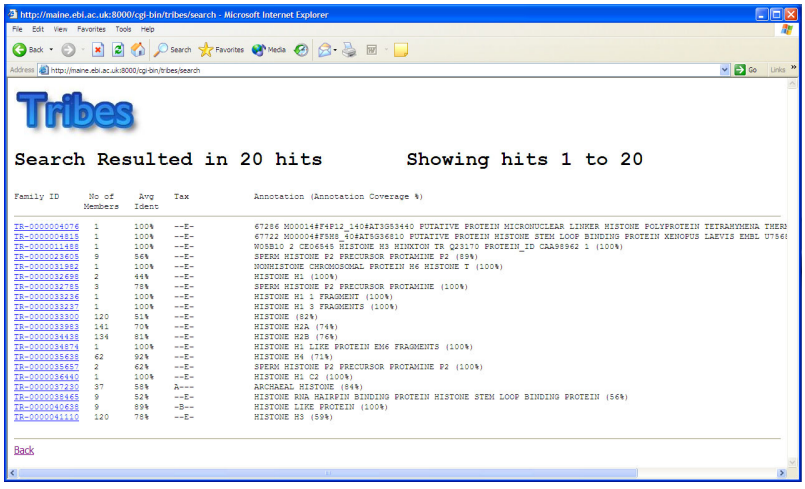


Figure 3.14: An example Tribes search for 'Histone' families.

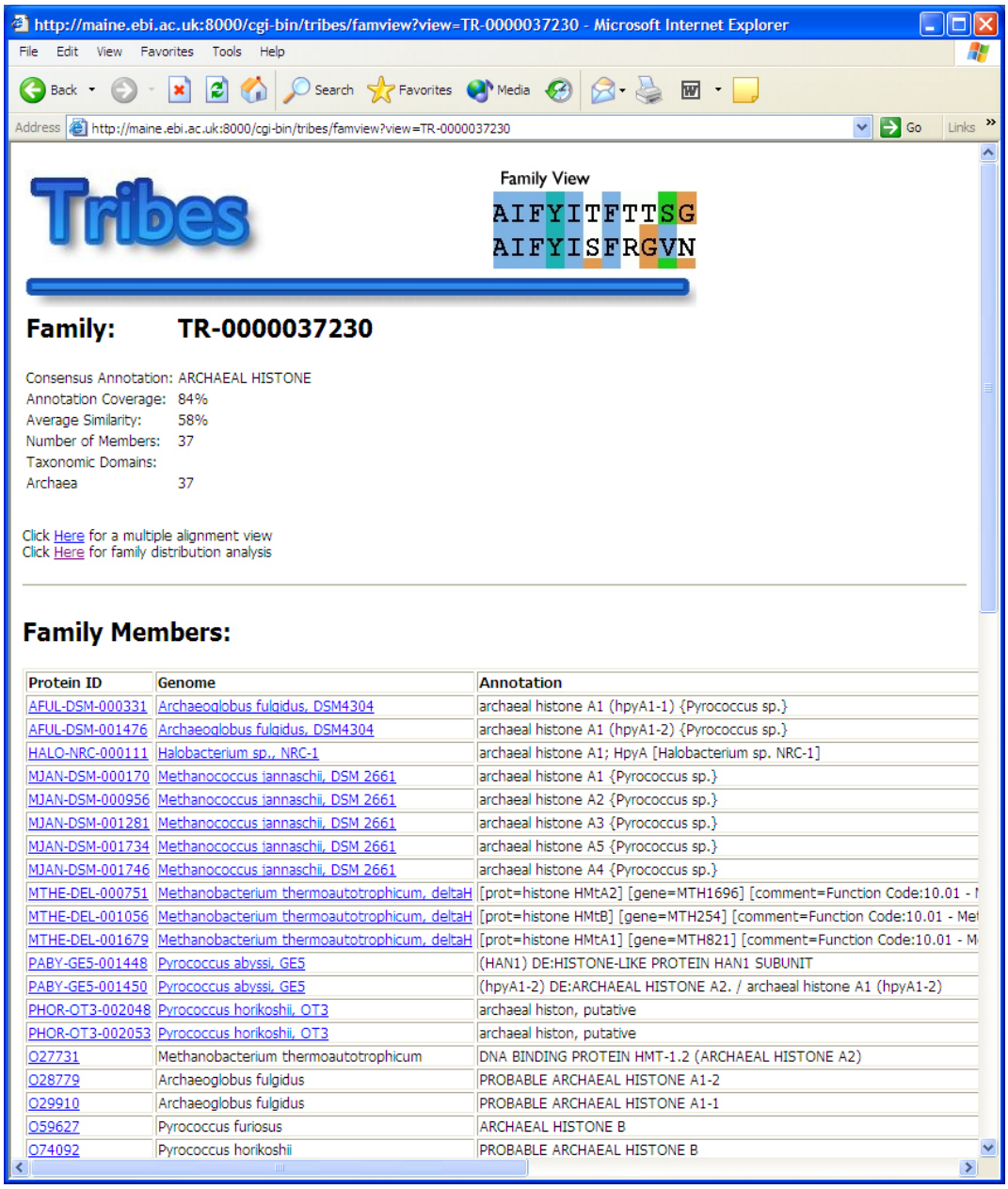


Figure 3.15: Tribes database family view page.

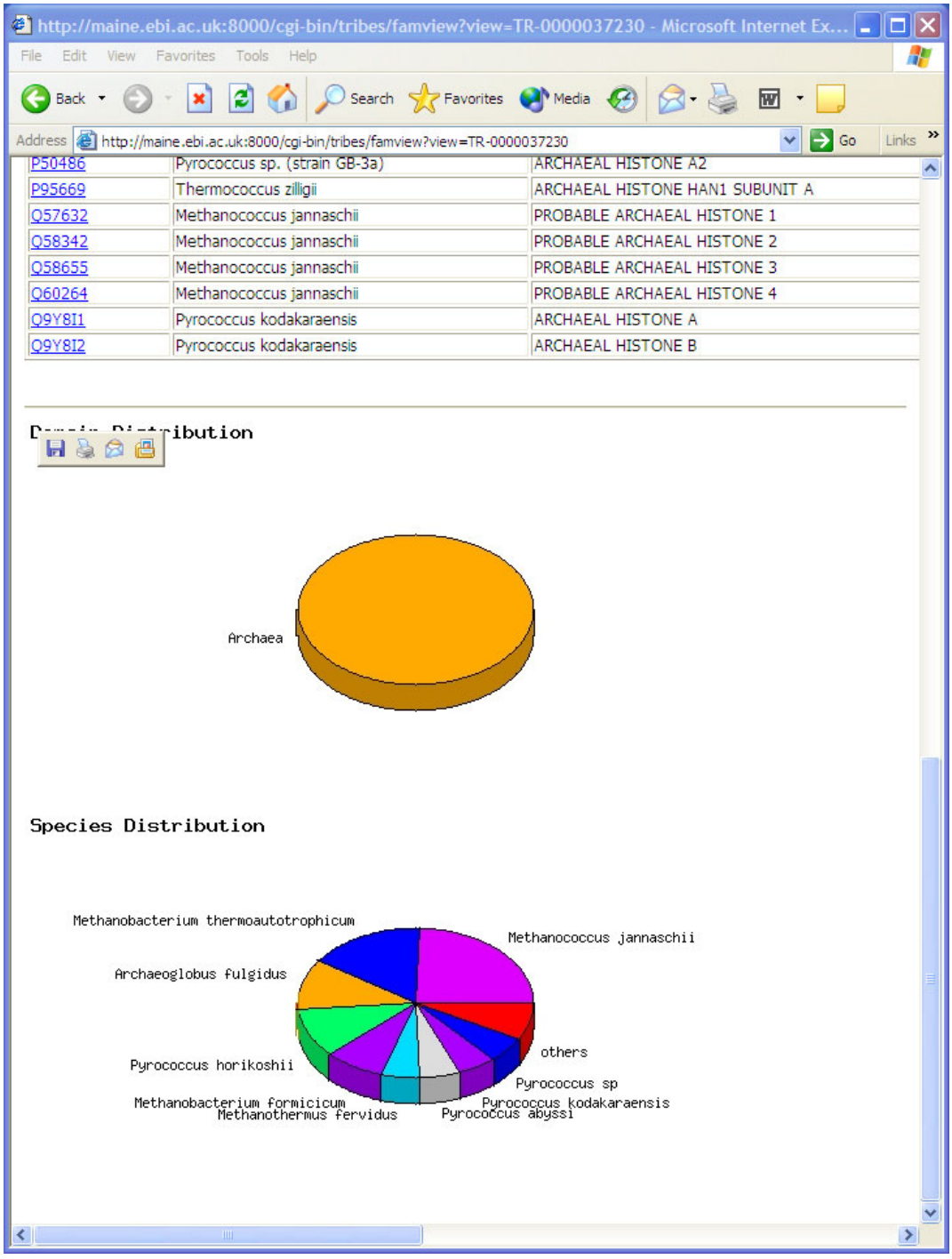


Figure 3.16: Tribes species specificity pie-charts.

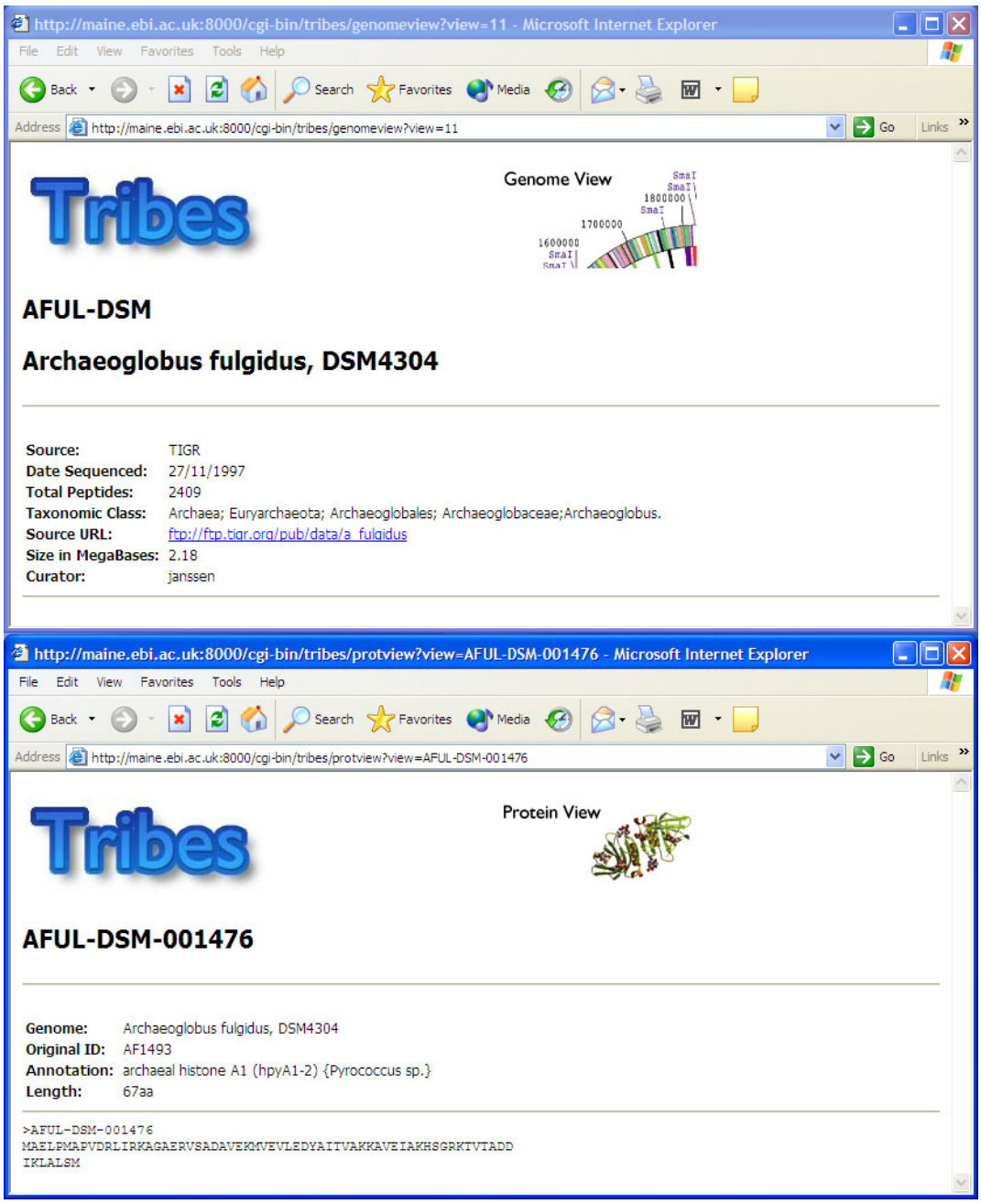


Figure 3.17: Tribes genome view and protein view pages.

Chapter 4

Genomic Analysis of Protein Interaction

Currently a large-scale effort to measure detect and analyse protein-protein interactions using experimental methods is underway (Mendelsohn and Brent, 1999; Blackstock and Weir, 1999). These methods come from a wide variety of areas, including biochemical techniques such as co-immunoprecipitation or crosslinking, molecular biology approaches such as two-hybrid or phage display, and genetic approaches such as unlinked noncomplementing mutant screening. Recently it has also been shown that it is possible to detect protein-protein interactions using protein chip experiments (MacBeath and Schreiber, 2000). Some of these methods are being applied in a high-throughput manner for the detection of whole-genome protein interaction networks. These approaches are currently labour intensive and ultimately inaccurate. Detection of protein-protein interactions using computational approaches in a rapid, accurate and automatic manner would complement these approaches and allow more detailed analysis of whole genome protein interaction networks. From a computational perspective the question is: How do we predict that two proteins interact solely based on their sequence or structure ? In this chapter an algorithm for the computational detection of protein-protein interactions via the detection of gene-fusion events within complete genome sequences will be described (Enright et al., 1999), together with further research into the applicability and accuracy of this method for protein-protein interaction detection (Enright and Ouzounis, 2001b).

4.1 Computational Detection of Protein-Protein Interactions

The problem of predicting protein-protein interactions computationally has been tackled by a number of groups in different fields of structural and functional genomics. From a structural perspective, protein-protein interface studies have perhaps been the most successful computational approaches. Structural analysis of the interfaces of known protein-protein interactions allows common structural interaction motifs to be identified. These motifs can be used to predict whether it is possible for two proteins of known structure to physically interact.

Functional genomics approaches are more recent and are based around a growing set of methods called *genomic context* approaches (Enright and Ouzounis, 2001c). These methods transcend conventional pure-homology based methods as they take into account the genomic context of proteins and genes within complete genomes. Examples of such approaches are gene co-localisation analysis, gene-fusion analysis and phylogenetic profiles of protein sequences (Enright and Ouzounis, 2001c). These methods can determine whether proteins, which are not necessarily homologous, are functionally associated or possibly, that they physically interact.

4.1.1 Structural Biology Approaches

Computational prediction of protein-protein interaction has been undertaken by structural biologists for quite some time. Most of these studies are based on analysis of protein-protein interfaces (Jones and Thornton, 1995) on the basis of structural and sequence motifs within the protein-protein interface of a known interaction that can allow the construction of general rules for protein interaction interfaces. Properties such as solvent accessible surface area (ΔASA measured in $\AA^2$), surface complementarity, residue interface propensity, hydrophobicity, hydrogen bonding and accessible patch analysis are widely used for this type of analysis (Lee and Richards, 1971; Lawrence and Colman, 1993; Janin et al., 1988). These rules can then be used to examine the structures of candidate pairs of proteins for possible protein interaction interfaces[1] (Jones and Thornton, 1996). These methods are successful for many well known proteins but are hampered by the rather sparse

[1] http://www.biochem.ucl.ac.uk/bsm/PP/server/

amount of three-dimensional structure data for many known proteins. These methods can also be very computationally intensive, and hence less useful for high-throughput detection of protein interactions in complete genomes.

4.1.2 Sequence and Genomic Context Approaches

Phylogenetic Profiles

The most general form of genomic context is the co-occurrence of orthologous genes in complete genomes. The context of genes in this case represents a phylogenetic or evolutionary context, i.e. genes that are functionally related tend to be inherited together through evolution (Pellegrini et al., 1999). In more general terms this means that if two proteins are functionally associated, then their corresponding pair of orthologues will tend to occur together in a given genome. The reverse of this argument is also true, if an orthologue of one gene is absent in a given genome, then it is likely that an orthologue of its associating partner will also be absent. In order to study the concept of an evolutionary genome context, a phylogenetic or 'phyletic' profile is constructed. Such a profile lists genes or families of genes, followed by a binary representation of the presence or absence of its orthologue in different genomes (Figure 4.1). It is then possible to predict a functional association between two genes that possess very similar profiles.

This method becomes much more powerful with an increasing number of genomes, as this allows larger more accurate profiles to be constructed. However, while elegant this method suffers from a number of drawbacks. Firstly, the presence of many paralogues of a given gene in a genome makes detection of a corresponding orthologue for a specific function very difficult. Secondly, the detection of members of orthologous families reliably is hampered by evolutionary processes such as non-orthologous gene displacement, gene loss, and horizontal gene transfer. These evolutionary mechanisms confound the construction of phyletic profiles by making the detection of a member of an orthologous gene family in a given genome very difficult. However, given the number of available complete genomes, the fidelity of these predictions should increase.

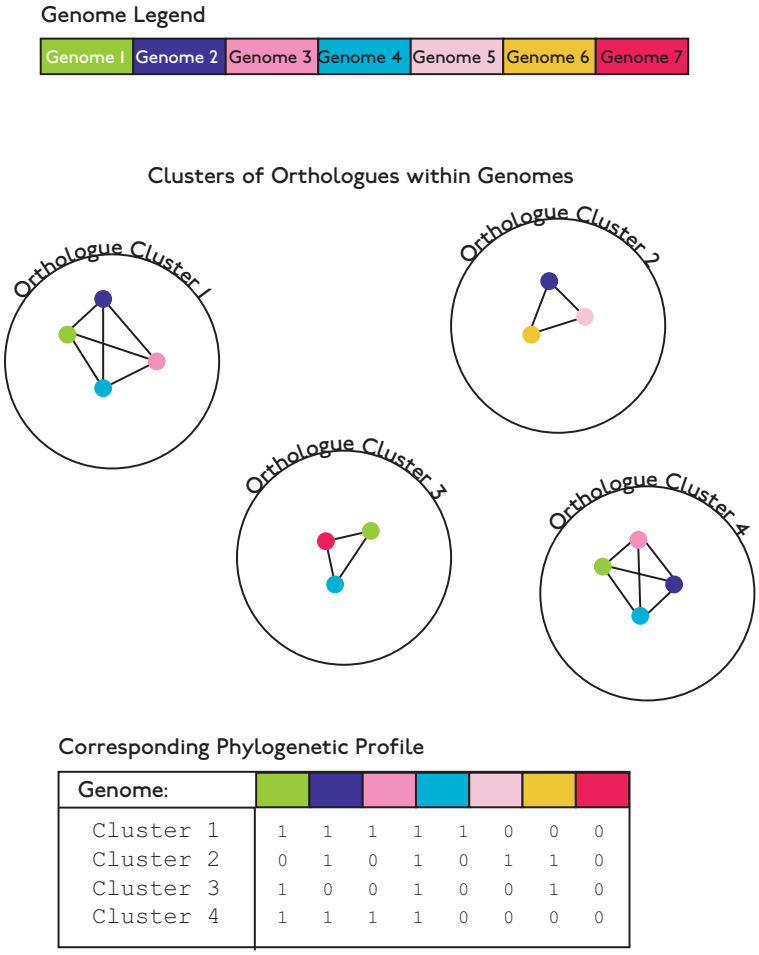


Figure 4.1: Cartoon depicting the construction of a phylogenetic profile for orthologue clusters from seven different genomes. Each orthologue is represented by a coloured circle according to the genome legend shown, and the resulting phylogenetic profile for each of the four orthologue clusters is shown. Orthologue clusters with very similar phylogenetic profiles, contain proteins which may be functionally associated.

Gene Co-Localisation

The first widely used method for exploring the genomic context of genes is the idea of co-localisation, or gene neighbourhood. This explores the notion that genes which interact or are at least functionally associated will be kept in physical proximity to each other on the genome (Dandekar et al., 1998). The most apparent case of this occurs with bacterial and archaeal operons, where genes that work together are generally transcribed on the same polycistronic mRNA (Blumenthal, 1998). Although this is generally not the case in eukaryotic systems, it is possible to infer functional association of proteins in a genome without operons. This involves detecting orthologues of these genes contained in an operon in other (most likely bacterial) genomes (Galperin and Koonin, 2000). This procedure has the inherent advantage that operon structures tend to vary considerably between species, providing yet more cases that can infer functional association of genes in genomes lacking operons. However, because it still remains difficult to predict operons, the amount of data available for this kind of functional inference is limited. The lack of operons in most higher eukaryotic species (Blumenthal, 1998) also means that information is limited to bacterial operons containing orthologous gene families that span both prokaryotic and eukaryotic domains. This may provide little or no information about orthologous families limited to the eukaryal domain. This method is however, not limited to studies of genes that occur in operons. It is also possible to predict functional association of a pair of genes if their orthologues tend to be in close physical proximity in many genomes (e.g. <300bp). A cartoon example of gene co-localisation is shown in Figure 4.2.

This method has been successfully used to detect missing members of metabolic pathways in a number of species (Overbeek et al., 1999). The power of this method becomes more apparent as more complete genomes become available. This method is also complementary to the analysis of gene fusion, which represents the ultimate form of gene proximity, complete fusion of two genes into one single unit. The use of gene-fusion events for the prediction of functional association and physical interaction between proteins is described in this chapter.

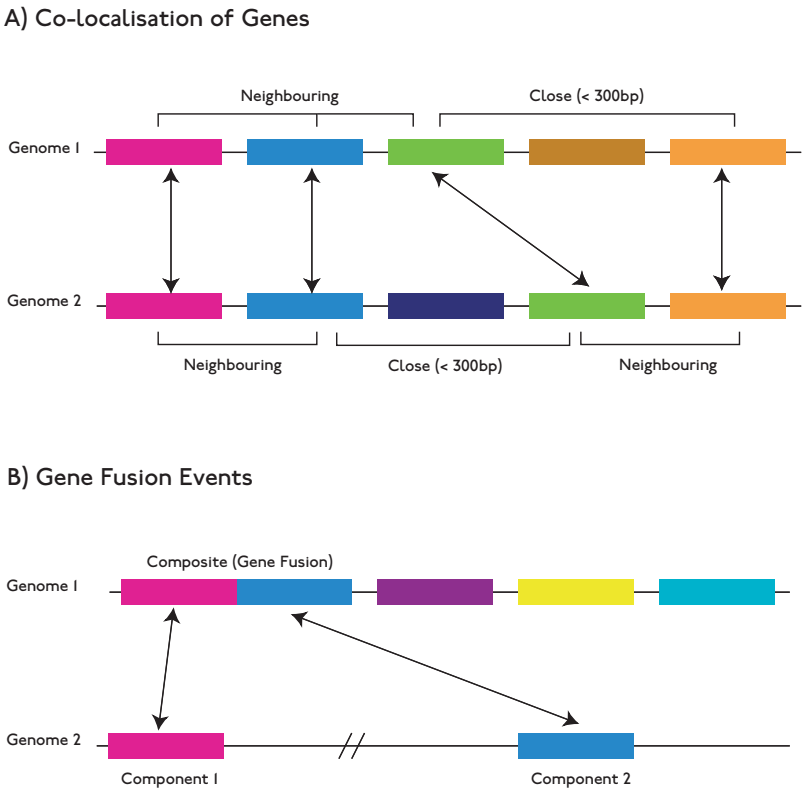


Figure 4.2: Cartoon depiction of (a) gene co-localisation and (b) gene-fusion events. Genes which are neighbouring or close (within 300bp) in many genomes may be functionally associated or physically interact. This is also true for genes which are deemed to be components of a fused gene in another organism.

4.2 Prediction of Protein Function and Protein-Protein interaction using Gene Fusion events

Previously, the only computational approach to the problem of predicting protein-protein interactions has been the computational structural analysis of subunit interfaces of proteins known to physically interact. These approaches suffer due to their computational complexity, and the relatively small number of available three-dimensional protein structures. Recently gene co-localisation studies have been successful in predicting physical interactions for many proteins (Dandekar et al., 1998; Overbeek et al., 1999) using genomic context. These analyses however result in a number of false predictions, because the constraint of proximity is not strong, and interactions between products of distantly located genes are not identifiable. In addition, this approach may not be applicable to eukaryotes, because the co-regulation of genes is not imposed at the genome structure level.

Clearly other modes of genome context can be used to infer functional association and physical interaction between protein sequences. One such method is the detection and subsequent analysis of gene-fusion events in complete genomes. Many genes in complete genomes become fused through the course of evolution due to selective pressure. Fusion of two genes may decrease the regulatory load in the cell, or allow metabolic channelling of substrates. The fusion of two genes in this manner provides evidence that the protein products of these genes are either closely functionally associated, or that they physically interact. The situation is comparable to experimental approaches that make use of artificial constructs of fused genes for biochemical analysis and protein-purification technology (Bulow, 1990; Wales and Wild, 1991). The gene-fusion process has been observed frequently in evolution, perhaps the most widely known example being the fusion of tryptophan synthetase α and β subunits from bacteria to fungi (Burns et al., 1990). Another well known gene-fusion is that of the TrpC and TrpF genes in the complete genomes of *E. coli* and *H. influenzae* (Ross et al., 1990), the three-dimensional structure of this bi-functional fusion protein (PDB accession 1PII) from *E. coli* is shown in Figure 4.3. A sequence alignment between the fused *E. coli* protein and its unfused counterparts in other organisms is shown in Figure 4.4.

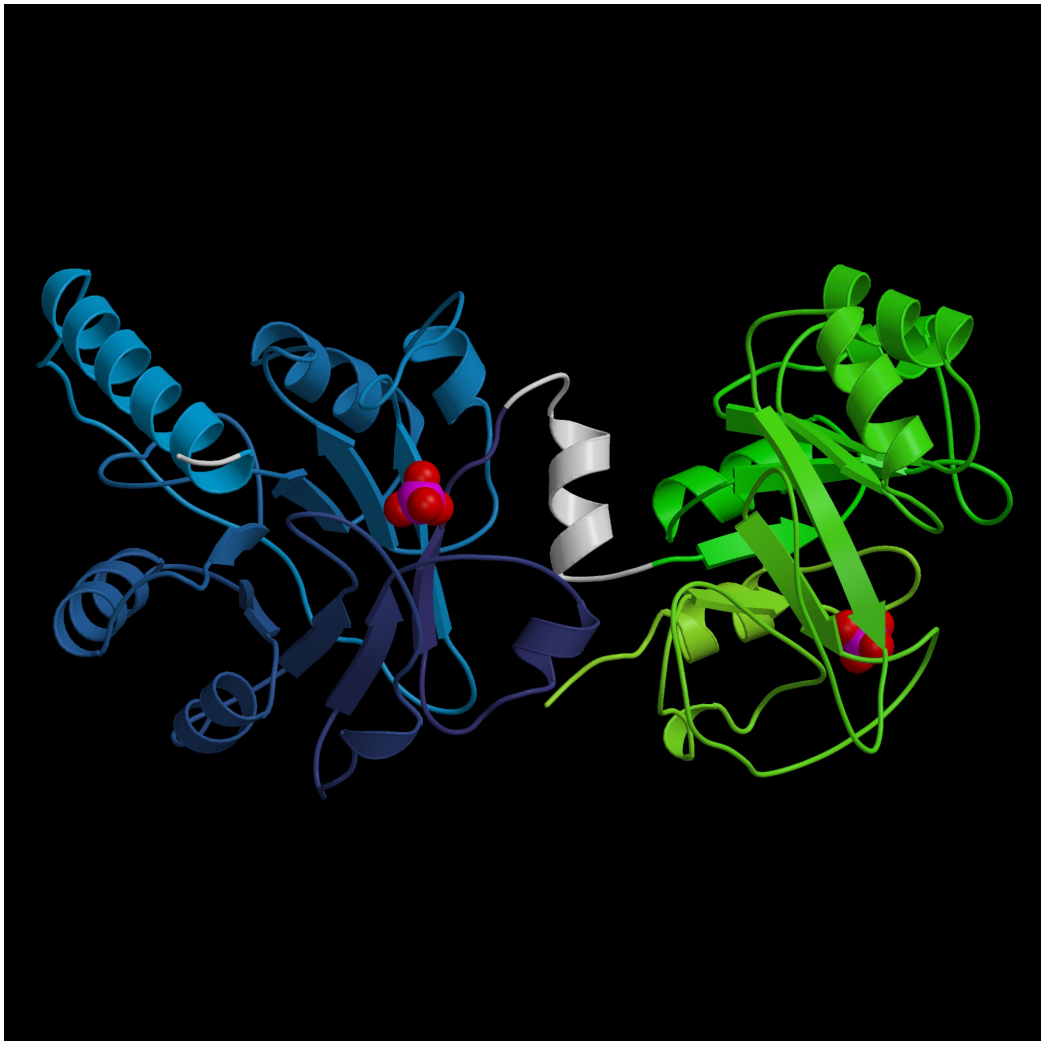


Figure 4.3: Three-dimensional structure of the *E. coli* fusion protein 1223.PRO (PDB accession 1PII). The blue section of this structure corresponds to the unfused TrpC protein in *Synechocystis sp.* and *Thermotoga maritima*. The green section corresponds to the unfused TrpF protein from these genomes. The region of the structure coloured white represents a region of this protein which is not homologous to the unfused component proteins from these genomes. The corresponding alignment of these proteins is shown in Figure 4.4. This structure was drawn using MolScript and Raster3D (Kraulis, 1991; Merritt and Murphy, 1994).

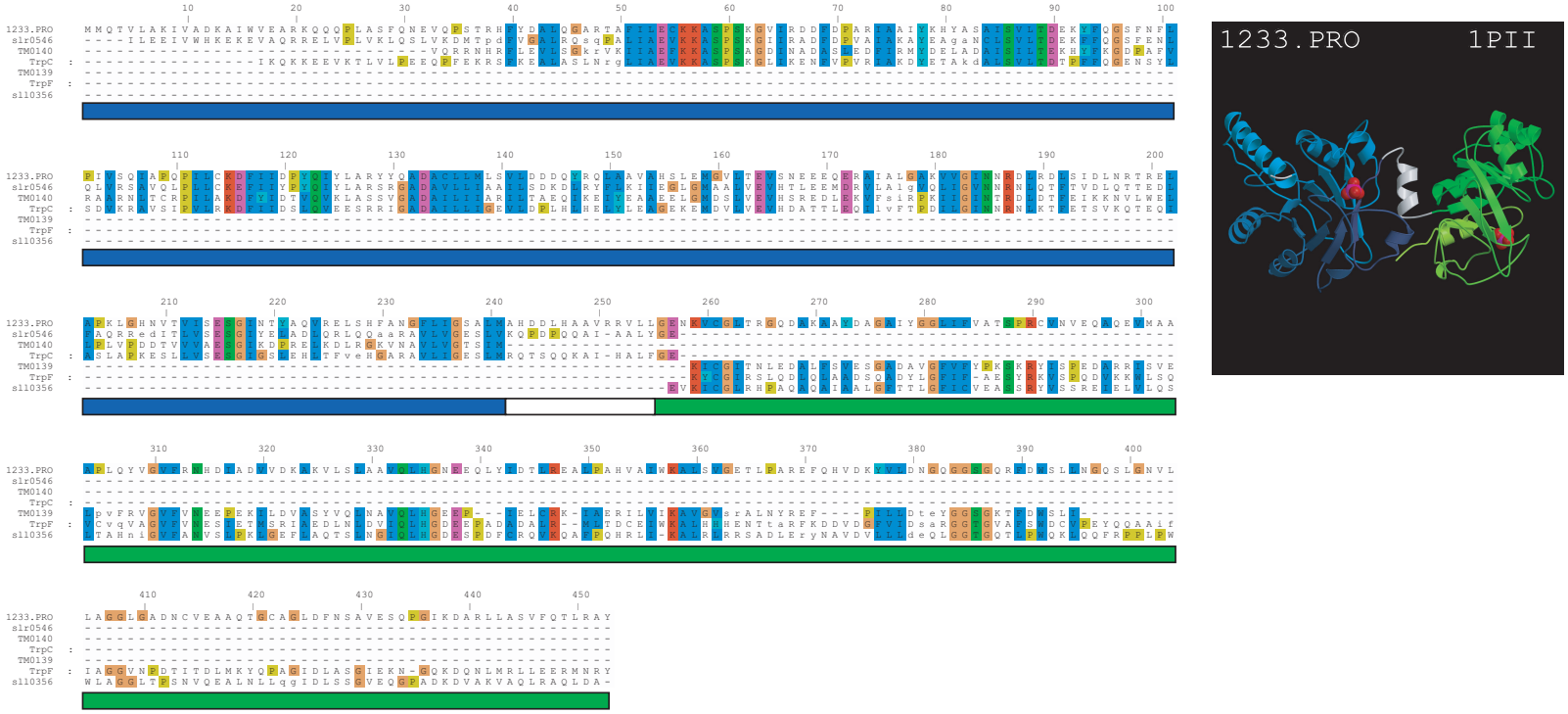


Figure 4.4: Alignment illustrating the fused *E. coli* protein (1233.PRO) which is a result of the fusion of TrpC and TrpF genes. Unfused TrpC and TrpF proteins from the complete genomes of *Thermotoga maritima*, *Synechocystis sp.* and *Bacillus subtilis* are aligned to the fused *E. coli* protein. Unfused TrpC proteins align to the N-terminal end (blue alignment region) of the *E. coli* fusion protein, while TrpF proteins align to the C-terminal end (green region). The inset figure shows the positions of these two alignment regions on the 3-dimensional structure of this *E. coli* fusion protein (PDB accession 1PII). The white regions on both the alignment and the structure show the unaligned linker peptide between these two regions. This alignment is generated from BLAST results using the Mview program (Brown et al., 1998).

Accurate detection of gene-fusion events in complete genomes can hence be used to infer functional links and direct physical interaction between proteins whose genes are not proximal, and is also applicable to eukaryotic genomes (Enright et al., 1999). Clearly this method possesses many advantages over conventional gene co-localisation methods. The term 'interaction' is used henceforth to imply either direct physical interaction or an indirect functional association (for example, involvement in the same biochemical pathway or similar gene regulation).

In order to exploit this concept of protein interaction prediction through gene-fusion, an accurate algorithm for the detection of these fusion events was developed. This algorithm (DifFuse) detects fused *composite* proteins in a reference genome with domains that correspond to individual full-length *component* proteins in other genomes. The underlying assumption is that if a composite protein is uniquely similar to two component proteins in another species (which may not necessarily be encoded by neighbouring genes) the component proteins are most likely to interact. Complete genome sequences are used for the identification of these fusion events because this allows the detection of orthologous proteins across species.

4.2.1 An Algorithm for the Detection of Gene Fusion Events

The most reliable prediction of protein-protein interactions is that within complete genomes, and only databases that have this property are considered. The initial input for the algorithm is two such genomes. The genome containing unfused component protein sequences is termed the *query genome* and the genome in which corresponding full-length fused composite protein sequences are sought is termed the *reference genome*. Generalisations of this formalism are possible and query and reference databases may be interchangeable. The advantage of performing this analysis only within complete genomes is that these data are both comprehensive (no better candidate may be identified) and unbiased (no cases occur in which more fusions will be detected because of experimental sampling).

In order to accurately detect gene-fusion events in complete genomes the algorithm was designed to identify a full length composite protein sequence in a reference genome, that corresponds to two separate (unrelated) component proteins in a query genome. This situation is shown in Figure 4.5 (inset). A

flow chart of the algorithm is shown in Figure 4.5 and the algorithm can be described as follows.

- The query database is compared against itself using BLASTp (v 2.0) (Altschul et al., 1997), after masking compositionally biased regions using the CAST algorithm, which is described in Section 5.1 (Promponas et al., 2000), and all pairwise sequence similarities are recorded in a binary matrix T (Figure 4.5).

- The matrix is symmetrified (Rivera et al., 1998; Enright and Ouzounis, 2000), using a Smith-Waterman (Smith and Waterman, 1981) dynamic-programming alignment algorithm (Pearson and Miller, 1992), which is executed only for non symmetrical pairs.

- The query database is also compared against the reference database, as above, and similarities are recorded in binary matrix Y (Figure 4.5).

- For all entries C in the reference database, entry pairs (A,B) from the query database deemed to be similar to reference entry C are collected (Figure 4.5).

- Every pair (A,B) similar to C is looked up in the self-comparison matrix T. If dissimilar, it is further checked for similarity using a second dynamic programming alignment pass to eliminate the possibility that it was a false-negative case during the initial self-comparison phase. All Smith-Waterman runs are executed an additional 25 times, with randomisation of the sequences, and a Z-score is obtained:

  - If this Z-score is lower that a certain threshold, then this situation represents a false-negative BLAST similarity assignment which is ignored.
  - If this Z-score is higher than a certain threshold, the similarity is accepted as significant. In this case A and B in the query database are candidates for a fusion event and can be predicted to interact.

The key abstraction is that a candidate pair (A,B) of query proteins can either represent a false-negative similarity assignment, or a component pair matching the composite protein C, and representing components of a gene-fusion event. This is very similar to the multi-domain detection strategy

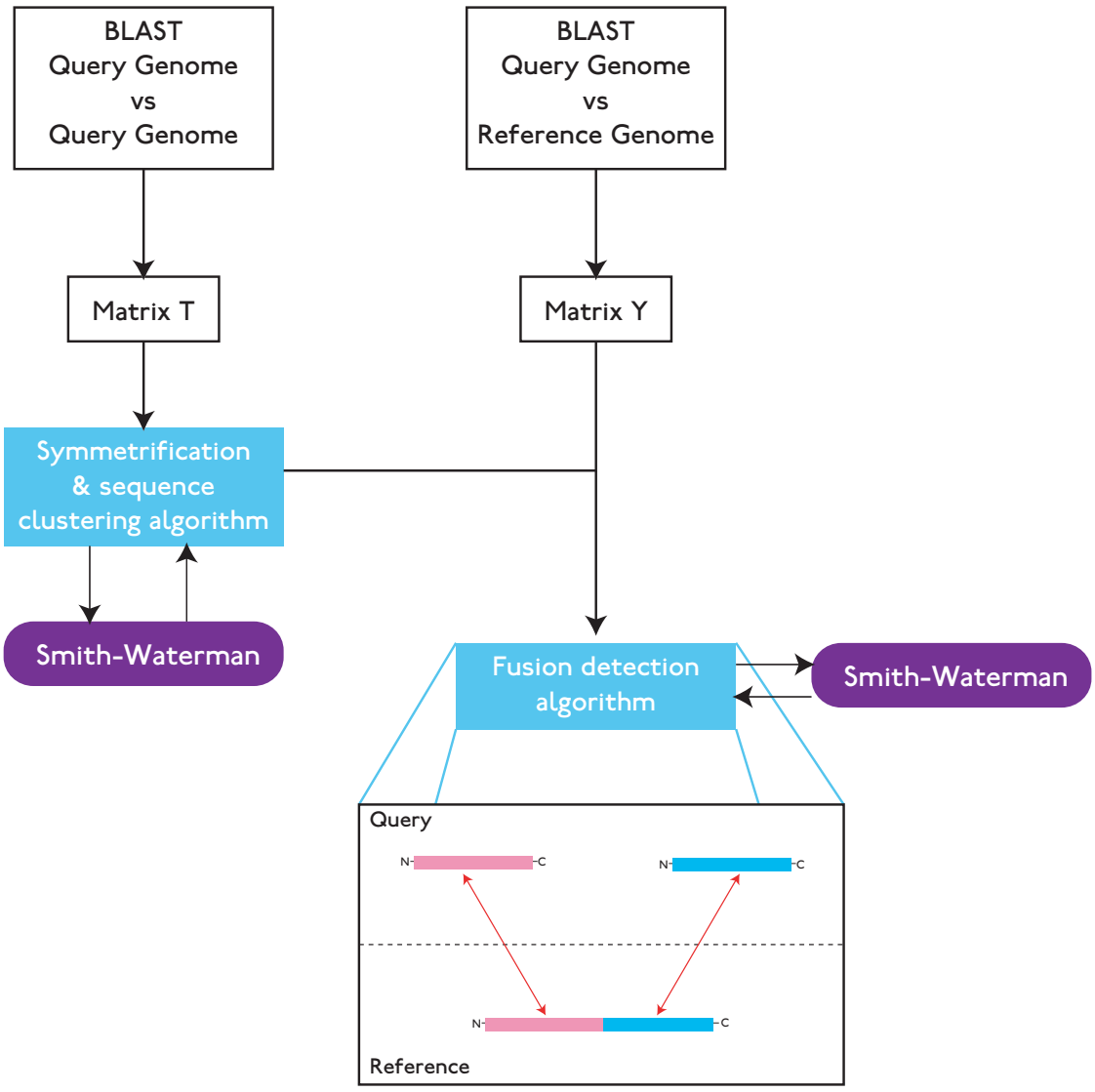


Figure 4.5: Flowchart of the DifFuse algorithm. This protocol is fully described in the accompanying text.

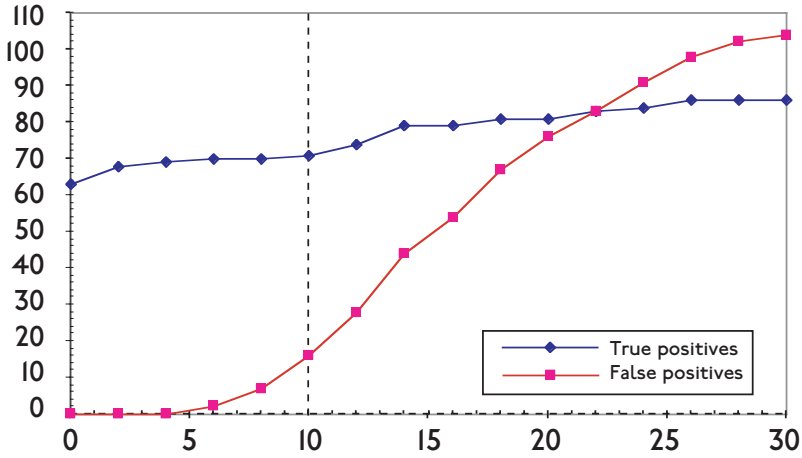


Figure 4.6: The dependence of the number of true and false positive hits with respect to the Z-score threshold used. Dashed line indicates the threshold used. True positives are 'unrelated' genes that are involved in a fusion event and are candidates for pairwise interactions; false positives are 'related' genes that are also accepted. The decision of whether two genes are related or not (in terms of sequence similarity) depends on the Z-score value: the higher that value is, the more permissive the criterion becomes for two related genes to be considered. True positives are all cases listed in Table 4.1. False positives were eliminated manually by inspection of the alignment overlap between the two component genes. For Z-scores < 10, all cases have been manually and automatically verified. For Z-scores > 10, all cases have been only automatically checked; therefore, these values represent an upper bound for true positives.

of GeneRAGE (described in Section 2.3.3). It is apparent that, although the coverage of the query database may strongly depend on the reference database, the precision can be very high. Unfortunately, no pertinent data set exists to estimate precision accurately. Precision and coverage of the algorithm are controlled by the Z-score parameter setting (Figure 4.6).

4.2.2 Application and Validation of the Algorithm using Four Complete Genomes

In order to test the effectiveness of the DifFuse algorithm it was applied systematically to the complete genome sequences of *Saccharomyces cerevisiae*, *E. coli*, *H. influenzae* and *M. jannaschii*. Analysis of these four genomes al-

lows the effectiveness of the algorithm for the detection of gene-fusion events and protein-protein interaction to be assessed. The algorithm was applied as described above, by comparing the complete genomes of *E. coli*, *H. influenzae* and *M. jannaschii* against *S. cerevisiae* as a reference genome. Subsequently each of the three query genomes were compared against each other as both query and reference genomes. The total number of such analyses is hence: $(3 \times 4) + (3 \times 3) = 21$.

These 21 analyses identified a total of 215 component proteins in the 3 query genomes that are involved in 88 fusion events (of which 64 are unique) (Figure 4.7; triangles in Table 4.1). There are 39 fusion events in *E. coli*, 24 in *H. influenzae* and 25 in *M. jannaschii*, with 2.44 proteins per fusion event on average due to both multiple-fusion events (for example: case 8 in Table 4.1), and paralogous genes (for example: case 21 in Table 4.1). Also detected were 94 composite proteins in the 4 reference genomes, representing 77 fusion cases or protein families (circles in Table 4.1). This is due to the presence of 17 paralogous composite proteins. The *E. coli* genome contains only 24 composite proteins (18 fusion-protein families) with reference to the component proteins, as opposed to the much smaller *H. influenzae* genome which contains 25 composite proteins (23 fusion-protein families). In contrast, the *M. jannaschii* genome has only 9 composite proteins (8 fusion-protein families). The remaining 36 composite proteins (28 fusion-protein families) are detected in the genome of *Saccharomyces cerevisiae*. Because query genomes were in turn used as reference genomes, there are eight cases detected as shared between them: *E. coli* and *H. influenzae* share six; *E. coli* and *M. jannaschii* none; and *H. influenzae* and *M. jannaschii* share two (Table 4.1).

There are only three multiple-fusion events: case 8 deriving from the yeast ARO1 gene (Duncan et al., 1987; Duncan et al., 1988), case 15 deriving from the yeast URA2 gene (Denis-Duphil, 1989) and case 39 based on the E. coli gene B2282, of unknown function (Table 4.1). The total number of possible pairwise interactions can be as many as 122 (column N in Table 4.1), depending on the degree of paralogy for certain proteins, which introduces some uncertainty in the prediction. Paralogy in the component proteins increases the number of possible interactions (for example, case 45), thereby decreasing the certainty of the prediction. Conversely, paralogy in the composite proteins increases the certainty that the component proteins interact, because the fusion event is repeatedly observed within (or even across) genomes. Notably, as many predicted interactions occur between

Case	Component	Component	Composite	EC	HI	MJ	SC	N
1	GalE	GalM	GAL10	▲▼	▲▼		●1	2
2	AccC	B0712-hypothetical	DUR1,2	▲▼	▲▼		●1	2
3	Hypothetical	Hypothetical	PYC2,PYC1			▲▼	●2	1
4	HisH	HisF	HIS7	▲▼	▲▼	▲▼	●1	3
5	HisI(E)	HisD	HIS4	▲▼	▲▼	▲▼	●1	3
6	RpoA 9	RpoA 0	RPO21,RPO31,RP A190			▲▼	●3	1
7	GltB	GltD	GLT1	▲▼			●1	1
8	AroB/AroA/AroK/AroD/AroE	Multiple fusion	ARO1	▲▼▲▼▲	▲▼▲□□	□▼□□▲	●1	3
9	Aconitase subunit	Aconitase subunit	LYS4			▲▼	●1	1
10	ArgA	ArgC	ARG5,6	▲▼			●1	1
11	LeuC	LeuD	LEU1	▲▼	▲▼	▲▼	●1	3
12	TrpA	TrpB	TRP5	▲▼	▲▼	▲▼	●1	3
13	PurD	PurM	ADE5,7	▲▼	▲▼	▲▼	●1	3
14	PurL	PurQ	ADE6	●1	●1	▲▼	●1	1
15	CarA/CarB/PyrB	Multiple fusion	URA2	▲▼▲		▲▼▲	●1	2
16	B1378	CysI	ECM17	▲▼			●1	1
17	TrpG	TrpC	TRP3	▲▼	▲▼	▲▼	●1	3
18	AgaG	AgaF	HyuA,HuyB			▲▼	●1	1
19	IlvG_1	IlvG_2	ILV2	▲▼	●1	●2	●1	1
20	GmpA	GmpB	GUA1	●1	●1	▲▼	●1	1
21	GyrB,ParE	GyrA,ParC	TOP2	▲▲▼▼	▲▲▼▼		●1	4
22	FolK	FolP	Folate biosynthesis (probable)	▲▼	▲▼▼		●1	3
23	PabA	PabB	ABZ1	▲▼	▲▼		●1	2
24	PurK	PurE	ADE2	▲▼	▲▼	▲▼	●1	3
25	RpoB 0	RpoB 9	RPB2,RET1,A135	●1	●1	▲▼	●3	1
26	ThiE	ThiM	THI6		▲▼		●1	1
27	ThiD	TenA	thi21,thi20,thi22		▲▼		●3	1
28	TklA	TklB	TKL1,TKL2	●2	●1	▲▼	●2	1
29	LysC	hom	ThrA,MetL	●2	●1	▲▼		1
30	ABC transporter	Hypothetical	B0879	●1		▲▲▼▼		4
31	Hypothetical	Putative methyltransferase	B0948	●1		▲▼▼		2
32	TrpG	TrpD	TrpD	●1	▲▼	▲▼		2
33	FumA	FumB	FumA	●1		▲▼		1
34	Hypothetical tkt	PheA	PheA	●1	●1	▲▼		1
35	FprA	Rubredoxin	B2710	●1		▲▲▲▼▼		6
36	TrxM	Hypothetical	B0492	●1	▲▼			1
37	Hypothetical	Hypothetical	B1816,B2063	●2	▲▼▼▼			3
38	CpxR,YgiX	TyrR	AtoC,YfhA,Gl nG,HydG	●4	▲▲▼			2
39	Hypothetical	Multiple fusion	B2324	●1	▲▼▲			1
40	Hypothetical	Hypothetical	B2474	●1	▲▼			1
41	Hypothetical	Hypothetical	Su-	●1	▲▼			1
42	HemX	Hypothetical	HemX	●1	▲▼			1
43	TrpC	TrpF	TrpC		●1	▲▼		1
44	CitX	CitG	CitG	▲▼	●1			1
45	SbmA	Hypothetical	ABC transporter/ATP-binding	▲▼▼▼▼▼▼▼	●1			7
46	B3777	B3776	Hypothetical	▲▼	●1			1
47	B2612	YfjD	Hypothetical	▲▼	●1			1
48	YgfQ	YgfR	Hypothetical	▲▼	●1	●1		1
49	YabK	B0263	Hypothetical	▲▼	●1			1
50	YhaQ	YhaP	SdaA	▲▼	●1			1
51	YbfH	YbfG	Hypothetical	▲▼	●1			1
52	PurF	YhfN	GlmS	▲▼	●1			1
53	FrwB,FrwD	FrwC,B2386	FruA	▲▲▼▼	●1			4
54	UgpC	YtfS	RbsA,MglA	▲▼	●2			1
55	B1515,B1899	AraH	RbsC,MglC	▲▲▼	●2			2
56	NrfF	NrfG	NrfF	▲▼	●1			1
57	MsrA	B1778	MsrA	▲▼	●1			1
58	SbmA	Hypothetical	ABC transporter/ATP-binding	▲▼▼▼▼▼▼▼	●1			1
59	YhgK	YhgJ	Probable RNA cyclase	▲▼		●1		1
60	FrdB	GlpC	Iron-sulfur-binding reductase	▲▼		●1		1
61	RffH,RfbA,GalF,GalU	B0359	Glucose-1-P thymidyltransferase	▲▲▲▲▼		●1		4
62	B3016	B3015	Hypothetical	▲▼		●1		1
63	LeuS	YgjH	MetS	▲▼		●1		1
64	TopB	YrdD	TopA	▲▼	▲▼	●1		2
								122

The component gene/protein names (or identifiers) and the composite (fusion) gene/protein names (or identifiers) are listed. Columns EC, HI, MJ and SC correspond to E. coli, H. influenzae, M. jannaschii and S. cerevisiae, respectively; N lists the maximum number of possible pairwise interactions taking into account paralogy in the query genomes (multiple-component cases are counted as a single interaction). Symbols represent a corresponding component or composite genes/proteins: triangle pairs, ▲▼, a pair of component proteins in the query genome predicted to interact based on their similarity to a composite protein in the reference genome; alternating triangles, ▲▼▲/▲▼▲▼▲, multiple-component genes/proteins (cases 8, 15 and 39); open squares, □, absence of a component from a multiple-fusion event (case 8); consecutive triangles, ▲▲... /▼▼... , the exact number of detected paralogous component genes/proteins in the query genome; filled circles, composite genes/proteins, the number represents the number of paralogous composite genes/proteins in the reference genome. The sort order follows the three species against the composite-protein sequence identifiers for the yeast genome, and then the other three species in succession. Genes are named where possible; where none is available, the sequence identifier is used instead. All fusions were confirmed by reverse BLAST searches using the composite protein as query, which identified all the component proteins. Note that functional annotation is not necessary but frequently useful in resolving paralogous cases (for example, case 21). Predictions imply functional associations and not necessarily direct molecular interactions.

Table 4.1: Summary of the 64 detected gene fusions. A detailed description and legend is given at the bottom of the table.

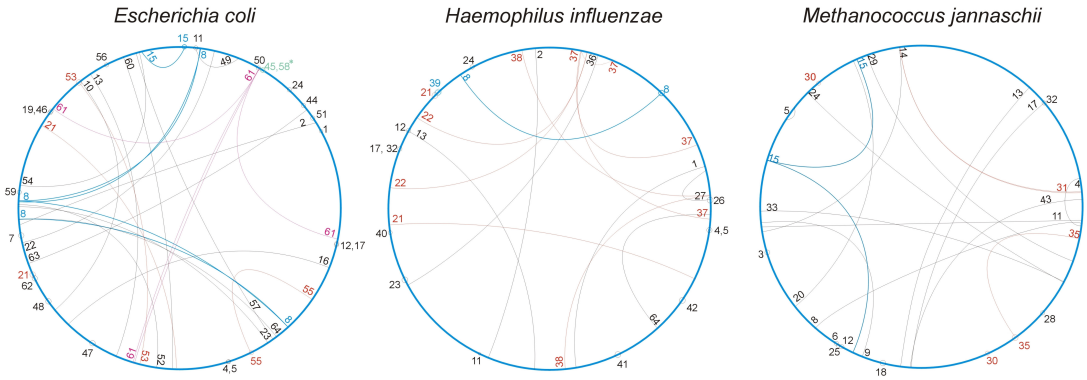


Figure 4.7: Representation of protein interaction maps for the most likely interactions predicted for *E. coli*, *H. influenzae* and *M. jannaschii*. In the large blue circles, which represent the three genomes, 0° corresponds to the first base pair, and 360° the last base pair of the genome. Predicted interactions are indicated by linking the circular map positions of the genes involved. In cases of neighbouring genes ($< 5°$), a small circle indicates the predicted interaction between two genes at that region; otherwise, an arc links the two genes in question. Multiple interactions are not cross-labelled. Some paralogous cases are resolved and only the most likely case is indicated by an arc. All cases are numbered according to Table 4.1. Predictions are colour coded: black, pairwise interactions; blue, multiple interactions; red/purple, cases where, due to paralogy, more than one pairwise interaction is possible (red, two possibilities; purple, more than two possibilities); green (marked by asterisk), because of a large number of paralogues, no interaction can be easily resolved. The source of the prediction (composite protein from a given species) is not indicated.

products of distant genes as between products of neighbouring genes (Figure 4.7): there are only 8 interactions between products of neighbouring genes in *M. jannaschii*, 14 in *H. influenzae* and 18 in *E. coli* (Figure 4.7). This underlines the potential of this method to identify interacting proteins in complete genome sequences beyond the simple proximity constraint (Dandekar et al., 1998; Overbeek et al., 1999). Some of the interactions between products of neighbouring genes are cases of well-known gene pairs (for example, case 6, the DNA-dependent RNA polymerase A'/A" pair in *M. jannaschii*), although there are also cases that may be sequencing artefacts (for example, case 19 in Table 4.1, which is *intact* in all species except *E. coli*).

The predicted interactions display some complex patterns of distribution along the genome sequences (Figure 4.7). For example, certain regions seem to contain a disproportionate number of genes or proteins involved in fusion events (for example, cases 12, 17 and 32 representing the tryptophan biosynthesis gene cluster). Also, some intricate symmetries occur, deriving from multiple-fusion events or paralogous proteins (for example, cases 8 and 61 in E. coli, respectively).

The method identifies a number of well-known interacting partners (for example, cases 4 and 13). In total, 26 out of the 64 cases (40%) listed in Table 4.1 are involved in the same protein complex or biochemical process. A number of unconfirmed cases (for example, cases 31, 57 and 60 from Table 4.1) constitute some interesting, testable predictions; for example, case 60 represents a predicted interaction between gene products FrdB (fumarate reductase) and GlpC (heterodisulfide reductase) in *E. coli*. In total 85 fusion events were detected of which 64 are unique and 21 are false positives (Figure 4.6); thus, the precision of the method can be estimated as 75% (64/85) with the parameter settings used for this analysis.

Coverage cannot easily be estimated, as it is not known in advance how many proteins potentially interact within the query genome. A standard set of fusion events to estimate coverage for the given query genomes is also not available. An attempt to estimate coverage was made by counting the number of false negatives. The maximum estimate for the coverage is as high as 95% (215/226) with the current parameter settings and the above assumption. Another false-negative case, fatty acid synthase (and its bacterial homologues) (Smith, 1994), is not detected due to low sequence similarity relationships. Precision and coverage can be controlled by modifying the cut-off scores in the post-processing of the homology searches as described

previously.

From a total of 7,768 protein sequences in the 3 complete query genomes, the minimum number of components involved in a multidomain-fusion event is 215, or 2.8%. Most of these proteins of known function appear to be metabolic enzymes, an effect possibly due to metabolic channelling of substrates (Welch and Easterby, 1994). These results, together with another study (Marcotte et al., 1999a), provided the first estimates for the frequency of gene-fusion events based on complete-genome comparisons. The exactness and high efficiency of the method makes it applicable to proteomics research, and complementary to continuing experimental approaches for the identification of protein interactions. This initial experiment proved the effectiveness of the method for the prediction of functional associations and interactions between proteins. In order to fully test the power of the method, a more ambitious experiment was required. The next section of this chapter details the use of the method for exhaustive detection of gene fusion events in many complete genomes.

4.3 Exhaustive Detection of Protein-Protein Interactions

The use of gene-fusion events to predict interactions and functional associations of proteins proved successful in our initial experiment (Enright et al., 1999; Sali, 1999). This experiment illustrated that the DifFuse algorithm was correctly predicting gene-fusion events with high accuracy, and that these gene-fusion events could be used to infer functional links between proteins in the absence of direct homology. Given the accuracy of the method and the availability of large numbers of completely sequenced genomes, we decided to employ our method exhaustively to much larger datasets. Such an analysis allows the exploration of the predictive power of the method, and the evolutionary prevalence of the gene-fusion phenomenon. In order to perform this exhaustive analysis a database of 24 complete genome sequences was assembled from public sources. These genomes are shown in Table 4.2. For an exhaustive analysis, each genome is taken as a query (i.e. the genome in which unfused component proteins are present) and compared against each one of the remaining 23 genomes as a reference (the genome containing fused composite proteins).

For this exhaustive analysis a number problems need to be addressed. One issue which became apparent from the initial analysis, was that paralogy in either the query or reference databases, can make an estimation of the number of fusion events difficult. A fusion event in an organism leads to the creation of a single fusion protein, yet through the process of gene duplication, multiple instances of this fusion protein may become apparent. Our method will hence detect multiple fusion events. In order to tackle this problem, we employed the GeneRAGE sequence clustering algorithm (Enright and Ouzounis, 2000), which is described in Section 2, to both fused (composite) and unfused (component) proteins predicted to be involved in fusion events by the DifFuse algorithm. This analysis detects component and composite protein families, and allows multiple paralogues from a single fusion event to be combined into a single fusion family or class. The analysis of the distribution of these gene fusion classes among genomes allows us to investigate the dynamics and distribution of this evolutionary process and to assess the extent of the predictive power of the approach.

Another problem with exhaustive analysis is the computational cost of applying the DifFuse algorithm to 24 complete genomes. This necessitates

Organism name (strain)	Number of ORFs	ID
Aeropyrum pernix(K1)	2,694	aerpe
Aquifex aeolicus(VF5)	1,522	aquae
Archaeoglobus fulgidus (DSM4304)	2,409	arcfu
Bacillus subtilis(168)	4,100	bacsu
Borrelia burgdorferi (B31) + plasmids	1,639	borbu
Caenorhabditis elegans	19,099	caeel
Chlamydia pneumoniae (CWL029)	1,052	chlpn
Chlamydia trachomatis (serovar D)	894	chltr
Drosophila melanogaster	13,710	drome
Escherichia coli (K12- MG1655)	4,290	escco
Haemophilus influenzae(KW20)	1,707	haein
Helicobacter pylori (26695)	1,577	help2
Helicobacter pylori(J99)	1,495	helpj
Methanococcus jannaschii (DSM 2661)	1,773	metja
Methanobacterium thermoautotrophicum (delta)	1,871	metth
Mycoplasma genitalium (G-37)	479	mycge
Mycoplasma pneumoniae (M129)	677	mycpn
Mycobacterium tuberculosis (H37Rv)	3,924	myctu
Pyrococcus horikoshii (shinkaj OT3)	2,061	pyrho
Rickettsia prowazekii (Madrid E)	837	ricpr
Saccharomyces cerevisiae (S288C)	6,305	sacce
Synechocystis sp. (PCC 6803)	3,168	synsp
Thermotoga maritima(MSB8)	1,849	thema
Treponema pallidum (Nichols)	1,030	trepa

Table 4.2: The species/strain names, the number of ORFs and the species name abbreviation used in all figures are given. References for each genome can be found elsewhere (Bernal et al., 2001).

that every genome is compared to every other genome with the BLAST algorithm, then these results are processed by the DifFuse algorithm. The total number of separate analyses at each stage is hence $23 \times 24 = 552$. The computational cost of applying the BLAST and Smith-Waterman algorithms to these complete genomes is by no means trivial, and requires a robust and parallel computational protocol (Figure 4.8).

4.3.1 Computational Protocol for Exhaustive Detection of Gene Fusion Events

The computational protocol developed for exhaustive detection of gene fusion will be described in this section. The protocol was designed to be as automatic and robust as possible, and requires no manual intervention from the user. The system takes genomes sequences as input, and produces annotated HTML output. This protocol is shown in Figure 4.8, and will be described in detail in this section.

Genome sequences

Complete genome sequences for the 24 species (Table 4.2) were obtained from their original sources (Bernal et al., 2001). The species names, number of ORFs and the identifiers used throughout this study are shown in Table 4.2. All sequences were stored in FASTA format for use as both query and reference genomes.

Genome comparison

All 24 genomes were filtered using the CAST compositional bias filtering algorithm (Promponas et al., 2000), which is described in Section 5.1, then compared against themselves and each of the other 23 genomes using the BLASTp (Altschul et al., 1997) sequence similarity searching algorithm with a cut-off E-value of $E \leq 1 \times 10^{-10}$. The DifFuse algorithm was then applied automatically to each genome in turn as a query against the other 23 (reference) genomes. Using other protein databases as reference yields fewer composite cases. For example, the well-known case of the TopA/TopB pair appears multiple times in such analyses, showcasing the extreme bias of annotated databases, such as SwissProt. Performing the same computation using the non-redundant sequence database (NRDB) is prohibitively

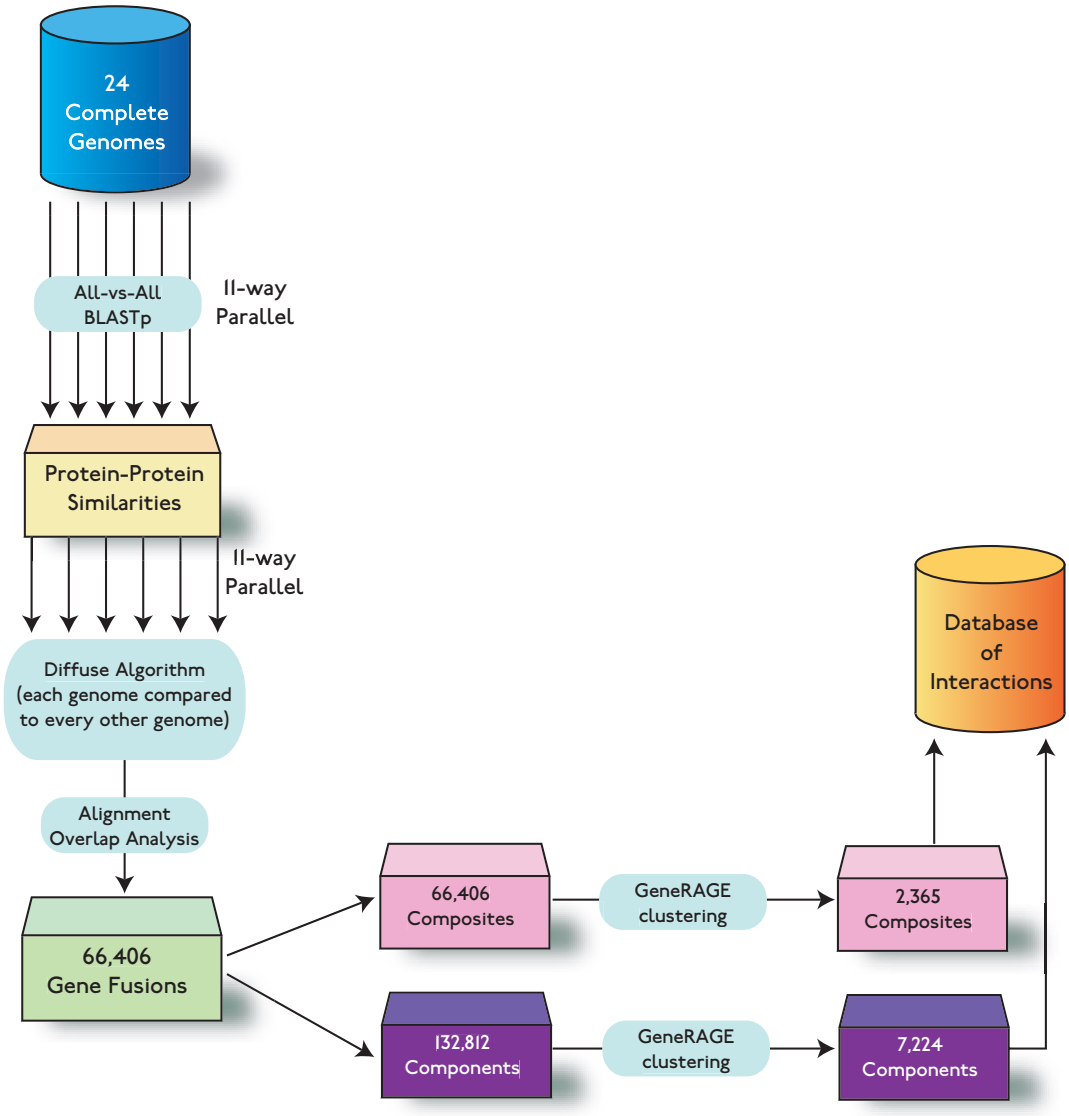


Figure 4.8: Protocol for the exhaustive detection of gene fusion events. This protocol is explained fully in the accompanying text.

expensive in terms of computation time for an analysis of this size. The detected gene fusion results for each of the 552 comparisons were further automatically filtered for significant overlap of the BLAST alignments of the component proteins. In this case, component proteins that overlap by more than 10% of their total length when aligned together with the composite protein. This step avoids the detection of *promiscuous domains* (Marcotte et al., 1999a; Enright et al., 2002) and gene prediction errors, which result in false-positive fusion detection cases. The detected component and composite proteins are far fewer in number than for the two previous reports regarding *E. coli* (Marcotte et al., 1999a) and *S. cerevisiae* (Marcotte et al., 1999b), due to the much stricter criteria employed in the present analysis and the multi-step protocol we have developed (Figure 4.8). This analysis was fully automatic and carried out in parallel over a period of four weeks on eleven SUN UltraSparc CPUs running Solaris 7.

Sequence clustering

All proteins involved in gene fusion events as either component or composite proteins were identified automatically from the results of the fusion analysis. From these data we obtain raw counts of the number of gene fusion events detected and the number of proteins involved in these events as either composite or component proteins. These figures are skewed however, due to the presence of paralogy in both the query and reference sets. Proteins involved in gene fusion events as either component or composite genes are then assembled into two lists. These lists are used to generate two sequence databases, the first one containing all component sequences from the 24 genomes and the second containing all composite sequences. These sequence databases of component and composite proteins are then compared against themselves using BLAST version 2.0 (Altschul et al., 1997) with a cut-off E-value of $E \leq 1 \times 10^{-10}$). Sequences are then clustered according to their detected similarity using the GeneRAGE algorithm. GeneRAGE lists all composite and component proteins in clusters according to their detected similarity and domain structure. Homologous proteins with similar domain structure were clustered together. Each cluster in this case indicates a distinct class of fusion event and cluster members indicate which proteins are involved in this type of event from different genomes. These clusters are used to calculate the number of unique fusions detected within and across genomes. This is

done by examining how many distinct types of fusion are present in any given genome.

Data Storage

All results from this analysis are automatically processed and composite with component fusion alignments are constructed. All information is then stored in a MySQL database which is accessible via Perl scripts locally or using a world wide web interface[2].

4.3.2 Results

The detection of gene fusion events yielded 132,812 component and 66,406 composite proteins in an all-against-all genome comparison. The three-dimensional structures of four of these detected composite proteins are shown in Figures 4.9 & 4.10. Many detected fusions represent multiple occurrences of the same proteins across species. After sequence clustering with GeneRAGE, there are 7,224 component and 2,365 composite fusion families across the 24 species (an 18 and 28 fold reduction respectively). The multiple detection of these cases within or across genomes signifies that the majority of components and composites are observed more than once and therefore represent genuine cases (as opposed to sequencing artefacts, which are usually isolated cases).

The high precision of the method allows the prediction of 39,730 unique pairwise functional associations (or possibly physical interactions) of the components with reference to the composite protein set. Eighty six percent of the 66,406 predicted associations obtained from the total number of composite proteins yield a Z-score value of less than 3 (Figure 4.11), previously shown to result in virtually no false-positive cases (Enright et al., 1999). This increased precision is due to the introduction of an additional constraint that does not permit any overlap between the component proteins. All the above results are available on the world wide web. Some of these interactions are known, but we estimate that more than half of them are newly detected cases, testable with functional genomic and proteomics techniques (Ito et al., 2000).

[2]http://maine.ebi.ac.uk:8000/allfuse/

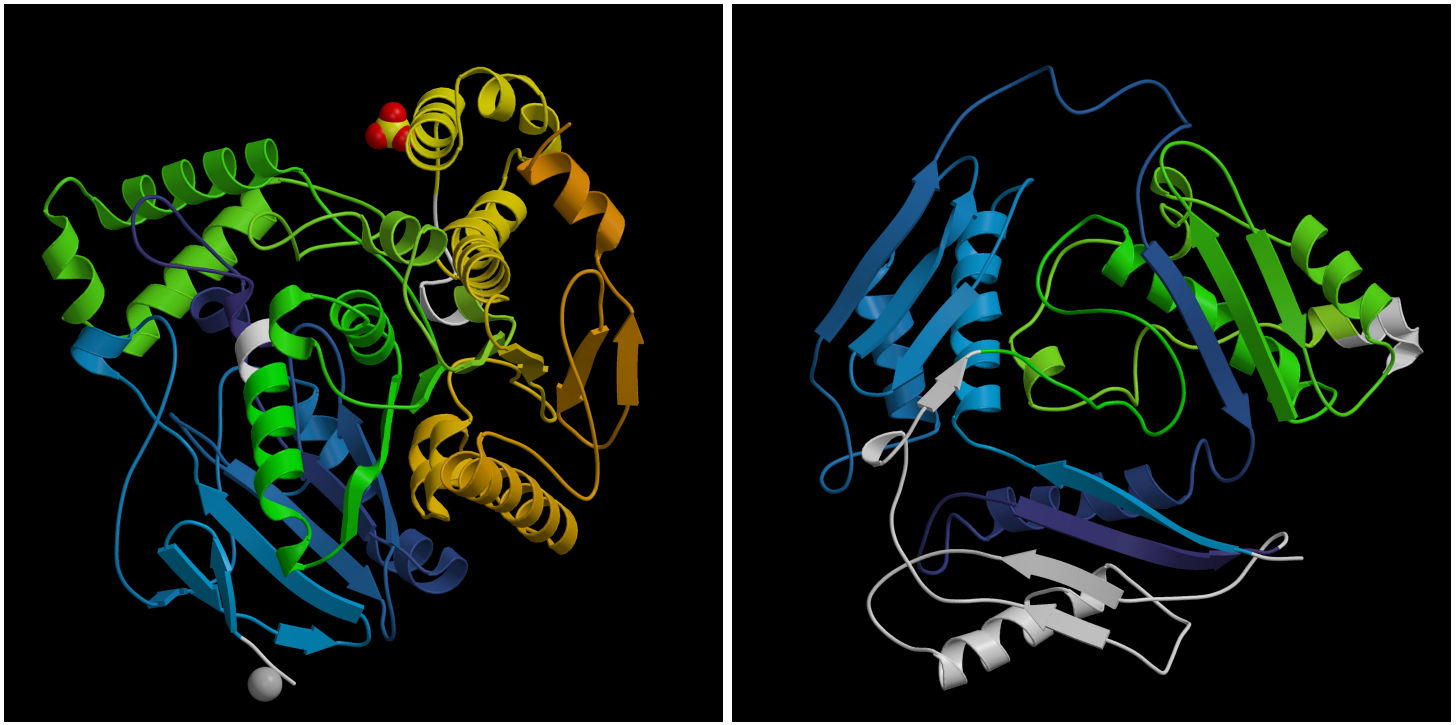


Figure 4.9: Three-dimensional structures of two example composite proteins which are present in PDB. The first structure (1C7J Chain A) represents the *B. subtilis* triple fusion protein *PnbA* (Paranitrobenzyl esterase). The blue and green regions of the structure are unfused (component) proteins in *C. elegans* and *M. tuberculosis*, the orange region is also a separate protein in *M. tuberculosis*. The second structure (1FUG Chain A) represents an *E. coli* fusion protein (2877.PRO; S-adenosylmethionine synthetase; *MetR*). The blue and green regions correspond to separate unfused component proteins in *C. elegans*. Figures are produced using MolScript and Raster3D (Kraulis, 1991; Merritt and Murphy, 1994).

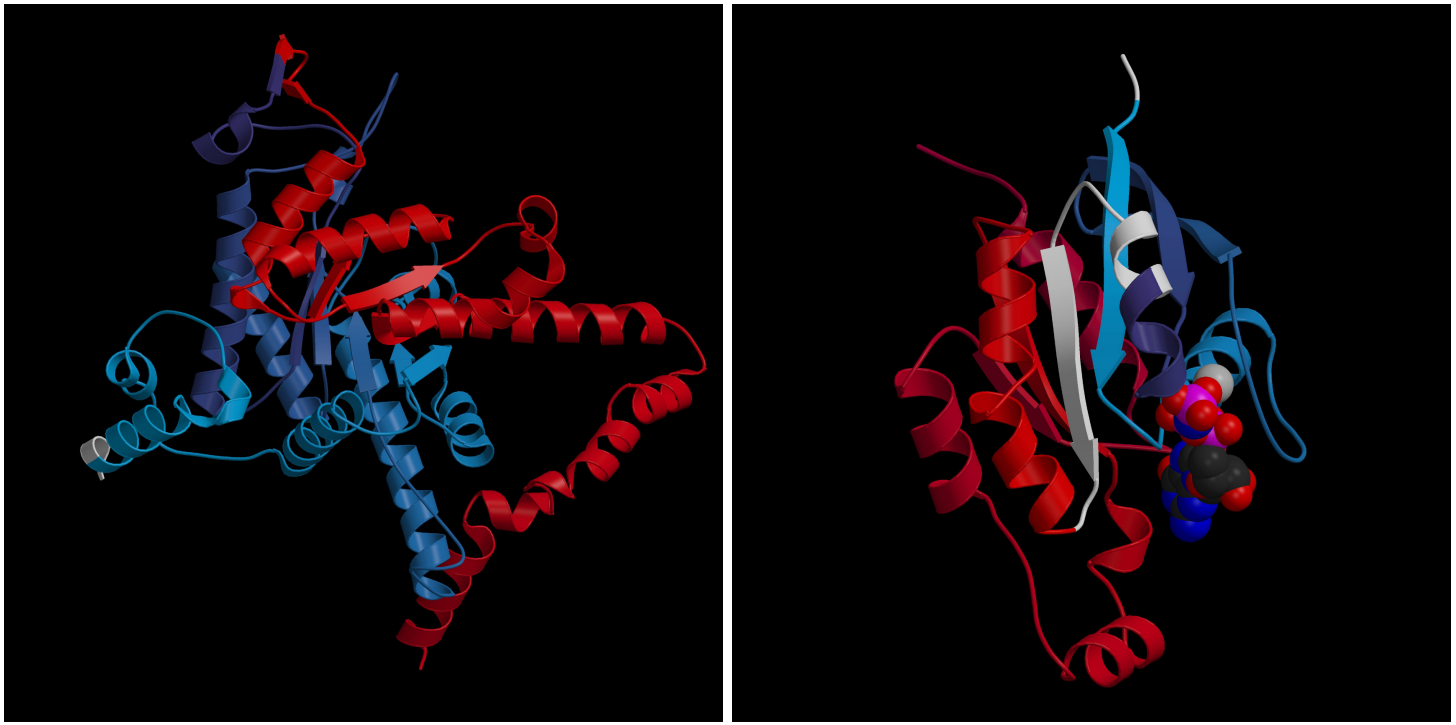


Figure 4.10: More examples of three-dimensional structures of composite proteins which are present in PDB. The first structure (1IGW Chain C) is of the *E. coli AceA* protein (3910.PRO; Isocitrate lyase) The blue and red regions correspond to unfused component proteins in *M. tuberculosis*. The second structure (1MH1) is a small G-Protein in *H. sapiens*, which is almost identical (93% identity) in sequence to the *D. melanogaster* composite protein (FBan00002248). The red and blue regions correspond to unfused proteins in *C. elegans*. Figures are produced using MolScript and Raster3D (Kraulis, 1991; Merritt and Murphy, 1994).

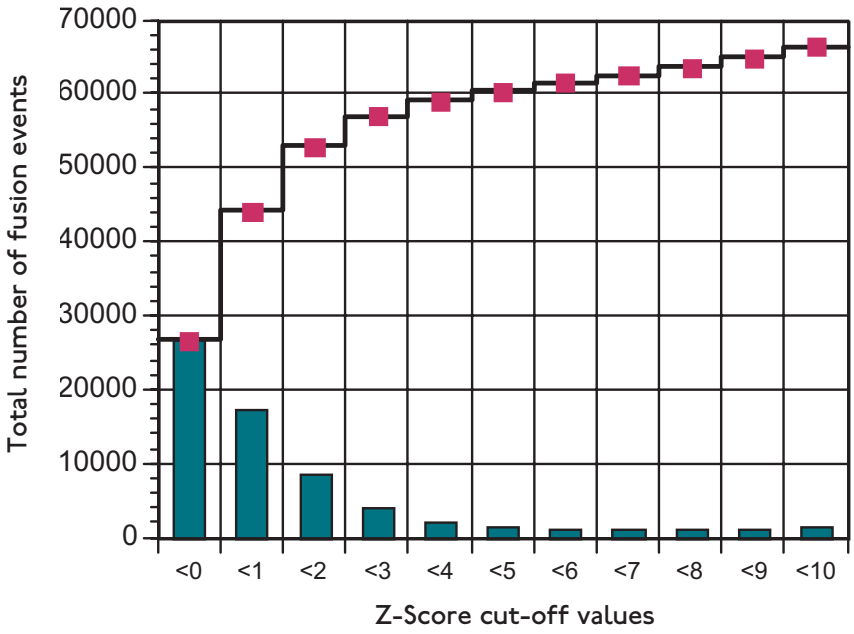


Figure 4.11: The graph illustrates the Z-score (blue bars) distribution and its cumulative sum (step function, with red rectangles) between components, for all detected fusion events (66,406 in total). The Z-score is a statistical measure of similarity for each pair of components. Components that have a Z-score similarity of less than 10, and both exhibit similarity to the same composite protein are detected as fusion events. In general, fusion events where the Z-score between components is less than 3 (marked by a vertical line) result in fewer false-positive fusion detections.

Validation

Currently, the only species for which predictions can be extensively validated is the yeast *Saccharomyces cerevisiae*, given the ongoing work on transcript profiling (DeRisi et al., 1997) and two-hybrid technology (Uetz et al., 2000). For yeast, there are 440 distinct component cases (predicted by all other genomes as reference, excluding some highly paralogous *Drosophila melanogaster* homologues) involved in 706 predicted interactions, most of which are detected by their homology to composite proteins from *Caenorhabditis elegans* and *D. melanogaster*. Two examples of predicted protein pairs that are known to interact are CPA1 (YOR303w) with CPA2 (YJR109c) (Lim and Powers-Lee, 1996) and MET3 (YJR010w) with MET14 (YKL001c) (Blaiseau et al., 1997), both derived from *C. elegans* homologues.

We have attempted to test the validity of our predictions by comparing this set of components to a list of potentially interacting gene products, using results from a large-scale two-hybrid experiment (Uetz et al., 2000). However, there is only one case shared between the 1,004 proteins involved in 957 putative interactions detected by the two-hybrid system and the complete set of 706 pairs in this analysis: YIL033C (SRA1) and YKL166C (TPK3) matching the *C. elegans* protein C09G4.2 and *D. melanogaster protein* CT10911. This very low count of common pairs may be expected by the sampling biases of the two rather independent methodologies, given that each approach can only detect a very small subset of the total number of actual interacting pairs in yeast. Interestingly, based on a simple conditional probability calculation, an estimate for the total number of detectable interactions in the yeast cell may be of the order of 675,000.

Another validation procedure for the *S. cerevisiae* predictions was obtained by comparing all 706 component pairs against their expression profiles from publicly available gene expression data. For each of the pairs, a profile from 87 experiments involving cell cycle (Cho et al., 1998), sporulation (Chu et al., 1998) and diauxic shift (DeRisi et al., 1997) was used to determine whether expression data corroborated our predictions for the association of the component proteins. The analysis was carried out as follows:

- Gene expression ratios for all experiments were transformed into log-odds values so that induction and repression measurements are directly comparable (positive and negative values, respectively).

- The log-odds values were then normalised across all time points for each experiment, using Z-score values. The Z-score values for all time points of each experiment thus allow cross-comparison of gene expression across separate experiments (DeRisi et al., 1997). Our predicted functional associations for *S. cerevisiae* with available expression data represent 536 component pairs in total.

- For each pair of proteins, a Pearson correlation coefficient was calculated between two corresponding experiments and averaged over all experiments. To estimate noise in these data, a control set of 536 randomly selected *S. cerevisiae* proteins was taken and treated as above (Figure 4.12).

- The distribution of averaged Pearson correlation coefficients for the predicted functional associations was compared against the distribution of coefficients for the control set using a t-test for mean values (where the null hypothesis is that the two means are equal). The test results in a t-value of 3.6 (critical t-value is 1.64), which is highly significant (P-value is 0.000173), indicating that there is a higher average correlation of expression profiles for the predicted functional associations against the background.

These results indicate that at least 20% of our predictions exhibit very strong correlations across gene expression experiments. The detected pairs of components from fusion analysis clearly exhibit similar patterns of expression for the above mentioned experiments (Figure 4.12 inset). Despite the noise level present in this gene expression data (a result of the limited number of experimental conditions available), there are twice as many predicted associations than random, above the threshold of average correlation value of 0.5. With a higher threshold of 0.55, precision is increased, with four times as many predicted associations over the random background. Above this value, 92 predicted functional associations (20% of 536 available pairs) exhibit high correlation across all experiments (Figure 4.12). Below that threshold, it is very difficult to estimate the precision rate of our predictions, because of the high level of noise and the rather limited number of publicly available gene expression data sets. This comparison between fusion detection and transcript profiling contrasts with previous approaches (Marcotte et al., 1999b), where

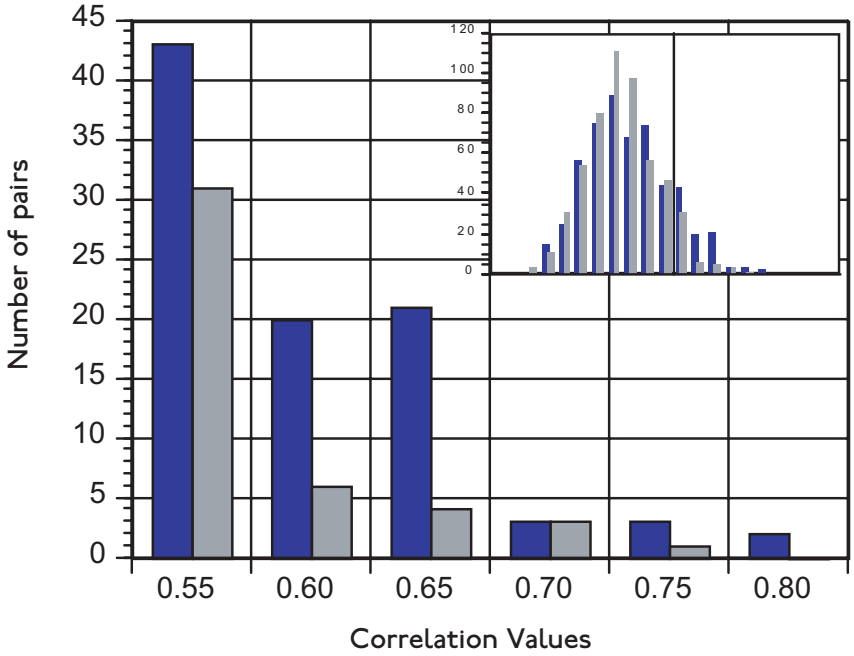


Figure 4.12: The graph illustrates the distributions of average correlation values of gene expression between component pairs (blue bars) and randomly selected pairs (grey bars), above a threshold value of 0.5. Inset: Distributions of average correlation values for both predicted and random associations (vertical line indicates the cut-off value of 0.5).

expression data were used as a filtering step for the detection of functional associations, and not as a validation criterion.

Novel Interaction Predictions

We have analysed the *S. cerevisiae* predictions and detected many interesting cases, which appear to be hitherto undetected functional associations or interactions between yeast proteins. Two of these interesting and novel predictions are discussed in some detail here.

First, MXR1 (peptide methionine sulfoxide reductase, involved in antioxidative processes) (Moskovitz et al., 1997) and YCL033C (function unknown) are predicted to be functionally associated by virtue of gene fusion in three other species - *Helicobacter pylori* (both strains), *Haemophilus influenzae* and *Treponema pallidum*. This observation is supported by experimental results (Lescure et al., 1999). MXR1 is 39% identical to the amino terminus of the

H. pylori composite proteins and YCL033C is 38% identical to the carboxyl terminus of these proteins. It appears that YCL033C is a selenoprotein, also homologous to the human SelX protein, which may be involved in scavenging reactive oxygen species (Lescure et al., 1999). These two proteins may be associated to protect the yeast cell from oxidative damage.

The second example is another interesting observation involving the yeast proteins MSS4 (phosphatidylinositol 4-phosphate kinase), which is involved in a signalling pathway responsible for the cell cycle dependent organisation of the actin cytoskeleton (Helliwell et al., 1998), and CCT3 (cytoplasmic chaperonin subunit gamma) which is involved in microtubule and actin assembly (Stoldt et al., 1996). A central domain of CCT3 is 25% identical to a large domain of *C. elegans* protein VF11C1L.1 and the carboxy-terminal domain of MSS4 is 29% identical to its carboxyl terminus. Thus, these two proteins are predicted to cooperate in cell cycle dependent cytoskeleton organisation and assembly.

Phylogenetic Distribution and Analysis of Fusion Events

The distribution of components and composites differs dramatically between species. There are 7,224 component cases, with an average of 350 cases per genome, exhibiting significant variation (Figure 4.13a, blue bars). The query genome sequences detected 2,365 composite cases, with an average of 115 cases per genome (Figure 4.13b, blue bars). Interestingly, we have observed some relatively small genomes containing composite proteins, which may yield predictions for components of higher organisms. For instance, there are 71 proteins (forming 30 families) in the *C. elegans genome* that match a fused protein in *Mycobacterium tuberculosis*. Two such examples are the component pair T06C10.1/C49H3.7 matching composite Rv0957 and the component pair W04C9.1/Y65B4B_12.b matching both composites Rv1272c and Rv1273c. Another clear prediction is the *S. cerevisiae* component pair YER052c/YJR139c (encoding HOM3/HOM6 respectively) matching composite MetL (3847.PRO) from *Escherichia coli* and other species. This has been a key observation that dictated the all-against-all genome comparison in this analysis. In other words, when species A is used as a query against species B, the resulting set of component and composite proteins is different from that with the reverse comparison, when species B is used as a query against species A. The three principal factors in gene fusion during evolution

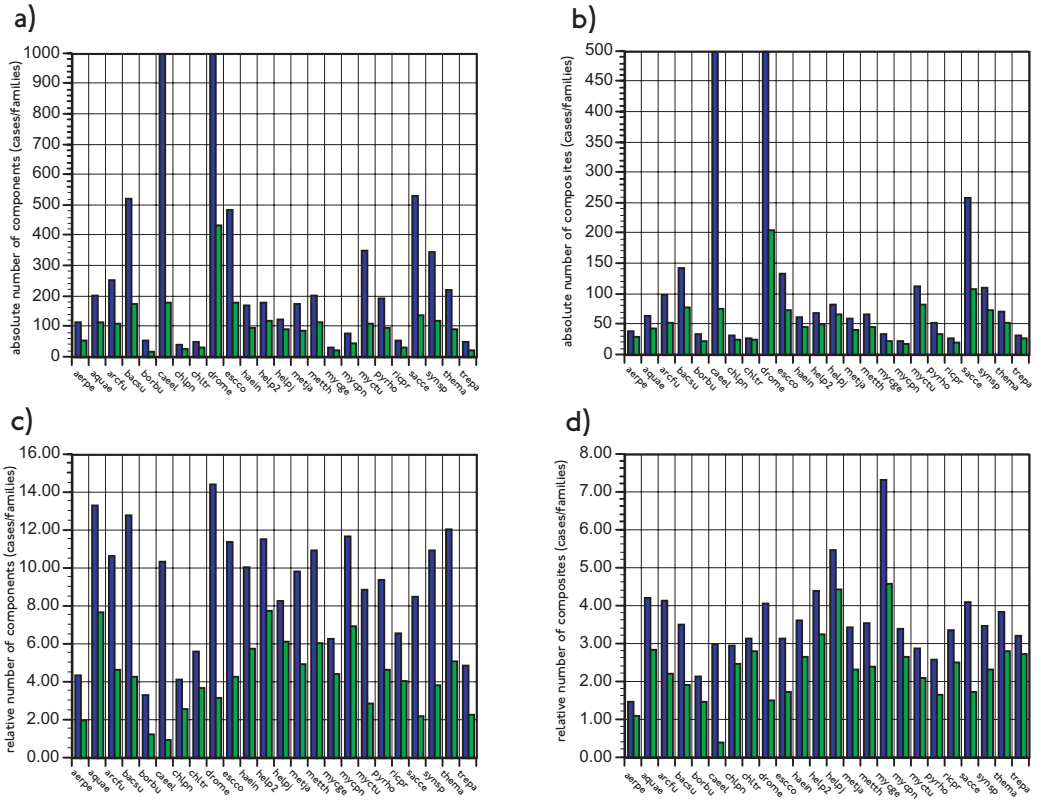


Figure 4.13: Absolute and relative numbers of component and composite proteins: Absolute number of (a) component and (b) composite proteins as individual cases (blue bars) and protein families (green bars), by species. Data for *C. elegans* and *D. melanogaster* are clipped (1,973 and 1,981 components, 567 and 559 composites, respectively). Relative numbers of (c) component and (d) composites per species, as individual cases (blue bars) and protein families (green bars), normalised by total genome size (number of ORFs). Average values per genome are 9% for components and 4% for composites. Species name abbreviations as in Table 4.2.

appear to be paralogy, genome size and phylogenetic distance. For instance, larger genomes have more composite, possibly paralogous, proteins. At the same time, closely related species evidently show similar patterns of gene fusion. The results below address each of these factors in turn and examine their relative contribution to the gene fusion process and their effects on the prediction of functional association of proteins. For every genome, both sets of component and composite proteins were subsequently clustered using GeneRAGE, to detect the degree of paralogy for these proteins (Figure 4.13, green bars). There are 2,534 component families, with an average of 105 families per genome (Figure 4.13a, green bars) and 1,323 composite families, with an average of 55 families per genome (Figure 4.13b, green bars). Comparing these numbers with the number of unique cases, it is evident that there is a paralogy level of two to threefold per genome for the composite and component proteins, respectively. As mentioned above, this effect contributes to the confidence of the predictions, depending on whether paralogy is observed in the query or the reference genome.

Another characteristic of this process is the redundancy of both sets of component and composite cases: the number of instances of these may be high but they are widely present across species, falling into well-defined protein families. When all components and composites are clustered as a single set (as opposed to within species, above), sequence clustering results in 1,287 single component families and 621 single composite families (as represented in the current analysis for the 24 species). Comparing these numbers with the number of families per species, it is apparent that there is a further two-fold reduction for both sets. This result indicates that gene fusion is widespread in evolution but forms a finite set. Different species may contain a common core of composite families, but also provide new families that are used to predict functional association. For instance, *D. melanogaster* provides far more composite families (more than 200) compared to *C. elegans* (fewer than 100) (Figure 3b, green bars). Genomes with unique composite families, such as *D. melanogaster*, contribute strongly to the majority of predicted interactions. It may also be that only certain classes of proteins are involved in gene fusion and that there is an upper limit for the predictive power of this approach obtainable from these 621 (currently available) families. Evidently, the number of component and composite proteins detected in each species is also dependent on genome size (Figure 4.13). When the above numbers for unique cases and families of components (Figure 4.13c) and composites (Figure 4.13d) are

normalised by the number of open reading frames (ORFs) for the species examined, the patterns of distribution are significantly altered. For instance, *Aquifex aeolicus* and *Thermotoga maritima* appear to have a large number of components involved in gene fusion (more than 12% of their genes are involved in this process) (Figure 4.13c), whereas the absolute numbers are low (Figure 4.13a). This is also the case for composites, where, for example, *S. cerevisiae* yields as many cases as *D. melanogaster* in relative terms (4% of the genome) (Figure 4.13d), while the absolute counts are dramatically different (Figure 4.13b). Finally, when the factors of paralogy and genome size are removed by sequence clustering and normalisation, respectively, the effect of phylogenetic distance between species can be detected. A distance measure based on the sharing of composite families between genomes has been devised and was used to identify relationships between the 24 species examined as follows:

- All composite proteins were clustered into 621 families and a distance measure was derived according to the sharing of clusters between the 24 species examined.
- This pairwise distance measure is calculated as:

$$\delta = (1 - S_{A,B}/T_{A,B}) \times 100$$

  where $S_{A,B}$ is the number of shared composite clusters and $T_{A,B}$ is the average of the composite cluster counts from the two species. This measure is reminiscent of a recent genome-wide orthologue analysis (Snel et al., 1999).
- This measure was used to calculate a distance matrix representing the distance between every pair of genomes when measured in terms of fusion events shared.
- The PHYLIP package (Retief, 2000) was used to construct an unrooted nearest neighbour dendrogram for the 24 species. Bootstrap values were generated using the *consensus* program and a 'delete-half' jack-knife procedure.

The tree produced by this analysis is shown in Figure 4.14. The fact that the tree based on this distance measure does not significantly contradict

other trees based on sequence alignments is a strong indication that our hypotheses about the factors involved in gene fusion are valid. This result also indicates that certain types of fusion events appear to be confined to specific phylogenetic groups, such as the Archaea, various bacterial clades and the Eukarya (Figure 4.14).

4.3.3 Data Storage & Availability

All results of this present analysis are stored in a MySQL database, which is accessible over the world wide web through Perl CGI scripts. This website[3], called *AllFuse*, contains all predicted gene fusion events, and their associated alignments. This resource is easily searchable, and all data are available from the site. A screenshot of the main page and an example fusion alignment are shown in Figures 4.15 & 4.16. A number of research groups have already started the use these data for comparison to other experimental and computational techniques.

4.3.4 Discussion

The exhaustive detection of gene fusion events in entire genome sequences allows the prediction of functionally associated components based merely on genome structure. The all-against-all species comparison is a necessary step because we have repeatedly observed fused, composite proteins in taxonomically lower organisms. The landscape of gene fusions appears to be a complex one, affected by paralogy, genome size and phylogenetic distance.

Although gene fusion is widely present across various phylogenetic groups, it is a process that may involve only certain types of proteins (Tsoka and Ouzounis, 2000a). Yet, this approach for the prediction of functional associations of proteins results in robust predictions for physical interactions, pathway involvement, complex formation and other types of functional associations of protein molecules.

With the present analysis, we delineate the available universe of fusion events and detect a set of 621 composite protein families from which predictions may be obtained. This approach results in 39,730 pairs of functionally associated proteins across 24 species, with high precision and coverage. This novel set of predictions is made available to the scientific community for

[3]http://maine.ebi.ac.uk:8000/allfuse/

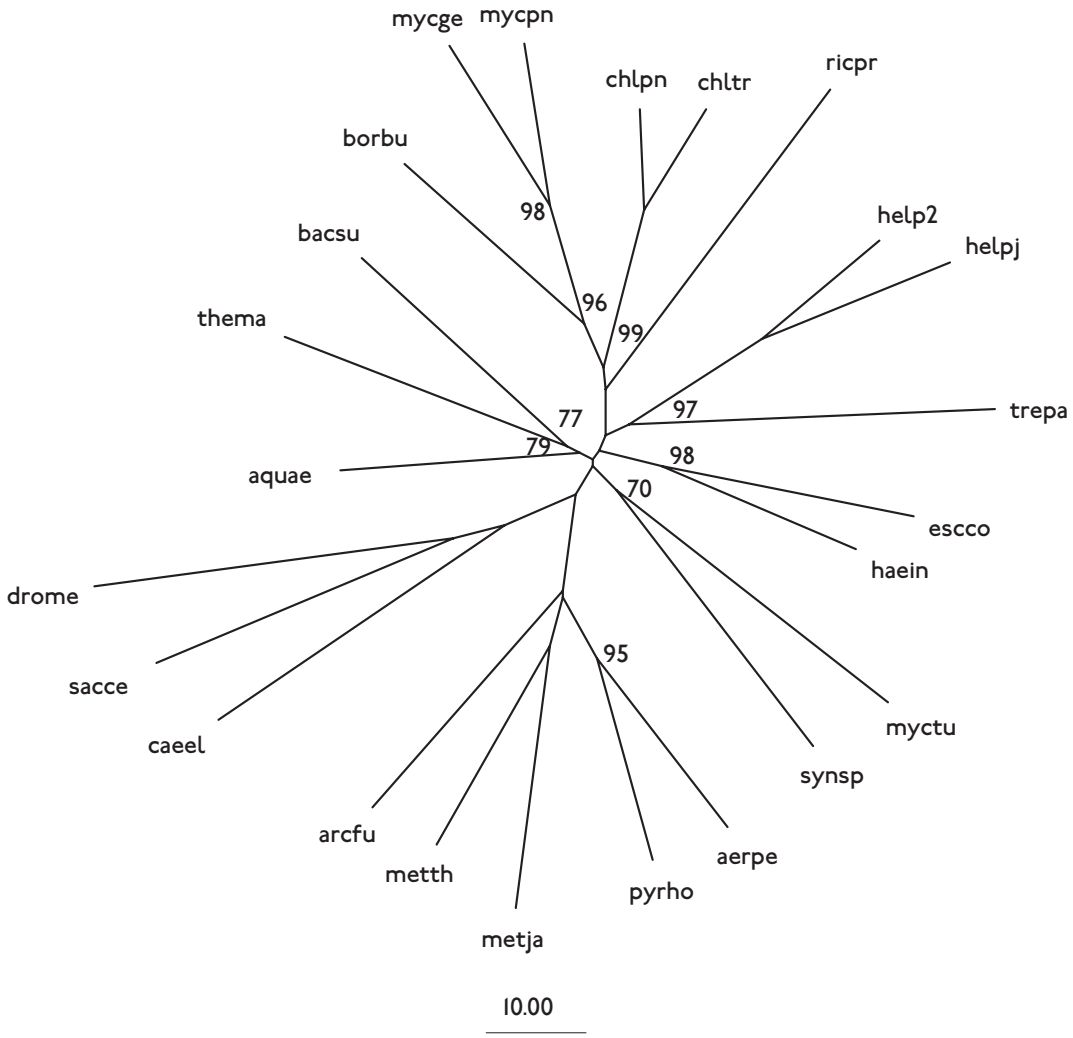


Figure 4.14: Neighbour-joining dendrogram representing the phylogenetic proximity of each of the 24 species in terms of detected gene fusion events. The distance measure is derived from the count of composite families. Scale bar is set to indicate a distance of 10 (ranging from 0 to 100). Species name abbreviations as in Table 4.2. Only bootstrap values less than 100 are shown.

the first time, and we believe that many of these cases can be subsequently verified by experimental methods.

This research together with related research (Marcotte et al., 1999b) has generated an enormous amount of interest in genome context approaches to the prediction of protein function and protein interaction (Sali, 1999; Huynen et al., 2000; Eisenberg et al., 2000). A number of separate research groups are now using very similar approaches to predict functional associations between proteins and explore the underlying evolutionary events (Yanai et al., 2001; Mellor et al., 2002).

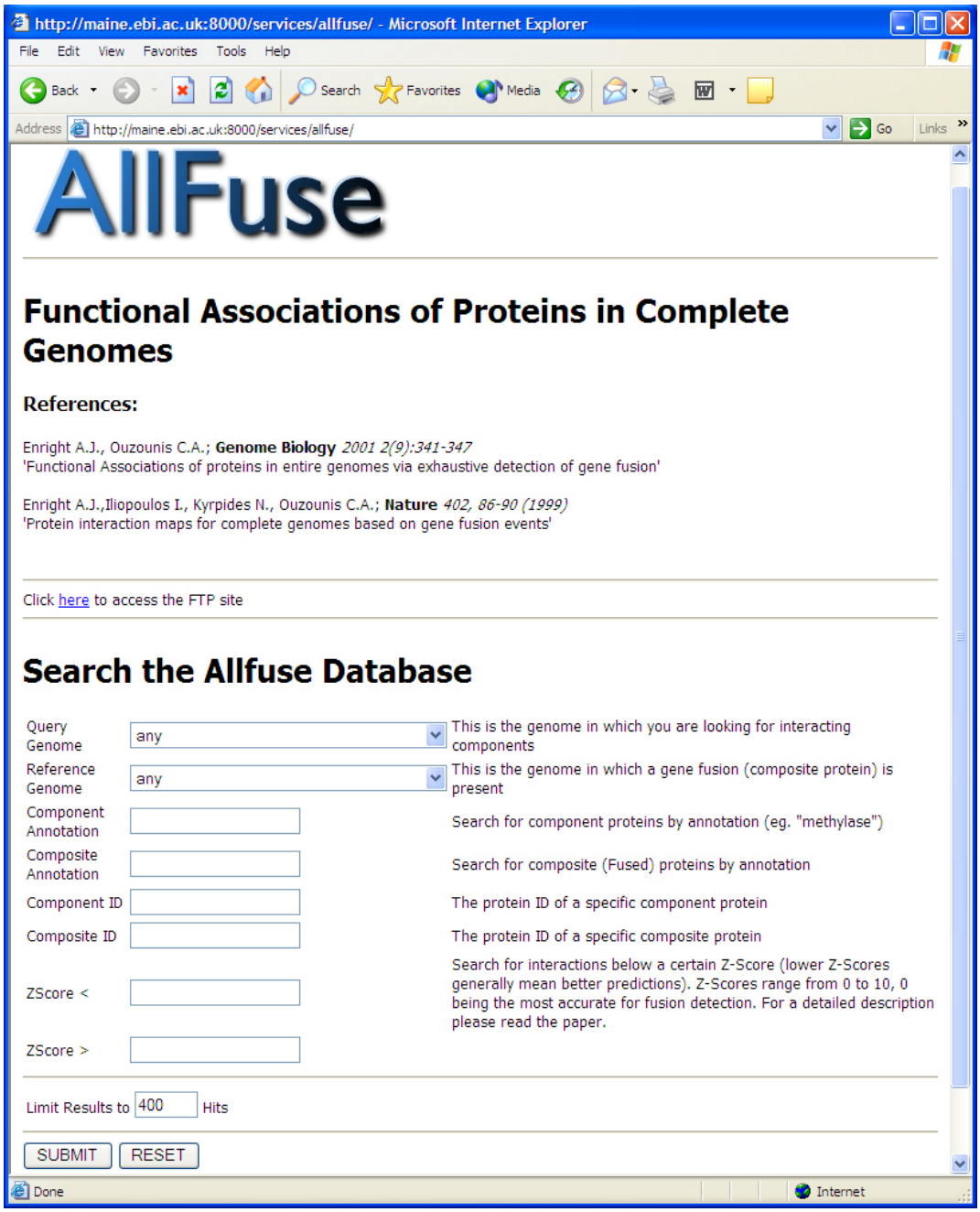


Figure 4.15: Screenshot of the AllFuse main page.

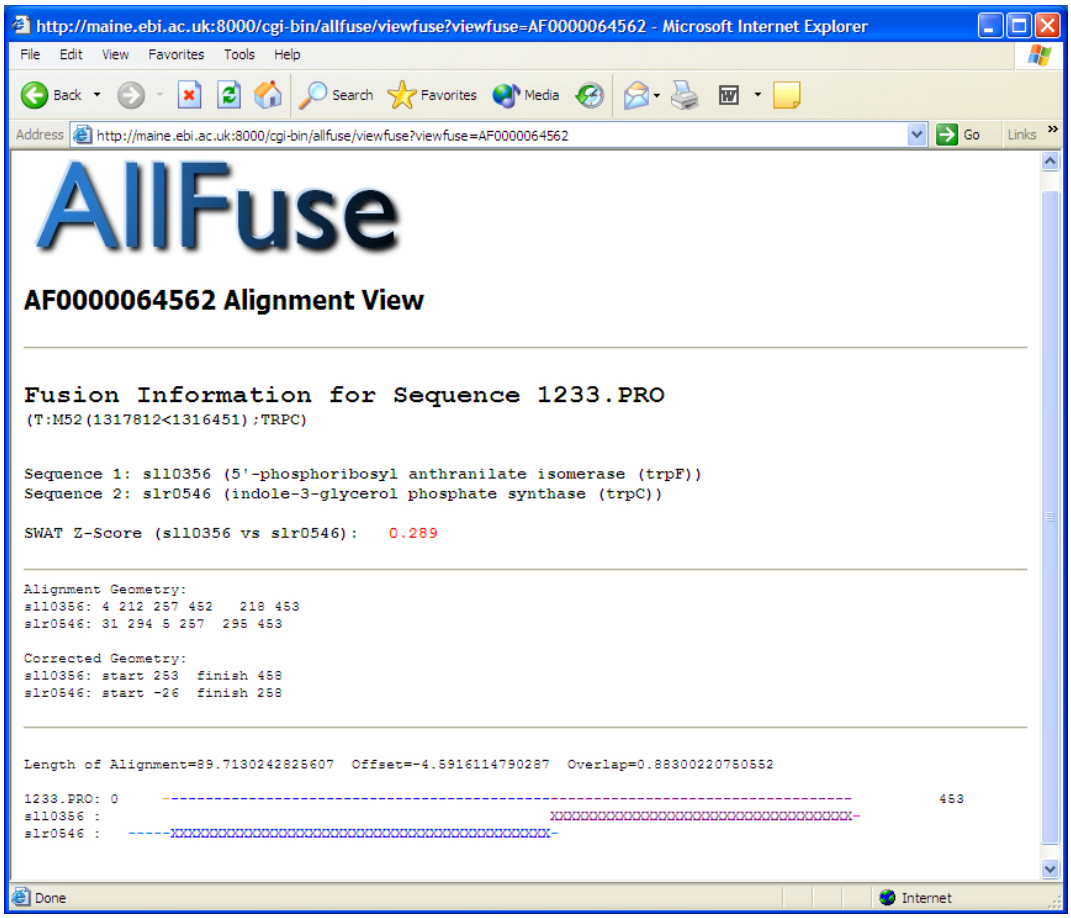


Figure 4.16: Example fusion alignment from the AllFuse web server.

Chapter 5

Methods for Genome Analysis and Annotation

The research detailed in this thesis involves the computational comparison of complete genomes for the prediction of protein function and protein-protein interaction. This research required the development of many novel sequence analysis tools and methods. Some of which represent interesting research projects in their own right.

A selection of novel algorithms and tools which have been used as the backbone for most research described in this thesis will be presented in this chapter. These methods are diverse, yet explore fundamental issues within bioinformatics research such as visualisation of data, high-performance computing and fundamental sequence analysis issues. Some of these methods (such as document clustering of Medline abstracts) are rapidly becoming separate fields of research within bioinformatics and computational biology.

5.1 CAST: Detection of Compositionally Biased Regions in Protein Sequences

5.1.1 Introduction

Probably the most commonly used bioinformatics tools are those which build local and global sequence alignments between protein sequences. An introduction to these methods is given in Chapter 1. One such tool in common use is BLAST (Altschul et al., 1997). These methods allow one to determine the degree of similarity between any two peptide sequences according to a scoring function. These sensitive algorithms are commonly used to detect the function of a protein by comparing it to a database of proteins whose functions are known. In general these methods use dynamic programming techniques (Smith and Waterman, 1981) to detect the highest scoring alignment between two sequences. Many sequences however, contain regions of low-complexity (Wootton, 1994). These regions tend to contain biases towards certain residue types. Two proteins which are functionally unrelated, may contain regions of bias towards the same amino acid, and hence will obtain a high-scoring alignment for these regions with a sequence similarity search tool.

These low-complexity regions need to be filtered out of protein sequences so that accurate similarity scores can be generated when comparing these sequences. Algorithms have been developed to filter peptide sequences for low-complexity regions. Typically these algorithms find regions of compositional bias and filter the region by replacing the biased amino acids with the 'X' character which is ignored by most sequence similarity search tools when an alignment score is being generated. This 'X' character is commonly used to denote an amino acid of undefined type.

Two such methods are XNU and SEG. The first method XNU identifies amino acid repeats on the basis of a self comparison of the query sequence (Claverie and States, 1993), while SEG (Wootton and Federhen, 1993) detects low-complexity regions based on an information-content measure. Both methods replace biased or low-complexity regions with 'X' characters. These methods focus on a specific region or window of a peptide sequence and filter iteratively across the peptide sequence removing regions which are deemed to be of low-complexity. While these methods are effective at detecting and removing regions of low-complexity with high efficiency, they are not without

flaw. Due to the implementation of these algorithms it is possible that the region filtered out of the input sequence may contain both low-complexity amino acids, and also possibly amino acids representing a functional motif that occurs close to or within a low-complexity region.

For this reason it was decided to develop a novel algorithm for the filtering of low-complexity amino acids in protein sequences, which was sensitive yet highly specific. Such an algorithm should accurately detect low-complexity regions and filter out only these regions, without affecting the rest of the amino acid sequence. One method of doing this is to examine the query sequence for bias of each amino acid type separately. This idea was originally developed to a certain extent for use in the GeneQuiz project (Scharf et al., 1994).

Because sequence similarity searching is at the core of much of the research described in this thesis, it was decided to redevelop and reimplement the original algorithm (BIASDB) as a general algorithm written in C for the complexity analysis and filtering of peptide sequences. This accurate and sensitive filtering algorithm is used for practically every similarity analysis described in this thesis.

5.1.2 Algorithm Concept

In contrast to previous approaches, low-complexity regions in protein sequences shall be defined as those regions that score highly in homology searches with degenerate sequences composed of a single amino acid type (i.e. homopolypeptides). A similar definition has been used previously (Robison et al., 1994). This description is a sensible one, as even the most degenerate sequences can achieve high-scores in database searches. This definition of low-complexity sequences can be reformulated into a general method for detecting low-complexity regions. If one compares a homopolymeric peptide sequence of a specific length and residue type, against another query sequence and a high scoring similarity is detected, then this region may be compositionally biased. These concepts have been implemented in the CAST algorithm for the general detection and filtering of compositionally biased residues in protein sequences.

If one supposes that the fractions of different residue types a and b in a search and test sequence respectively are statistically unrelated, then it follows that the probability of finding a match of residue types a and b can

be calculated from the independent residue frequencies as:

$$P_{ab} = f_a P_b$$

where f_a, P_b are the fractions of amino acid types a and b in the search and test sequences respectively. If one scores all the possible a-b matches with a comparison matrix M composed of elements $M_{a,b}$, the average expected score over a region of length l will be:

$$l \sum_{a,b} (P_{ab} M_{ab}) = l \sum_{a,b} (f_a P_b M_{a,b})$$

High scores reflect similar sequence patterns. As we admit any local region of length l in both of the proteins, we can ignore the factor l. Consequently, the frequencies f_a and P_b correspond to the local residue frequencies in the two compared regions of the search protein. The residue composition is invariant and the sum score over all residue types can be performed. The only remaining variable is the composition of the search sequence f_{ab} and can be written as:

$$\sum_{a,b} (f_a P_b M_{a,b}) = \sum_a f_a \sum_b (P_b M_{a,b}) = \sum_a (f_a C_a)$$

where C_a is a parameter related only to the residue type a in the search sequence.

The residue frequencies f_a naturally lie between the values 0 and 1, and sum to 1. Therefore the last part of the previous equation is an interpolation between the 20 possible values C_a and can only result in scores between the smallest and largest value of C_a. The maximum sum that can be obtained is the case where the sum is equal to the largest C_a, arbitrarily sorted as C_1. The maximum score will always be obtained when the corresponding residue frequency f_1 equals 1, which corresponds to the homopolymer. Therefore one of the 20 homopolymers will always have the highest score obtainable by any unrelated sequence.

This argument does not apply if the two complex sequences share more than just a similar composition. In these cases, the complex sequence can have much higher scores that reflect real sequence similarity. However, there is a well-developed statistical theory that allows the estimation of the likelihood that such similarities have arise by chance (Karlin and Altschul, 1990), these statistics are described in Chapter 1.

5.1.3 Detection of Low-Complexity Residues

Using the ideas above, one can define a method for detection of compositionally biased residues, as comparing a query sequence against a database of 20 degenerate protein sequences of arbitrary length. Each of these 20 sequences being a homopolymer of a different natural amino acid. Any homologue detected by such a search will allow one to identify both the amino acid type and the exact region of low-complexity sequence in the query sequence.

Because we are searching a query sequence against an artificial *bias-database* of 20 homopolymeric sequences, we can simplify the problem of similarity detection. In this case the Smith-Waterman algorithm (Smith and Waterman, 1981) can be optimised because the position in the homopolymeric sequence is irrelevant. A single pass over this sequence is all that is required for any given residue type, to find high-scoring regions in the query sequence.

If one takes a biological peptide sequence r of length n:

$$r_1, r_2, r_3, \ldots, r_i, \ldots, r_n$$

A score at position $i(1 \leq i \leq n)$ is derived by adding the value counted at the previous position and the score given in a mutation matrix for a match of residue type a with the amino acid type r in the sequence at position i.

$$s_i^\alpha = M_{a,r_i} \begin{cases} s_{i-1,}^\alpha & s_{i-1}^\alpha \geq 0 \\ 0, & s_{i-1}^\alpha < 0 \end{cases}$$

Areas of maximal score can be identified as stretches in the resulting run of values that have positive scores and range from the first positive score to its maximum. Applying the algorithm for all 20 possible residue types a, allows detection of each individual type of bias. The score at the maximum allows one to quantify the bias detected with a bias score. Hence, it is possible with this algorithm to identify a region of bias, the type of bias and a score for the biased region.

In order to reduce complexity, gaps are not allowed with this implementation. Effectively this algorithm is equivalent to 20 passes of Smith-Waterman for each query sequence against the homopolymers of equal length with infinite gap open and extension penalties.

5.1.4 Filtering Procedure

One of the most important features of this algorithm is that filtering is applied iteratively to detected regions of amino acid bias. Instead of replacing each amino acid in the biased region of the query sequence with 'X' characters, only the amino acids of the detected bias type are replaced. Other biased amino acids of different types in this region, may however be later replaced with 'X' characters in subsequent passes of the Smith-Waterman algorithm. Amino acids that are not detected as biased by any of the 20 passes will be left intact in the query sequence. This is advantageous as functional or structural motifs, within or near a biased region will be preserved. This iterative detection and filtering procedure is the key advance represented by the CAST algorithm.

5.1.5 Implementation

The algorithm has been implemented as described above and in Figure 5.1. An input sequence is iteratively scanned against a homopolymeric sequence for each of the 20 amino acid types. If any of these comparisons detects a similarity score above a threshold value, then amino acids of that type are replaced with 'X' characters in that region of the input sequence. This process continues until no pass detects a significant similarity to a homopolymeric sequence.

Previously some of these ideas had been implemented as a simple Perl program for use within the GeneQuiz project (Scharf et al., 1994). In order to improve the performance and extend the algorithm, it was decided to rewrite the algorithm using the C programming language. By default the algorithm uses a 40 half-bit threshold and a variant of the BLOSUM62 scoring matrix, for the detection and scoring of biased regions. We calculate the scores for 'X' amino acids as the mean value of the similarity scores in each row or column as proposed by Altschul (Altschul et al., 1994). This eliminates the effect of the *neutral* masking character in subsequent iterative rounds of bias detection. The threshold and also the mutation matrix may be modified by the user if required.

Usage of the program is straightforward, one supplies a query sequence, and the CAST algorithm returns the query sequence filtered with 'X' characters and optionally, a list of biased regions sorted by residue. Implementing the algorithm in C increased the performance of the algorithm substantially,

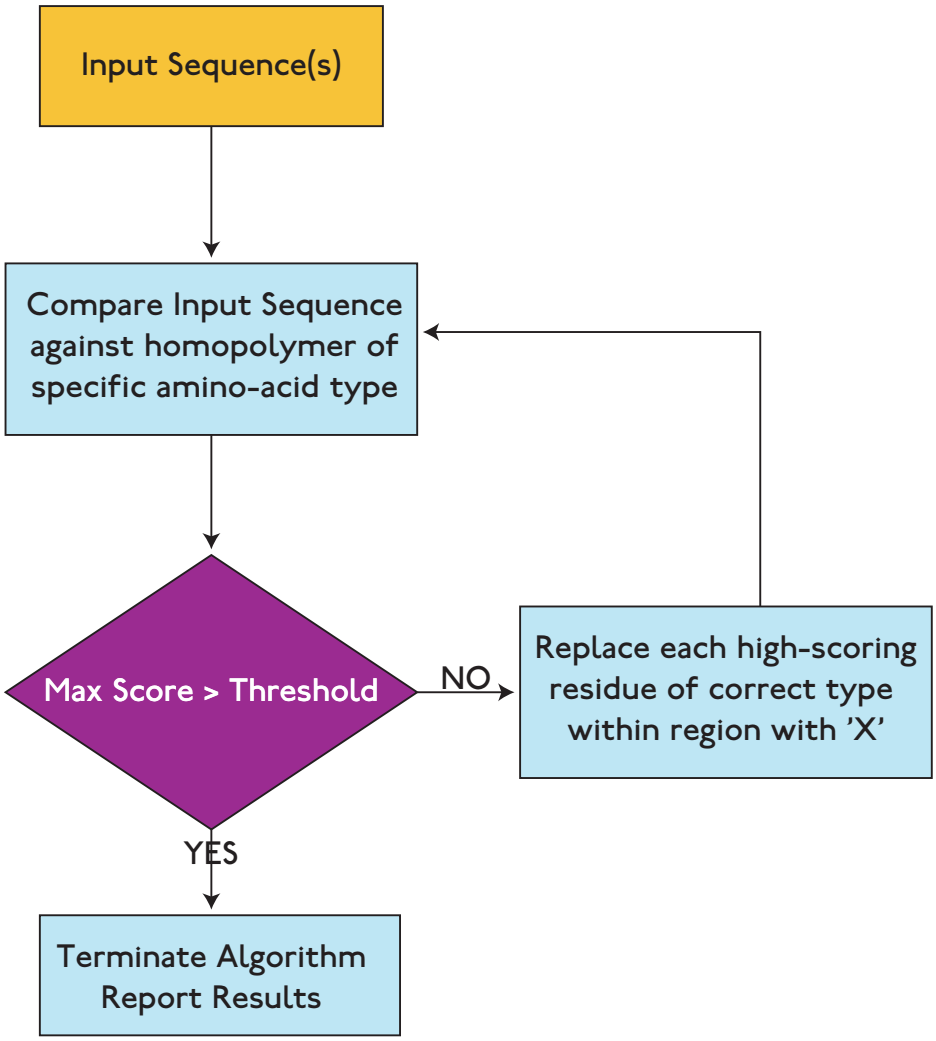


Figure 5.1: Flowchart of the CAST algorithm.

as the language is ideally suited to this type of iterative processing.

In addition to the basic CAST program, a web-based service has also been developed[1]. This is shown in Figure 5.2 and allows users to submit sequences across the internet. Each request is queued on the server, and when complete, returns the filtered sequence and statistics for the filtering process.

The CAST algorithm has been extensively tested with genomic protein sequence data and its performance has been shown to be superior to that of the previously mentioned algorithms (SEG & XNU) (Promponas et al., 2000).

[1] http://maine.ebi.ac.uk:8000/services/cast

Figure 5.2: A screenshot of the CAST web-server.

5.2 BioLayout: A Visualisation Algorithm for Protein Sequence Similarities

Accurate visualisation of complex data is important in many fields. Bioinformatics has become a data-intensive field and many bioinformatics research projects routinely generate vast quantities of data. Many of the analyses described in this thesis involve the use of protein similarity information such as that obtained using the BLAST (Altschul et al., 1997) similarity search tool. Aside from merely listing protein similarities as a text or HTML document, no tool exists for simple and accurate graphical visualisation of such similarity information. Ideally such a visualisation tool would allow one to quickly identify salient features within a set of similarity results. Such features would include: related families of proteins, multi-domain protein relationships, outlier and protein fragments. One possible visualisation is as a simple graph where nodes represent proteins and edges represent similarities. While these graphs are simple to construct, no data usually exists for the coordinates of each node within the graph. Assigning coordinates to each node is essential in order to produce a visually appealing graph. This problem is known as the *graph layout problem* and has been extensively studied in the fields of mathematics and computer science. We sought to implement some of these ideas for use in the field of bioinformatics visualisation and graph layout (Enright and Ouzounis, 2001a).

5.2.1 Graph Layout Theory

The graph layout problem for simple undirected graphs can be formulated as follows (Fruchterman and Rheingold, 1991):

> A graph $G = (V, E)$ consists of a set V of vertices (nodes) and a set E of edges. Each edge joins a pair of vertices. For an undirected graph with straight edges an ideal graph layout should conform to the following criteria.

- An even distribution of vertices within the graph space.
- A minimal number of edge crossings.
- Uniform edge lengths.

- Inherent symmetry with the initial data should be conserved.
- The graph should conform to the boundaries of the graph space.

These criteria are specified in order to efficiently produce an aesthetically pleasing graph layout. Such a graph should be easier to interpret than graph layouts that do not satisfy the above criteria.

5.2.2 Graph Layout Algorithms

One of the first methods for the layout of simple undirected graphs is called *force directed placement* (Eades, 1984). This method introduces the concept of mathematically modelling a graph as a physical system of rings and springs in order to generate a graph layout. If one replaces the vertices of a graph with steel rings, and the edges with springs that connect these rings, then model this as a physical system of mechanical forces, one will obtain a graph layout. This is because the physical system will strive to assume a minimum energy state which we can consider a graph layout. In other words vertices that share many connections (springs) will be placed closer together in order to reduce the total energy of the system. While in principle these algorithms are simple, they become computationally intensive for large graphs. The time complexity of such algorithms (commonly called *n-body* algorithms) is $\Theta(E^2 + V^2)$. This means that, for any iteration of such an algorithm, forces must be calculated for every node against every other node, and similarly for each edge (vertex).

However modelling a physical system to produce a graph layout does not require one to model each force with physical accuracy. Application of physical forces in an unrealistic manner should also produce a reasonable graph layout (Fruchterman and Rheingold, 1991). Another approach called the *spring embedder* algorithm (Kamada and Kawai, 1989) deviates from these specific laws of physics and Newtonian mechanics in two important ways. Firstly, the use of Hooke's Law for calculating the forces on each spring, is abandoned in favour of a more simple force. Secondly, forces of attraction between nodes are only calculated for neighbouring nodes, and not for distant inter-node interactions. While this method deviates from physical reality, the graph layout produced is almost identical to that of

force directed placement yet the time-complexity is reduced substantially to $\Theta(E + V^2)$.

This idea of applying unrealistic forces in an unrealistic manner for graph layout was extended further by Fruchterman and Rheingold (Fruchterman and Rheingold, 1991). Their system resembles particle physics, where nodes represent protons and neutrons. In this system nodes exert an attractive force on each other (similar to the strong nuclear force). However, once nodes become too close to each other, this attractive force becomes repulsive. These forces will hence separate nodes at an optimal distance that balances repulsion and attraction.

The Fruchterman & Rheingold algorithm behaves as follows (Figure 5.3):

- Vertices in the graph repel other vertices within a radius r with a distance d dependent force of repulsion $f_r(d)$.
- Vertices connected by an edge are attracted to each other with a distance d dependent force of attraction $f_a(d)$.
- The optimum distance k between two vertices is a result of both attractive and repulsive forces.

The optimum distance k between any two nodes is calculated as follows:

$$k = C\sqrt{\frac{\text{total graph area}}{\text{total vertices}}}$$

The attractive and repulsive forces at distance d are hence:

$$f_a(d) = d^2/k$$
$$f_r(d) = -k^2/d$$

5.2.3 The BioLayout Algorithm

The BioLayout algorithm extends the approach of Fruchterman & Rheingold to the problem of sequences similarity graph layout (Enright and Ouzounis, 2001a). Vertices in such a graph represent proteins while edges represent detected similarities between these proteins. Proteins have different degrees of similarity, so each edge is weighted according to a similarity score. The

weighting scheme chosen for BioLayout is $W = -\log_{10}(E)$, where E is an Expectation value for protein similarity obtained from BLAST (Enright and Ouzounis, 2001a). In order to place similar proteins closer together, we modify the Fruchterman & Rheingold algorithm to use these weights by automatically adjusting the optimum distance k between any two nodes according to their degree of similarity.

The attractive and repulsive forces, taking into account similarity weights W, for a pair of nodes at distance d are now calculated as follows:

$$f_a(d) = Wd^2/k$$
$$f_r(d) = -(\frac{k}{W})^2/d$$

This new weighting scheme should place nodes representing proteins with high degrees of similarity close together in the resulting graph layout. The modified algorithm is described in pseudocode in Figure 5.3. A graph G representing all nodes (proteins) and edges (similarities) is created, with each node in a random position in a layout area (or frame) of size $W \times L$. A finite number of iterations (typically 60) is chosen for the layout algorithm. At each iteration, attractive forces are calculated between all nodes connected by an edge according to $f_a(d)$. Repulsive forces are then calculated between all pairs of nodes closer than a set distance r according to $f_r(d)$. A displacement vector is calculated for each node based on observed attractive and repulsive forces on that node. The displacement is limited by a cooling function to a maximum value t called *temperature*. Displacement is also limited where a node would leave the graph space (frame). Each node is then moved to a new position according to its displacement vector, and the algorithm continues its next iteration.

The cooling function is used to generate more optimal layouts (Fruchterman and Rheingold, 1991). The basic idea of cooling is to allow large movement of nodes in the initial iterations of the algorithm, as nodes strive to leave their random starting positions. Lowering the temperature gradually refines the graph, as nodes assume more optimal positions within the graph. The maximum displacement of each node is controlled by the t parameter as previously described. The cooling function $f(c)$ is applied at each iteration to modulate the temperature through each iteration. When the algorithm is reaching completion, the temperature is such that only small adjustments are allowed in the layout.

Protein 1	Protein 2	E-Value
LEVR_BACSU	ATOC_ECOLI	5×10^{-17}
AFQ1_STRCO	ATOC_ECOLI	1×10^{-11}
ALGB_PSEAE	ATOC_ECOLI	1×10^{-83}
BASR_ECOLI	ATOC_ECOLI	4×10^{-11}
BASR_SALTY	ATOC_ECOLI	7×10^{-13}
YGIX_ECOLI	ATOC_ECOLI	5×10^{-13}
YGIX_HAEIN	ATOC_ECOLI	1×10^{-11}
CHVI_AGRTU	ATOC_ECOLI	7×10^{-11}
CHVI_RHIME	ATOC_ECOLI	1×10^{-11}
CIAR_STRPN	ATOC_ECOLI	9×10^{-11}

Table 5.1: BioLayout Input Format

Work carried out previously (Fruchterman and Rheingold, 1991) has shown that the best performing cooling functions start with a linear decrease in temperature, then assume a constant low temperature phase for final adjustment. The BioLayout cooling function also performs in this manner.

5.2.4 Implementation and Usage

The layout algorithm has been implemented as described (Enright and Ouzounis, 2001a). The input for the algorithm is a set of sequence similarities obtained from an algorithm such as BLAST. The input format is shown in Table 5.1. These similarities and associated similarity scores are read and transformed into an internal graph structure. The layout algorithm then produces a graph layout through 60 iterations. A graphical user interface to this engine has been written in C using the Sun Microsystems OpenWindows toolkit[2]. This interface is shown in Figure 5.4. The interface allows the user to select and search for proteins or sets of proteins in the graph, move nodes, label nodes and save the final graph layout as a postscript file (an example is shown in Figure 5.5). In addition, the user can select groups of proteins from the graph, and request information about these proteins (based on their identifiers) automatically using the 'Sequence Retrieval System' (SRS) (Etzold and Argos, 1993) server at the European Bioinformatics Institute[3]. A non-graphical C implementation is built from the same source

[2]http://www.sun.com/
[3]http://srs.ebi.ac.uk/

```
begin for i := 1 to iterations do
        {calculate repulsive forces}
        for u in V do
            {each vertex has two vectors: .pos and .disp}
            v.disp := 0;
            if (u ≠ v) then
              {Δ is short hand for the difference}
              {vector between the positions of the two vertices}
              Δ := v.pos − u.pos;
              if (Δ ≤ r) then
                {nodes only repel if closer than distance r}
                v.disp := v.disp + (Δ/|Δ|) * f_r(|Δ|);
              end
            end
        end
        {calculate attractive forces}
        for e in E do
            {each edge is an ordered pair of vertices .v and .u}
            Δ := e.v.pos − e.u.pos;
            e.v.disp := e.v.disp − (Δ/|Δ|) * f_a(|Δ|);
            e.u.disp := e.u.disp + (Δ/|Δ|) * f_a(|Δ|)
        end
        {limit the maximum displacement to the temperature t}
        {and then prevent from being displaced outside of the frame}
        for u in V do
            v.pos := v.pos + (v.disp/|v.disp|) * min(v.disp, t);
            v.pos.x := min(W/2, max(−W/2, v.pos.x));
            v.pos.y := min(L/2, max(−L/2, v.pos.y));
        end
        {reduce the temperature as the layout approaches a good configuration}
        t := cool(t)
  end
```

Figure 5.3: BioLayout Pseudocode.

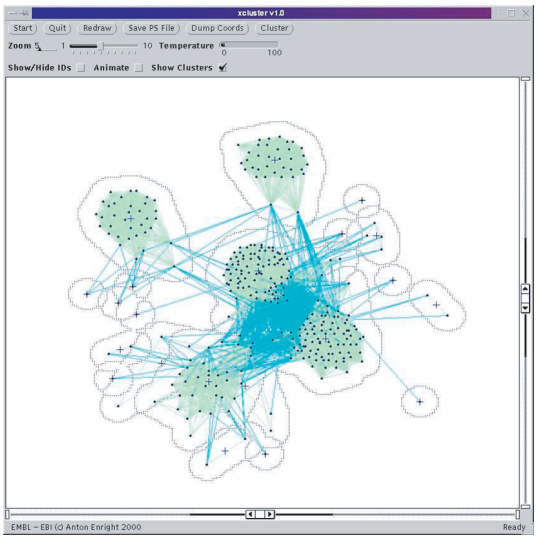


Figure 5.4: The BioLayout graphical user interface.

code and also produces postscript output files containing the graph layout produced.

This two-dimensional graph visualisation of proteins and similarities proves to be remarkably useful for sequence analysis. Interestingly, as the method is general, it can also be applied many types of similarity information. One such example is graph generation and layout for relationships between terms in Medline documents (Iliopoulos et al., 2001a).

It is very easy to visually interpret each graph for features such as protein families and multi-domain proteins. Closely related proteins in a family will form distinct clusters within the graph. Multi-Domain proteins tend to lie between large protein family groups, and outlier proteins or fragment peptides appear towards the edges of each graph. Figure 5.4 represents similarities between transcriptionally related proteins from different protein families. Although there is significant similarity between the separate families, it is very easy to visually determine that there are five main protein families within these similarities.

Another implementation for 3-Dimensional visualisation of highly complex datasets has also been written (Figure 5.6). The implementation described above is two-dimensional, but extending it to three dimensions is a trivial procedure, yet slightly more computationally intensive. The same C code is reused for the core engine of the algorithm, but a new 3D visualisa-

tion module was written using the Silicon Graphics 'Open Graphics Library' (OpenGL[4]). This visualisation proves to be an exceptionally useful way of examining highly complex datasets. The viewer interface allows one to interactively rotate, translate and zoom through highly complex representations of protein sequence space. An example graph layout is shown in Figure 5.6. In this case promiscuous protein family relationships within SwissProt are shown (see also Figure 2.12).

[4]http://www.sgi.com/software/opengl/

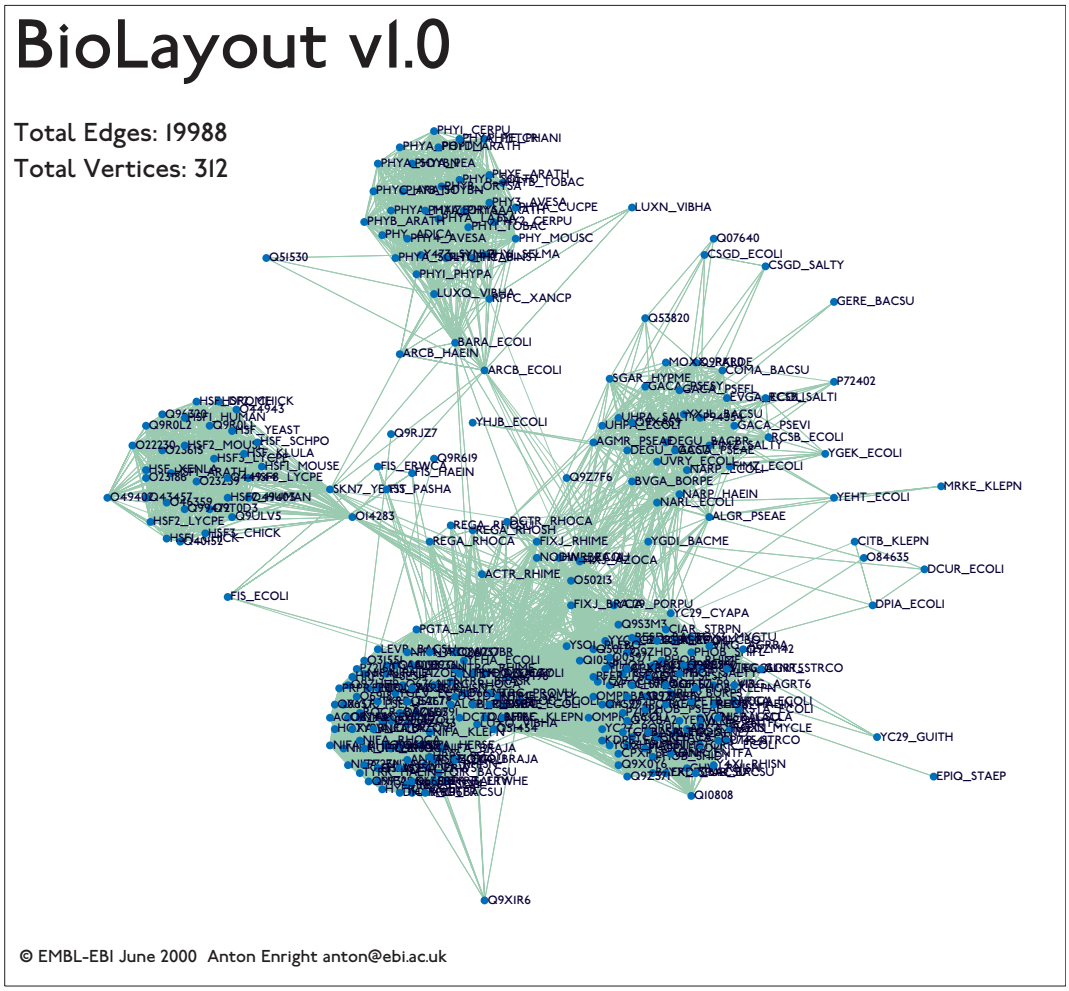


Figure 5.5: An example PostScript graph generated by the BioLayout algorithm. This graph shows five protein families involved in transcription, which are connected together due to similarity relationships arising from promiscuous and multi-domain proteins.

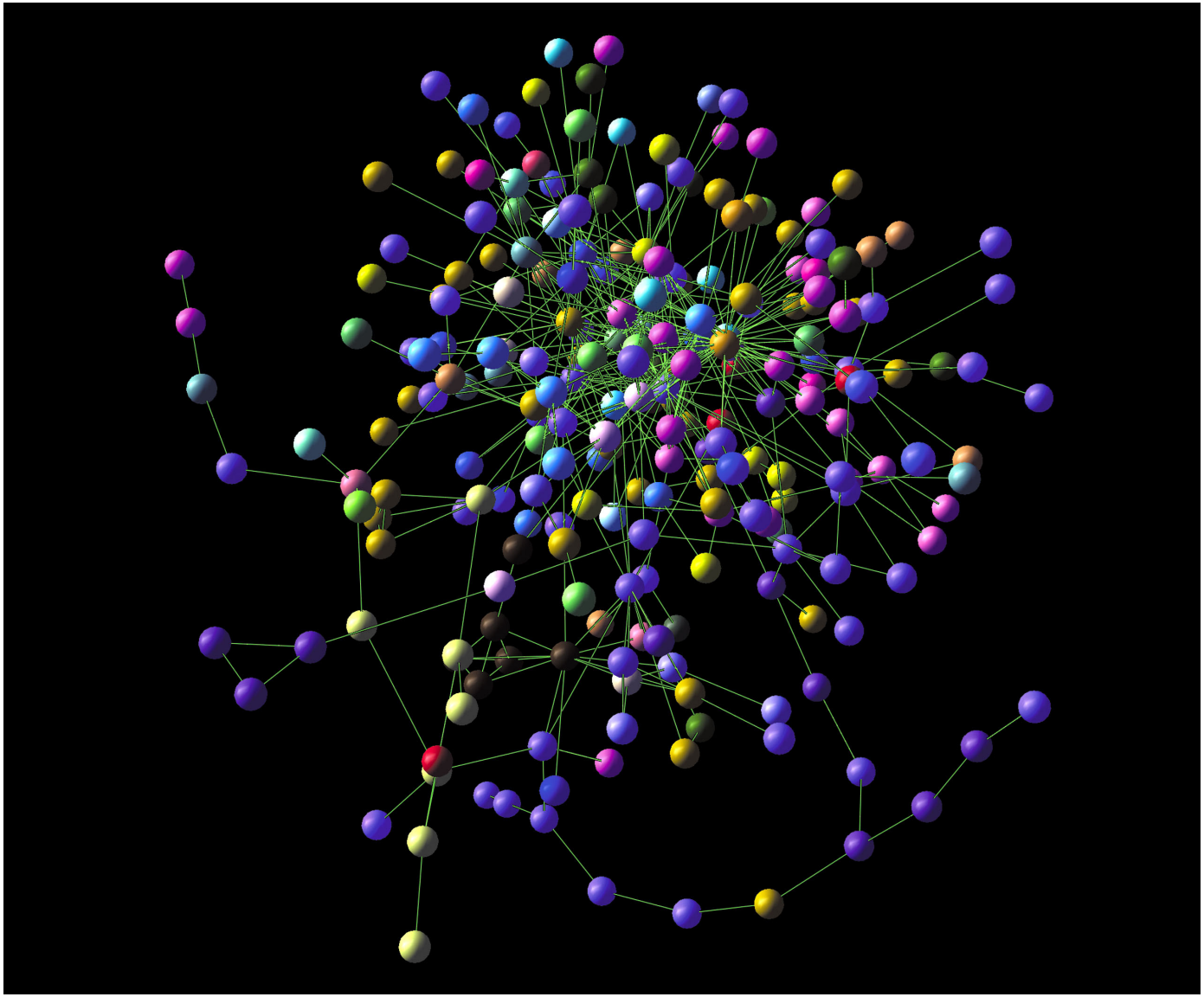


Figure 5.6: A three-dimensional graph layout generated using BioLayout3D. This is a 3-Dimensional layout of the graph shown in Figure 2.12, and shows promiscuous protein family relationships in the SwissProt database. The graph is colour coded according to the functional class of each family.

5.3 TextQuest: Automatic Classification of Medline Abstracts

5.3.1 Introduction

Much of the work we have undertaken involves the prediction of protein function from genome sequences. While we have been primarily concerned with functional prediction based on sequence analysis, there are other useful methods available. Many proteins and nucleotide sequences deposited in sequence databases are associated with an article published in a scientific journal. The U.S. National Library of Medicine (NLM), Medline database[5] stores information pertaining to over 16 million published articles. While Medline does not seek to explicitly store functional information for proteins, it does contain abstracts for each article. Careful analysis and data-mining of Medline abstracts has been shown to be a reliable method for the detection of terms and phrases which can indicate the functional role of a protein or gene. Text mining and more specifically *document clustering* can prove very useful for the automatic annotation of genome sequences. Abstracts are easily obtainable using the query based retrieval system of the Medline database (PubMed).

Used directly, Medline is of little use for large-scale textual analysis. The query system expects the user to already know the exact type of information that is required. Abstracts are generally searched on the basis of specific information such as keywords. In order for the user to obtain meaningful information, the user needs to have some understanding of the subject area, and select useful keywords (Salton, 1970).

Within the field of bioinformatics, there has been a move towards exploiting the vast accumulation of textual information in databases such as Medline. It was decided to implement concepts from the fields of text analysis in order to exploit this information. To this end a system called TextQuest was developed for text-mining of the Medline database (Iliopoulos et al., 2001a).

[5]http://www.nlm.nih.gov/pubmed/

5.3.2 Text-Mining Approaches

The most extreme form of textual analysis is *information extraction*. This form of analysis seeks through syntactic and semantic analysis to actually understand a document (Allen, 1994). Using statistical and algorithmic approaches a document is scanned, and the information required is simply extracted from the text. Such methods have been shown to work on a variety of different datasets. Biological free text is however very different to text extracted from public sources such as newspapers, and requires a completely different understanding of grammar and sentence structure. These methods are also very computationally intensive and generally require very complex software and database systems to be successful (Thomas et al., 2000).

An intermediate form of text analysis is that of *document clustering* (Willet, 1988). This statistical approach seeks to identify key terms in a document and group documents together based on their sharing of sets of these terms. This approach is much simpler than information extraction, as it requires no knowledge of syntax and grammar. For a given set of documents this method should return a set of document clusters which share a large number of their key terms. The mechanics involved in the clustering process are straightforward, but the accuracy of the method relies heavily in the statistical recognition of these key terms. Poorly selected terms will produce very poor document clusters.

Recently there has been growing interest in applying these techniques to biological document databases such as Medline (Andrade and Bork, 2000). These approaches have focused on two areas: extraction of relationships within text, such as protein interactions (Blaschke et al., 1999; Thomas et al., 2000), and keyword discovery (Andrade and Valencia, 1998; Fukuda et al., 1998). It was decided to tackle the problem of biological text analysis from a different angle using a document clustering approach. The only document clustering attempted for biological documents so far has been the NLM 'neighbors' utility (Wilbur and Yang, 1996). Given that this method relies heavily on the quality of biological term recognition, a new method was developed for the detection of significant terms in biological documents. Document clusters are then generated using an unsupervised machine learning approach based on the co-occurrence of these automatically detected terms.

5.3.3 The TextQuest Algorithm

The TextQuest algorithm (Iliopoulos et al., 2001a) is a well-defined ten step protocol. An input set of biological abstracts is taken from the Medline database, processed and restructured to obtain an optimal number of terms that will associate large numbers of biological documents into meaningful groups that will represent the biological roles of the genes/proteins described within.

The TextQuest protocol can be described as follows:

1. A set of K abstracts is extracted from Medline using query-based retrieval. This set is saved in Medline standard format, or alternatively in other formats such as XML.

2. The selected abstracts are parsed in order to extract only the 'UID' (Unique Medline Identifier) and 'AB' (Abstract Body) fields. Other information such as MeSH terms are not used, as they can frequently be misleading. The parsing process also removes any non-alphanumeric characters, which can produce misleading results.

3. Because the key terms will almost certainly be biological words such as 'kinase' or 'cell-cycle', common English words (such as 'the' and 'and' etc.) are removed. Many previous methods used a predefined *stop-list* containing words that were deemed non-informative (Voorhees, 1998) to filter these words from document text. Given the complex nature of biological text, it was not possible to generate such a stop-list. In order to automatically remove words that occur at high-frequency in the English language, without a stop-list, the TF.IDF (TF = Term Frequency, IDF = Inverse Document Frequency) metric is used (Rocchio, 1971). This is a well known system for weighting words in text based on their frequency in a given document and their frequency in a reference set. The reference set used is the British National Corpus (BNC) collection, which details the frequencies of English words in common usage. Terms that appear frequently in a document, but rarely in the reference set, are more likely to be specific to that document. Terms are scored using the following variant of the TF.IDF system: $w_i = \log_2(N_i/n_i)$. In this variant w_i is the weight of term i in the document, N_i is the frequency

of term i in the reference set L and n_i is the number of documents in L that the term i occurs in.

4. Terms with a high TF.IDF value (typically, greater than 15) are selected. These terms represent words occurring at relatively high frequency in a document, but rarely or never within the BNC reference set. This cut-off value $\rho = 15$ was picked after experiment and evaluation with multiple biological document datasets.

5. These detected terms from each document in the set K are combined. Frequencies of each term are calculated across all documents in the set K. Terms which occur frequently and specifically are retained. The cut-off threshold σ for this step is defined as: $\sigma = K/100$. For a set where $K = 1000$, this step will only keep terms which occur at least 10 times, even if their TF.IDF score from the previous step is high. At this stage common English terms and infrequent terms have been eliminated. The remaining set of terms U from this procedure we term the *go-list*.

6. A stemming algorithm written in Perl processes each term from the go-list U with a stemming procedure. This method is commonly used in the text analysis field, and seeks to reduce the complexity of terms by reducing them to a common stem. For example, words such as 'transcriptional' and 'transcriptionally' would be stemmed to 'transcription'. This procedure merges highly similar words in the go-list, and reduces the complexity of the subsequent analysis. The stemming algorithm used was not the common 'Porters Stemming Algorithm' (Porter, 1980), but a novel recursive method that we have developed specifically for document clustering. Porter's stemming algorithm uses a dictionary of word endings (such as '-ly','-ing' and '-ion' etc) to detect word stems. This method did not prove satisfactory when applied to complex biological terms. Instead a regular expression matching algorithm was used to detect pairs of terms with a common word stem but word endings differing by a set number of letters (typically 3-4). This algorithm is applied recursively until no more stemming is possible, and appears to work extremely well for biological terms.

7. This stemmed go-list is used to generate a term occurrence array for each document in the set. This is a simple bit-vector Γ showing the

presence (1) or absence (0) of each term from the go-list in a given document. The length of this vector is equal to the size of the go-list, and a total of K vectors are generated, representing each document in the set. The total number of terms in the stemmed go-list has been reduced significantly when compared to set of all terms in the original K documents (term reduction for a control experiment is shown in Table 5.2). This set of selected terms should be highly reliable for document clustering techniques.

8. The fixed-length bit-vector representation of each documents abstract is used as input for a clustering procedure. The MineSet$^{\text{TM}}$data mining software package from Silicon Graphics was used for the clustering procedure. The *k-means* unsupervised clustering algorithm was used to detect clusters of documents sharing highly similar term bit-vectors. The MineSet$^{\text{TM}}$package was chosen because it allows many different clustering and statistical algorithms to be performed on a single source of data, and has a useful interactive 3D visualisation engine. When clustering is complete, a list of document clusters is generated along with significance values for each term used to generate that clustering. This is a very useful by-product of the clustering as it allows one to quickly determine the biological significance of each detected document cluster.

9. In order to generate a final set of terms that describe each cluster, a log-odds formula is applied to the resulting clusters and terms. This formula can be described as: $\Theta_{ij} = \log_2(f_{ij}f_i)$, where Θ_{ij} represents the preference of term i in a document cluster j, f_{ij} represents the frequency of term i in cluster j and f_i represents the frequency of term i in the total set of abstracts. If $f_{ij} = f_i$ (i.e. term i is as frequently found in cluster j as in the total set), then Θ_{ij} is zero. Values of Θ that are positive represent terms that are specific to a cluster and vice-versa. Terms are selected with a positive Θ value above a cut-off threshold τ.

10. Visualisation of the results allows one to quickly and intuitively analyse the results of the algorithm. For this reason the BioLayout algorithm (see Section 5.2) was used to visualise the results of the TextQuest algorithm when applied to a variety of datasets.

The algorithm reduces the complexity of the problem significantly due to its statistical detection of terms within abstracts which are biologically relevant. The algorithm is relatively fast as minimal processing of these data is required. Most of the processing involved is term-parsing and term-stemming, two steps which are extremely fast. The three threshold values described in the algorithm (ρ, σ, τ) have been optimised empirically by experiment. These parameters can however be modified from the default values by the user. While the method is relatively simple, it appears to be a highly robust algorithm for the document clustering of biological abstracts.

5.3.4 Document Clustering Results

In order to test the performance of the algorithm, a number of test cases were selected. These test cases involved the selection of specific sets of documents relating to different biological phenomena, mixing them together and clustering this mixed dataset. If the TextQuest method is performing well, then each biologically distinct set of documents should be separated again into their constituent clusters.

The first test-case used involved two sets of 830 articles relating to different biological areas. The first set of 830 articles were selected from Medline with the following query: "(escherichia AND pili)". The second set were selected using: "(cerevisiae AND cdc*)". Clearly the first set of documents should relate to bacterial processes involved in pili formation and function, while the second set relate to eukaryal cell division and cell cycle. This set of 1,660 documents was processed using the TextQuest algorithm in order to test the ability of the method to separate these two clearly distinct sets of documents. The term reduction (shown in Table 5.2) reduced the number of terms within these 1,660 abstracts from 162,499 to only 471 terms included in the go-list.

As hoped, the algorithm did indeed produce two distinct clusters from the input document set. These two clusters corresponded with the two initial input sets of 830 documents. Because the algorithm also generates log-odds scored clustering terms, it is also possible to list the key terms used in the separation of these two distinct document classes. Some of these words are listed in Table 5.3.

Clearly some of the these selective terms relate to species names such as "*melanogaster*", or experimental techniques such as "*elisa*", but the majority

Set of Terms	Step No.	Number of Terms
S	2	162,499
T	4	56,057
U	5	868
V	6	633
W	9	471
$\rho = 17$	$\sigma = 0.01K$	$\tau = 0.8$

Table 5.2: Term reduction during the control experiment.

Cluster 1	Cluster 2
alphafactor	adherence
budding	agglutination
centromere	antigenic
chromatin	bacteriophage
cln1	chloroform
cytoskeleton	conjugative
defines	diarrhoea
diploid	elisa
fission	fimbrial
gtpbinding	glycoproteins
meiosis	klebsiella
melanogaster	morphologically
microtubules	operons
nucleus	pfimbriae
phosphorylation	plasmidencoded
rad9	precipitation
rescues	pyelonephritis
spindle	serogroup
telomere	shigella
tumor	susceptible
ubiquitin	uropathogenic
uv	vaccination

Table 5.3: Most representative terms detected for each cluster.

of these terms appear to correlate well with the subject matter of the papers within a cluster. It is also evident that gene names are readily selected by the algorithm which is very encouraging. Terms such as "cln1" and "rad9" in cluster one clearly relate to genes involved in the cell-division and cell-cycle of eukaryotes. Some other informative terms include "nucleus" which is highly represented in the yeast documents, and "operons" which occur with high frequency in the *E. coli* set.

This first analysis using TextQuest provided validation for the method, and highlighted the importance of not only the document clustering procedure, but the detection of key terms that produce the clustering. Many clustering methods are *black-box* techniques which give very little information about why and how clusters are formed.

Given the accuracy of the method for the relatively easy task of separating articles from very different fields, it was decided to tackle a much more challenging problem. A set of 525 abstracts related to two separate *Drosophila melanogaster* developmental pathways were selected from Medline (Rongo et al., 1997). The query used to select this input set was: "(anterior-posterior AND drosophila) AND (dorsal-ventral AND drosophila)". Clearly these two developmental pathways are distinct, yet very closely related, and would provide much more of a challenge to the TextQuest system.

This analysis produced a final go-list of 409 terms, with the following parameter settings: $\rho = 19$, $\sigma = 0.01K$, $\tau = 0.5$. The unsupervised clustering of each 409 element bit-vector produced three distinct clusters from the 525 abstracts. This result was unexpected as we had provided two distinct document classes to the algorithm. Inspection of log-odds scores of key clustering terms showed clearly why three separate clusters had been obtained. The first cluster (206 abstracts) clearly described genes involved the development of the anterior-posterior axis of the *D. melanogaster* embryo (for example, bithorax, engrailed, hunchback, etc). The second cluster (251 abstracts) related clearly to words and genes associated with the dorsal-ventral pathway (dorsalizing, ventral-specific, pelle, notch, cactus, etc). The third unexpected cluster contained 68 abstracts labelled with terms such as 'oocyte', 'maternal-effect', 'germline' and 'polarized'. Close inspection of this cluster indicated that these abstracts were related to egg chamber development and oocyte patterning (van Eeden and Johnston, 1999), these processes occur before dorsal-ventral and anterior-posterior development, yet affect both of these pathways significantly. The third detected cluster hence contains abstracts

related to the early development of the *D. melanogaster* egg and embryo, while the other two clusters describe subsequent and separate events of *D. melanogaster* embryo development.

The ability of the TextQuest system to automatically detect these subtle differences between very closely related articles is reassuring. Indeed, the detection of a third and not obvious set of abstracts relating to egg-chamber patterning was impressive. We hope to extend the current TextQuest system, and perhaps to use the automatic term recognition of TextQuest with the rapid MCL clustering algorithm (Section 2.4) for clustering of the entire Medline database. Given the encouraging results we have obtained so far, a full clustering of Medline, may detect large quantities of accurate functional information concerning genes and proteins.

5.3.5 Implementation

The TextQuest system is composed of a set of routines[6] written in Perl and Awk. An input set of documents in Medline format is supplied and is processed sequentially by each of the routines. A matrix file containing bit-vectors for each document is produced. This matrix file may be clustered with any conventional clustering technique, not just the MineSet^{TM}system.

[6]http://www.ebi.ac.uk/research/cgg/projects/mining/textquest/

5.4 Consensus Annotation of Protein Families

A large proportion of this thesis details the analysis and detection of protein families. Frequently protein families are used for annotation purposes. In these cases a protein whose function is unknown may be annotated with respect to other proteins in the same family whose functions are known. The problem involves the generation of a consensus annotation for a set of proteins in a protein family, based on any annotation information available for proteins within that family.

5.4.1 Formulation of the Problem

Given a set of proteins (P) and their corresponding annotations (A), the problem is to generate a consensus annotation which will describe as many of these proteins as possible. An example set of such proteins and annotations is shown for a 'cytochrome c oxidase' family in Table 5.4.

Ideally these consensus annotations should be as descriptive as possible (longer) and cover as many proteins in the input set as possible (highly specific). In order to generate consensus annotations one needs a method for detecting common words or phrases shared by each annotation in the set. After experimenting with different pattern matching algorithms and regular expression systems. It was decided to use *longest common substring* detection to generate this information. This analysis is rapid and can be easily adapted for matching words or phrases. When used for word matching this method will return the longest set of words that two annotations share, in an order dependent manner. An example of this for two annotations is:

```
1) CYTOCHROME C OXIDASE POLYPEPTIDE IV, MITOCHONDRIAL PRECURSOR (EC 1.9.3.1)
2) CYTOCHROME C OXIDASE POLYPEPTIDE VB-LIVER (EC 1.9.3.1) (FRAGMENT)
```

The longest common substring between these two sets of annotations (at the word level) is:

```
CYTOCHROME C OXIDASE POLYPEPTIDE (EC 1.9.3.1)
```

The method can return a relatively meaningful consensus annotation of two different annotations. When two annotations have no words in common, then no consensus can be generated. Formally the problem of detecting longest

Protein ID	Genome	Annotation
ATHA-XXX-002421	Arabidopsis thaliana	cytochrome c oxidase subunit, putative similar to cytochrome c oxidase subunit Vb GI:1841354 from [Oryza sativa]
ATHA-XXX-011980	Arabidopsis thaliana	putative cytochrome c oxidase subunit Vb similar to cytochrome oxidase IV GB:223590 [Bos taurus];
CELE-XXX-006559	Caenorhabditis elegans	CE09693 cytochrome C oxidase (HINXTON) TR:P90849 protein_id:CAB03002.1
DMEL-XXX-007306	Drosophila melanogaster	last_updated:000321 (translated)
DMEL-XXX-007307	Drosophila melanogaster	last_updated:000321 (translated)
HSAP-XXX-016135	Homo sapiens	
SCER-S28-002187	Saccharomyces cerevisiae	S288CCOX4, Chr VII from 149708-150175, reverse complement
P00428	Bos taurus	CYTOCHROME C OXIDASE POLYPEPTIDE VB (EC 1.9.3.1)(VI)
P04037	Saccharomyces cerevisiae	CYTOCHROME C OXIDASE POLYPEPTIDE IV, MITOCHONDRIAL PRECURSOR (EC 1.9.3.1)
P10606	Homo sapiens	CYTOCHROME C OXIDASE POLYPEPTIDE VB, MITOCHONDRIAL PRECURSOR (EC 1.9.3.1)
P12075	Rattus norvegicus	CYTOCHROME C OXIDASE POLYPEPTIDE VB, MITOCHONDRIAL PRECURSOR (EC 1.9.3.1) (VIA*)
P19536	Mus musculus	CYTOCHROME C OXIDASE POLYPEPTIDE VB, MITOCHONDRIAL PRECURSOR (EC 1.9.3.1)
P29505	Dictyostelium discoideum	CYTOCHROME C OXIDASE POLYPEPTIDE V (EC 1.9.3.1)
P79010	Schizosaccharomyces pombe	CYTOCHROME C OXIDASE POLYPEPTIDE IV, MITOCHONDRIAL PRECURSOR (EC 1.9.3.1)
P80330	Oncorhynchus mykiss	CYTOCHROME C OXIDASE POLYPEPTIDE VB-LIVER (EC 1.9.3.1) (FRAGMENT)

Table 5.4: A set of proteins and annotations from a cytochrome c oxidase family.

common substrings is defined as follows:

A subsequence is defined as a subset of the characters of a sting S arranged in their original "relative" order . A subsequence of the string S of length n is specified by a list of indices $i_1 < i_2 < i_3 < \ldots < i_k$, for some $k \leq n$. The subsequence specified by this list of indices is the string $S_{i_1}, S_{i_2} \ldots, S_{i_k}$.

Given two strings S and T, a *common subsequence* is a subsequence that appears both in S and T. The *longest common subsequence problem* is to find the longest subsequence common to both S and T.

5.4.2 Longest Common Substring (LCS) Detection

Many methods have been explored for solving the longest common subsequence problem (Hirschberg, 1977). The algorithm described here is the common dynamic programming approach (Hirschberg, 1975) with memorisation. While not as fast as some optimal methods using suffix trees (Gusfield, 1997), the algorithm implementation is straightforward and fast enough for the purposes of protein family annotation. When given two arrays filled with words (from protein annotations), the algorithm returns the size of the longest common subsequence (number of shared words), and the common subsequence of words from both annotations.

5.4.3 An LCS Algorithm for Automatic Consensus Annotation

The algorithms described above compute the longest common subsequence of pairs of strings. In order to annotate a protein family, all common subsequences between all annotations need to be computed. To this end a recursive algorithm has been developed which processes calculates longest common subsequence annotations between all pairs of annotations. At each iteration, common subsequence annotations from previous iterations are reprocessed to find a maximal set of subsequence patterns which cover all annotations within the input set.

Each detected subsequence annotation i from the covering set of all com-

mon subsequences is then assigned a score S_i according to:

$$S_i = CL_i + (f_i \times 100)$$

Where L_i is the length of the annotation i counted in words, C is a constant and f_i is the frequency of this subsequence annotation in the original input set of protein annotations.

This scoring scheme attempts to score subsequence annotations in terms of their descriptiveness and specificity. Consensus annotations which are longer, are generally more descriptive, while subsequence annotations which occur frequently in the input set of original annotations are more specific to that family. A good example of this is the word "protein" which occurs at very high frequency in almost all protein annotations, while highly specific, it is by no means a descriptive annotation. The C constant is hence used to weight the scoring scheme between longer (less specific) annotations and shorter (more specific) annotations. Empirically a value of $C = 12$ has been chosen for most analyses using this algorithm. A score S is hence generated for each subsequence annotation generated by the algorithm, the highest scoring subsequence is then chosen as the consensus annotation for the input set of protein families.

Testing with data from SwissProt (Bairoch and Apweiler, 2000) protein families has shown that the algorithm is very accurate at producing meaningful consensus annotations for related proteins within families (Enright et al., 2002). Care must be taken however to filter out sets of words that are commonly used in sequencing projects. This is reminiscent of compositional bias filtering for amino acid sequences. Phrases such as:

```
"HYPOTHETICAL PROTEIN SEQUENCE"
```

can generate high scores. For this reason a pre-processing step eliminates words such as "predicted", "hypothetical", "putative" and "fragment", which are of little use in a consensus annotation.

This method has been used extensively for the annotation of protein families in the Ensembl database (Hubbard et al., 2002), and the Tribes database (described in Sections 3.2 & 3.3). The method represents a rapid and general method for assembling consensus annotations for sets of protein or nucleotide sequences.

5.5 High-Throughput Tools for Sequence Analysis

Working with complete genomes requires an enormous amount of computation. For protein family analysis, frequently all-against-all BLAST (Altschul et al., 1997) comparisons need to be performed. The analyses described in Sections 3.2 & 3.3, each performed sequence comparisons for more than 500,000 protein sequences, a total of 2×10^{11} individual BLAST comparisons. Obviously this computation was performed on a large supercomputer, but even single genome analyses can require more computation than can be provided by a single workstation, to run in a realistic time.

The computational genomics group has a single Sun Microsystems four CPU Enterprise server and eleven Sun UltraSparc workstations. This represents a total of fifteen CPUs which, as they are not constantly in use, can be utilised for sequence comparisons. To this end, we have extended conventional sequence analysis methods to operate on parallel multi-processor environments. This speeds up large-scale sequence analysis and exploits the processing power of multiple machines available within the computational genomics group. The most important of these tools (Table 5.5) will be described in the following sections.

5.5.1 htBLAST.pl and Parse.pl

The BLAST algorithm (from NCBI and Washington University) is designed primarily to take a single query sequence in FASTA format and compare it to a reference database of sequences. For many bioinformatics analyses a database of these query sequences needs to be compared against such a reference database. In the case of protein family analysis, all-against-all comparisons are usually undertaken. This involves searching a database of sequences against itself. Because BLAST is designed primarily for single sequence comparisons, it is possible to greatly improve performance by correct handling of sequences and databases.

A wrapper script called *htBLAST* was developed for high-throughput BLAST comparisons of multiple sequences. This small script improves the performance of large BLAST analyses by as much as 30% and works as follows:

1. Read in a set of FASTA formatted query sequences into memory.

Program	Language	Function
htBLAST.pl	Perl	Wrapper script for BLAST analyses
parallel.pl	Perl	Wrapper script for parallel analyses (such as: BLAST, CLUSTAL, GeneRAGE, etc.).
parse.pl	Perl	Rapid parsing of BLAST results.
fetcher.pl	Perl	Rapid fetching of FASTA sequences from a database.
chopper.pl	Perl	Division of a BLAST query file into sensible pieces.
queue	C	A Queuing system with file-locking for web based services (such as: CAST and BLAST).
lsfBLAST.pl	Perl	An lsf (load sharing facility) wrapper for super-computer analyses.

Table 5.5: A listing of all tools written for high-throughput sequence analysis.

2. Start an individual BLAST run for the first query sequence.

3. While a BLAST query is being run, load the next sequence into a buffer.

4. When a BLAST query completes, send the next query sequence from the buffer.

5. Store all BLAST results in memory while this procedure takes place.

6. When the last sequence has been compared, dump all BLAST results to a file.

Another small Perl script called *parse.pl* uses standard regular expressions to detect key information from a standard BLAST output file and display only desired information. This reduces the size needed to store BLAST results by approximately 10 times, yet does not result in any loss of information.

5.5.2 Parallel.pl

The fifteen processors available to us are reasonably powerful and fast. Given that users infrequently utilise their workstation processors fully, it makes sense to develop software to use these wasted CPU cycles for performing large-scale sequence analysis. All of these SUN machines use the same version of UNIX (Solaris 7) and have access to the same filesystems. During the night most group machines are inactive, it is be possible to start an individual set of BLAST procedures on each machine, then once finished, collate the individual results into a final result. This should result in something close to a 15× speedup. In principle the UNIX operating system is well designed to handle this procedure automatically.

Many commercial applications exist to perform this type of function in an easy and transparent fashion (for example the 'Load Sharing Facility™' package from Platform Computing ®[7]). However these applications are prohibitively expensive for a single research group to purchase. For this reason, the *parallel.pl* script was developed to automatically distribute sequence analysis tasks to each of the 12 machines and 15 processors in the group. The *parallel.pl* script is written using Perl and its action can be described as follows:

[7] http://www.platform.com/

1. Prompt the user for a list of hosts to use, or read from a file.

2. Briefly connect to each host using UNIX system calls to make sure the host is up and not busy. Store a list of these hosts and the total number of hosts available (N).

3. Read in a set of Q query sequences to be processed.

4. Use the *chopper.pl* (Table 5.5) script to divide this set Q into a set of S subsets of equal or near to equal size in bytes.

5. Use the UNIX *fork* system call to start N slave copies of the parallel.pl script, one on each of the N host machines. Record the PID identifier for each forked process in the master script. Each slave process is given one of the S subsets to process by the master script as they are forked.

6. The master copy of the script adopts a wait loop and monitors each slave process to detect when it has terminated. When a slave process finishes its job, the master script gives the slave another set of sequences from the set S of all sequence subsets.

7. The master script waits until all S subset jobs have been distributed, then terminates when all N hosts report completion.

8. When all slaves have finished, they self-terminate, and the master script collates results from each of the S jobs into a single results file for the whole analysis.

This small and useful script can be adapted to perform BLAST (Altschul et al., 1997), FASTA (Pearson and Lipman, 1988), Smith-Waterman (Smith and Waterman, 1981) and many other types of sequence analysis algorithms. In practice it is being used daily by members of the computational genomics group, and has allowed group members to perform very large analyses without having to adapt their scripts for working on large supercomputers or processor farms. The most common form of the script is called *blast-parallel.pl* and is a combination of the *parallel.pl*, *htBlast.pl* and *parse.pl* scripts. This allows the user to perform a large parallel BLAST analysis by typing in a simple one line command in UNIX that is very similar to the command-line invocation of simple single-sequence BLAST.

5.5.3 lsfBlast.pl

The comparison described in Section 3.3 involved a total of 2×10^{11} individual BLAST analyses. An analysis of this size would take over a year on a typical workstation, and weeks or months using the *parallel.pl* script described earlier. A powerful 400 node Compaq Alpha cluster (used for Ensembl human genome assembly) was used in this case for rapid parallel BLAST analysis.

Due to the number of machines involved, the commercial 'Load Sharing Facility' (LSF) package was installed on this machine. This package consists of a central daemon which allocates jobs, and a job management daemon on each of the 400 nodes. It is possible using lsf commands to submit a job or jobs to a queue on any machine. These queued jobs will eventually be started on one of the 400 machines by the master daemon.

Even using a commercial job scheduling and distribution package, it is still non-trivial to perform a BLAST analysis of this size. For this reason a separate script called *lsfBlast.pl* was written to perform very large-scale analyses on large supercomputers using the LSF package. The *lsfBlast.pl* program is written in Perl and works as follows:

1. We begin with a database of 500,000 protein sequences in FASTA format

2. A copy of this database is formatted for BLAST and distributed to local storage on each of the 400 machines (nodes) using the UNIX *rdist* tool.

3. Another local copy of the database is split into 400 pieces using the *chopper.pl* script.

4. Each of these 400 query sets is then filtered for compositional bias using the CAST package, which is described in Section 5.1 (Promponas et al., 2000).

5. A call is made to LSF using the *bsub* command to start 400 separate jobs on the machine. Each LSF job submitted can be described as follows:

   (a) Copy one of the 400 subsets to a free node.

   (b) Call the BLASTp program to search this subset against the database copied previously to local storage.

(c) Automatically parse the output from BLASTp using the *parse.pl* script, then compress the final result with *gzip*.

(d) Finally, copy this results file back to the submission machine.

6. The final step is to concatenate each of the resulting output files from each separate LSF job and check these results for errors.

7. The final parsed output file for all BLAST runs is then ready for decompression and analysis.

Conclusions

The enormous quantities of biological sequence data being produced represent a wealth of information. In this thesis I have described a number of approaches which seek to make use of these data for the large-scale functional and evolutionary analysis of proteins. The guiding philosophy for this research has been the development of tools which are as robust and automatic as possible. The quantities of data now available for biological research necessitate that methods should not require manual intervention in order to work successfully.

At one level, I have presented two novel algorithms for the classification of proteins into related evolutionary groups, or protein families. These families are useful tools for assigning function to proteins, and for conducting fundamental research into genome evolution. The GeneRAGE algorithm provides detailed and accurate clustering, tailored for bacterial genomes, while the Tribe-MCL algorithm is designed to handle much larger (and more complex) eukaryotic genomes. The success of these methods has allowed many interesting experiments to be performed which seek to explore the evolution of protein function in complete genomes.

On another level, the DifFuse tool was developed to determine functional associations or even direct physical interactions between proteins which exhibit little or no sequence similarity. This method relies on the detection of gene-fusion events in complete genomes to infer functional links between proteins. The method belongs to a family of approaches which have recently proved successful for computational genome research. These methods detect functional links between proteins by analysing the genomic context of genes and proteins in complete genomes. This method has proved very successful for prediction of protein-protein interaction networks and I am very much looking forward to validating these results more fully, when accurate high-throughput experimental and computational technologies for the prediction

of protein-protein interactions reach maturity.

During this work some novel and potentially interesting results were produced. I have striven to make this information freely available and easy to access by others. While the goal has not explicitly been to create computational services for the biological community, I intend to maintain and help curate a number of projects that have been described in this thesis.

The Tribes database represents and important and potentially very useful resource for the scientific community, and it is my intention to maintain, improve and update this database over the coming years. Other projects such as the Complete Genomes Database, were developed specifically for this work, but also represent potentially very useful projects that will hopefully be maintained.

I would like to believe that some of the other smaller tools and methods developed, such as BioLayout will also prove useful in the field. BioLayout has already attracted a lot of attention, as it is one of a very small number of bioinformatics visualisation tools. I believe that data visualisation will become more and more important over the coming years, as the flood of genomic data rapidly increases.

It has been a rewarding and enjoyable experience working at the European Bioinformatics Institute. The Wellcome Trust Genome Campus is home to more than 800 people, and the opportunity to meet researchers with such diverse backgrounds and nationalities has been a stimulating experience. I believe that I have learned a great deal from meeting people and exchanging ideas during my time on campus.

Appendix A

Papers Published During this Work

The following list details papers which were published in scientific journals during the course of this thesis.

- Enright A.J., Ouzounis C.A.; *Nucleic Acids Research* 7:43-53 (2002)
  'An efficient algorithm for the detection of protein families'.

- Iliopoulos I., Tsoka S., Andrade M., Enright A.J., Carroll M., Poullet P., Promponas V., Liakopoulos T., Palaios G., Pasquier C., Hamodrakas S., Tamames J., Yagnik A., Tramontano A., Devos D., Blaschke C., Valencia A., Brett D., Martin D., Leroy C., Rigoutsos I., Sander C., Ouzounis C.A.; *Submitted* (2002).
  'Evaluation of Annotation Strategies using an Entire Genome Sequence'.

- Enright A.J., Ouzounis C.A.; *In Press* (2002); Protein-Protein Interactions: a Molecular Cloning Manual (eds. Golemis, E., and Serebriiskii, I.), Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY.
  'Computational Detection of Genome-wide Protein Interaction'.

- Boucher L., Ouzounis C.A., Enright A.J., Blencowe B.J; *RNA* 7:1693-1701 (2001).
  'A Genome-wide survey of RS-domain proteins'.

- Enright A.J., Ouzounis C.A.; *Bioinformatics* 17:853-854 (2001).
  'Biolayout - An Automatic Graph Layout Algorithm for Similarity Vi-

sualisation and Analysis'.

- Enright A.J., Ouzounis C.A.; *Genome Biology* 2(9):research0034.1-0034.7 (2001).
  'Functional Associations of Proteins in entire Genomes via exh austive detection of Gene fusion events'.

- Coulson R.M., Enright A.J., Ouzounis C.A.; *Bioinformatics* 17:95-97 (2001).
  'Transcription-associated protein families are primarily taxon-specific'.

- Iliopoulos I., Enright A.J., Ouzounis C.A.; *Pacific Symposium on Biocomputing* 384-395 (2001).
  'Textquest: document clustering of medline abstracts for concept discovery in molecular biology'.

- Promponas V.J, Enright A.J., Kreil D., Tsoka S., Leroy C., Hamodrakas S., Sander C., Ouzounis C.A.; *Bioinformatics* 16:915-22 (2000).
  'CAST: an iterative algorithm for the Complexity Analysis of Sequence Tracts'.

- Enright A.J., Tsoka S., Ouzounis C.A.; *ERCIM News* 43:8-9 (2000).
  'Making sense of complete genomes'.

- Enright A.J., Ouzounis C.A.; *Bioinformatics* 16:451-457 (2000).
  'GeneRAGE, a robust algorithm for sequence clustering and domain detection'.

- McLysaght A., Enright A.J., Skrabanek L., Wolfe K.H.; *Yeast* 17:22-36 (2000).
  'Estimation of synteny conservation and genome compaction between pufferfish (fugu) and human'.

- Enright A.J., Iliopoulos I., Krypides N., Ouzounis C.A.; *Nature* 402:86-90 (1999).
  'Protein interaction maps for complete genomes based on gene fusion events'.

Appendix B

CGD and Tribes Database Schema

The Complete Genomes Database (CGD) and the Tribes database are linked together into a single MySQL entity. The MySQL table definitions are shown in this Appendix, together with an Entity Relationship (ER) diagram (Figure B.1) illustrating how the individual tables link together. These databases are populated separately using two Perl scripts which utilise the `DBI` and `DBD::mysql` modules for database access. Both scripts are fully listed at the end of this appendix.

The initial population of CGD requires a FASTA file containing sequences from a complete genome. The CGD database is populated by a Perl script called *add_genome.pl*. This script takes input from the user then automatically generates unique identifiers for each sequence and stores them in the database. Sequences are stripped of non-standard characters such as carriage returns and '*' or '\'. The current set of 74 genomes that have been indexed in the CGD database is shown on the following page.

Population of the Tribes database requires clustering results from the Tribe-MCL algorithm, results from BLAST and automatically generated consensus annotations derived from the RLCS algorithm (not listed). All this processing is handled by the *add_families.pl* Perl script which is fully listed at the end of this Appendix.

The Set of 74 Complete Genomes, currently indexed in CGD.

species_code	fullname	total_genes	size_mb	date_sequenced
AAEO-VF5	Aquifex aeolicus, VF5	1553	1.55	26/03/1998
AFUL-DSM	Archaeoglobus fulgidus, DSM4304	2409	2.18	27/11/1997
APER-XK1	Aeropyrum pernix, K1	2694	1.67	30/04/1999
ATHA-XXX	Arabidopsis thaliana	25761	115.43	14/12/2000
ATUM-C58	Agrobacterium tumefaciens, C58	5299	5.66	17/08/2001
BBUR-B31	Borrelia burgdorferi, B31	1639	1.23	11/12/1997
BHAL-C12	Bacillus halodurans, C-125	4066	4.20	01/11/2000
BMEL-M16	Brucella melitensis, M16	3198	3.29	08/01/2002
BSUB-168	Bacillus subtilis, 168	4093	4.21	20/11/1997
BUCH-APS	Buchnera sp., APS	575	0.64	07/09/2000
CACE-ATC	Clostridium acetobutylicum, ATCC 824	3916	4.13	15/08/2001
CCRE-XXX	Caulobacter crescentus, CB15	3737	4.02	27/03/2001
CELE-XXX	Caenorhabditis elegans	19957	97.00	11/12/1998
CJEJ-NCT	Campylobacter jejuni, NCTC 11168	1634	1.64	10/02/2000
CPER-X13	Clostridium perfringens, str. 13	2723	3.03	22/01/2002
CPNE-AR3	Chlamydia pneumoniae, AR39	1119	1.23	15/03/2000
CPNE-CWL	Chlamydia pneumoniae, CWL029	1052	1.23	01/04/1999
CPNE-J13	Chlamydia pneumoniae, J138	1070	1.23	15/06/2000
CTRA-MOP	Chlamydia trachomatis, MoPn	921	1.07	15/03/2000
CTRA-SVD	Chlamydia trachomatis, serovar D	894	1.04	23/10/1998
DMEL-XXX	Drosophila melanogaster	13054	137.00	24/03/2000
DRAD-XR1	Deinococcus radiodurans, R1	3116	3.28	19/11/1999
ECOL-EDL	Escherichia coli O157:H7, EDL933	5349	4.10	25/01/2001
ECOL-MG1	Escherichia coli, MG1655	4290	4.64	05/09/1997
ECOL-RIM	Escherichia coli 0157:H7, RIMD0509952	5447	5.59	28/02/2001
HALO-NRC	Halobacterium sp., NRC-1	2605	2.01	24/10/2000
HINF-KW2	Haemophilus influenzae, KW20	1707	1.83	28/07/1995
HPYL-266	Helicobacter pylori, 26695	1575	1.67	07/08/1997
HPYL-J99	Helicobacter pylori, J99	1491	1.64	14/01/1999
HSAP-XXX	Homo sapiens	27333	2900.00	15/02/2001
LINN-CLI	Listeria innocua, CLIP 11262	2968	3.01	26/10/2001
LLAC-IL1	Lactococcus lactis, IL1403	2266	2.36	02/05/2001
LMON-EGD	Listeria monocytogenes, EGD-e	2846	2.94	26/10/2001
MGEN-G37	Mycoplasma genitalium, G-37	479	0.58	20/10/1995
MJAN-DSM	Methanococcus jannaschii, DSM 2661	1773	1.66	23/08/1996
MLEP-XTN	Mycobacterium leprae, TN	1605	3.23	22/02/2001
MLOT-MAF	Mesorhizobium loti, MAFF303099	7281	7.60	03/12/2000
MPNE-M12	Mycoplasma pneumoniae, M129	689	0.82	15/11/1996
MPUL-UAB	Mycoplasma pulmonis, UAB CTIP	782	0.96	15/05/2001
MTHE-DEL	Methanobacterium thermoautotrophicum, deltaH	1871	1.75	15/11/1997
MTUB-CDC	Mycobacterium tuberculosis, CDC1551	4203	4.40	25/04/2001
MTUB-H37	Mycobacterium tuberculosis, Hv37	3924	4.41	11/06/1998
NMEN-MC5	Neisseria meningitidis, MC58	2081	2.27	10/03/2000
NMEN-Z24	Neisseria meningitidis, Z2491	2065	2.18	30/03/2000
NOST-PCC	Anabaena sp., strain PCC 7120	6129	7.21	31/10/2001
PABY-GE5	Pyrococcus abyssi, GE5	1765	1.76	22/12/1999
PAER-IM2	Pyrobaculum aerophilum, IM2	2605	2.22	22/01/2002
PAER-PAO	Pseudomonas aeruginosa, PAO1	5570	6.26	31/08/2000
PHOR-OT3	Pyrococcus horikoshii, OT3	2061	1.74	30/04/1998
PMUL-PM7	Pasteurella multocida, Pm70	2014	2.25	13/03/2001
RCON-MAL	Rickettsia conorii, str. Malish 7	1374	1.27	14/09/2001
RPRO-MAD	Rickettsia prowazekii, Madrid E	834	1.11	12/11/1998
RSOL-XXX	Ralstonia solanacearum	5116	5.70	11/12/2001
SAUR-MU5	Staphylococcus aureus, Mu50	2748	2.88	21/04/2001
SAUR-N13	Staphylococcus aureus, N315	2624	2.81	21/04/2001
SCER-S28	Saccharomyces cerevisiae, S288C	6357	12.07	12/06/1997
SENT-CT1	Salmonella enterica serovar Typhi, CT18	4763	5.10	25/10/2001
SENT-LT2	Salmonella enterica serovar Typhimurium, LT2	4553	4.95	25/10/2001
SMEL-102	Sinorhizobium meliloti, strain 1021	6206	6.68	27/07/2001
SPNE-TIG	Streptococcus pneumoniae, TIGR4	2140	2.20	20/07/2001
SPNE-XR6	Streptococcus pneumoniae, R6	2043	2.04	01/10/2001
SPOM-XXX	Schizosaccharomyces pombe	4945	13.80	21/02/2002
SPYO-SF3	Streptococcus pyogenes GAS, M1	1696	1.85	10/04/2001
SSOL-XP2	Sulfolobus solfataricus, P2	2996	2.99	03/07/2001
STOK-XX7	Sulfolobus tokodaii, str. 7	2826	2.69	31/08/2001
SYNE-PCC	Synechocystis sp., PCC6803	3167	3.57	30/06/1996
TACI-DSM	Thermoplasma acidophilum, DSM1728	1478	1.56	28/09/2000
TMAR-MSB	Thermotoga maritima, MSB8	1849	1.86	27/05/1999
TPAL-NIC	Treponema pallidum, Nichols	1030	1.14	17/07/1998
TVOL-GSS	Thermoplasma volcanium, GSS1	1526	1.58	20/12/1999
UURE-SV3	Ureaplasma urealyticum, serovar 3	613	0.75	12/10/2000
VCHO-N16	Vibrio cholerae, El Tor N16961	3835	4.00	03/08/2000
XFAS-9A5	Xylella fastidiosa, 9a5c	2830	2.68	13/07/2000
YPES-CO9	Yersinia pestis, CO92	4093	4.82	04/10/2001

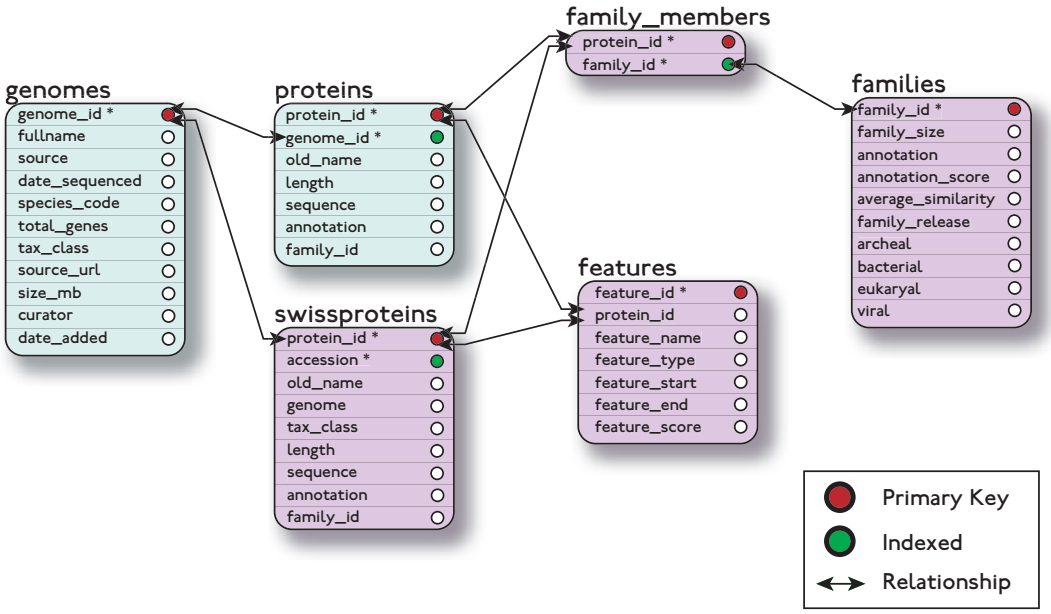


Figure B.1: Entity Relationship (ER) Diagram for the CGD and Tribes databases. Relationships between tables are shown with arrows. Each index is shown with a coloured circle.

```
+----------------+------------------+------+-----+---------+----------------+
| Field          | Type             | Null | Key | Default | Extra          |
+----------------+------------------+------+-----+---------+----------------+
| genome_id      | int(10) unsigned |      | PRI | 0       | auto_increment |
| fullname       | varchar(100)     |      |     |         |                |
| source         | varchar(40)      |      |     |         |                |
| date_sequenced | varchar(10)      |      |     |         |                |
| species_code   | varchar(8)       |      |     |         |                |
| total_genes    | int(10)          | YES  |     | NULL    |                |
| tax_class      | varchar(200)     |      |     |         |                |
| source_url     | varchar(150)     | YES  |     | NULL    |                |
| size_mb        | float(10,2)      | YES  |     | NULL    |                |
| curator        | varchar(8)       |      |     |         |                |
| date_added     | varchar(40)      |      |     |         |                |
+----------------+------------------+------+-----+---------+----------------+
```

Figure B.2: MySQL table definition for the Genomes table

Field	Type	Null	Key	Default	Extra
protein_id	varchar(15)		PRI		
genome_id	int(10) unsigned		MUL	0	
old_name	varchar(40)	YES		NULL	
length	int(10)	YES		NULL	
sequence	mediumtext	YES		NULL	
annotation	varchar(250)	YES		NULL	
family_id	varchar(40)	YES		NULL	

Figure B.3: MySQL table definition for the Proteins table

Field	Type	Null	Key	Default	Extra
protein_id	int(10) unsigned		PRI	0	
accession	varchar(20)		MUL		
old_name	varchar(20)	YES		NULL	
genome	varchar(100)	YES		NULL	
tax_class	varchar(200)	YES		NULL	
length	int(10)	YES		NULL	
sequence	mediumtext	YES		NULL	
annotation	varchar(250)	YES		NULL	
family_id	varchar(40)	YES		NULL	

Figure B.4: MySQL table definition for the Swiss_Proteins table

Field	Type	Null	Key	Default	Extra
feature_id	int(10) unsigned		PRI	0	auto_increment
protein_id	int(10) unsigned			0	
feature_name	varchar(15)				
feature_type	varchar(40)	YES		NULL	
feature_start	int(11)	YES		NULL	
feature_end	int(11)	YES		NULL	
feature_score	double(16,4)	YES		NULL	

Figure B.5: MySQL table definition for the Feature table

```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| protein_id | varchar(15) |      | PRI |         |       |
| family_id  | varchar(40) |      | MUL |         |       |
+------------+-------------+------+-----+---------+-------+
```

Figure B.6: MySQL table definition for the Family_Members table

```
+--------------------+-----------------+------+-----+---------+-------+
| Field              | Type            | Null | Key | Default | Extra |
+--------------------+-----------------+------+-----+---------+-------+
| family_id          | varchar(40)     |      | PRI |         |       |
| family_size        | int(10) unsigned | YES  |     | NULL    |       |
| annotation         | varchar(250)    | YES  |     | NULL    |       |
| annotation_score   | int(3) unsigned | YES  |     | NULL    |       |
| average_similarity | int(3) unsigned | YES  |     | NULL    |       |
| family_release     | int(10) unsigned | YES  |     | NULL    |       |
| archeal            | int(1) unsigned | YES  |     | 0       |       |
| bacterial          | int(1) unsigned | YES  |     | 0       |       |
| eukaryal           | int(1) unsigned | YES  |     | 0       |       |
| viral              | int(1) unsigned | YES  |     | 0       |       |
+--------------------+-----------------+------+-----+---------+-------+
```

Figure B.7: MySQL table definition for the Families table

add_genomes.pl

```
#!/ebi/mig/src/bin/perl
$|=1;

#use lib "/ebi/mig/src/lib/perl5/site_perl";
use DBI;
use DBD::mysql;
use Term::ReadLine;
use Term::ReadKey;

$term = new Term::ReadLine 'ProgramName';
$database="CGD";
$user="root";
$machine="maine.ebi.ac.uk";

system("clear");
print "-----------------------------------------------------------------------\n";
print "AddGenome v0.9\n";
print "\n";
print "Adds a Genome to the CGD MySQL Database\n";
print "-----------------------------------------------------------------------\n\n";

if (!$ARGV[0])
        {
        print "Usage:  addgenome genome.fasta\n";
        print "\n";
        print "Where genome.fasta is a FASTA formated flatfile of all peptides in a genome\n\n";
        exit(0);
        }

if (!-e $ARGV[0])
        {
        die "Error: File ($ARGV[0]) does not exist\n";
        }

ReadMode 4;
$accesspassword = $term->readline("Please Enter Password:");
ReadMode 0;
print "\n\n";


$dbh = DBI->connect("DBI:mysql:$database:$machine", $user, $accesspassword) or die "Cannot Connect\n";

$statement="LOCK TABLES genomes WRITE, proteins WRITE, features WRITE\;";
$sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
$rv = $sth->execute or die "cant execute the query: $sth->errstr";

$statement="SELECT COUNT(*) FROM genomes\;";

$sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
$rv = $sth->execute or die "cant execute the query: $sth->errstr";

$total_genomes= ($sth->fetchrow_array())[0];
$total_genomes++;


        $prompt="Please enter the fullname of this genome:";
        while (!$fullname)
                {
```

```
                $_ = $term->readline($prompt);
                if (length($_) > 2)
                        {
                        $fullname=$_;
                        }
                else
                        {
                        print "Error please try again\n";
                        }
                }

        $prompt="Please enter the date this genome was published (Format DD/MM/YYYY):";
        while (!$date)
                {
                $_ = $term->readline($prompt);
         if(((/([0-9]{2})\/([0-9]{2})\/[0-9]{4}/) && ("$1"<=31) && ("$2" <=12))
                        {
                        $date=$_;
                        }
                else
                        {
                        print "Error please try again\n";
                        }
                }


        $prompt="Please enter the source of this data (eg. NCBI, TIGR, University of Washington etc..:";
        while (!$source)
                {
                $_ = $term->readline($prompt);
                if (length($_) > 2)
                        {
                        $source=$_;
                        }
                else
                        {
                        print "Error please try again\n";
                        }
                }


print "\n\n\nPlease enter a 7 letter species/strain code for this species.\n";
print "The first four letters of the code indicate the species.\n";
print "The last three represent the strain.\n\n";
print "Seperate the species and strain codes with '-'\n\n";
print "A default code will be automatically generated. The first letter comes from the genus, the\n";
print "next 3 letters from the species and it assumes that the strain is unknown and attaches 'XXX'.\n";
print "If you accept this code please press - enter.\n";
print "If the strain of the species is known please enter the code for the strain.\n";
print "If it is a 2-letter code place\n";
print "an 'X' before the 2 letters.\n\n";


        print "Examples:\n\n";
        print "           Saccharomyces cerevisiae                 SCER-XXX\n";
        print "           Caenorhabditis elegans                   CELE-XXX\n";
        print "           Helicobacter pylori - Strain J99         HPYL-J99\n";
        print "           Escherchia coli - Strain K5              ECOL-XK5\n\n\n\n\n";


while (!$genome_code)
        {
```

```
        $automatic_code=uc(join("",substr((split(" ",$fullname))[0],0,1),
substr((split(" ",$fullname))[1],0,3),"-XXX"));

        $prompt="Enter Species Code (XXXX-YYY): [Default: $automatic_code]";
        $temp= $term->readline($prompt);

            if (!$temp)
            {
                $temp=$automatic_code;
            }

        $statement="SELECT species_code from genomes where species_code =\"$temp\"";
        $sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
        $rv = $sth->execute or die "cant execute the query: $sth->errstr";
        $used_already = ($sth->fetchrow_array())[0];
        if (!$used_already)
                {
                $_=$temp;

                if ( (length($temp) == 8 ) && (/[A-Za-z]{4}-.{3}/))
                        {
                        $genome_code=uc($temp);
                        }
                else
                        {
                        print "Error: Incorrect Species/Strain code entered\n";
                        }
                }
        else
                {
                print "\n\nError: This Species/Strain code is already in use, please try again..\n\n";
                }
        }


        $prompt="Please enter the full taxonomic string for the species - Genus Species, Strain:";
        while (!$taxon)
                {
                $_ = $term->readline($prompt);
                if (length($_) > 2)
                        {
                        $taxon=$_;
                        }
                else
                        {
                        print "Error please try again\n";
                        }
                }


        $prompt="Please enter the source URL (either HTTP:// or FTP://) [Optional]:";
        $source_url=NULL;
        $temp = $term->readline($prompt);
        if ($temp)
                {
                $source_url="\"$temp\"";
                }

        $prompt="Please enter the Size in Megabases of this species (Eg 14.23) [Optional]:";
        $genome_size=NULL;
```

```
        $temp = $term->readline($prompt);
        if ($temp)
                {
                $genome_size="\"$temp\"";
                }

$curator=$ENV{USER};
$current_date=`date +'%d/%m/%Y %H:%M:%S'`;
chop($current_date);

print "\nSubmission Information\n";
print "User: $curator ($current_date)\n";
print "------------------------------------------------------------------------\n";
print "The values you have entered are:\n\n";
print "Organism: $fullname\n";
print "Source: $source\n";
print "Date: $date\n";
print "Species Code: $genome_code\n";
print "Taxonomic Classification: $taxon\n";
print "Source URL: $source_url\n";
print "Genome Size (Mb): $genome_size\n\n";
print "Are these values correct- yes/no:";

while (($values ne 'yes') && ($values ne 'no'))
{
    $values=$term->readline('[yes/no]>');

    if($values eq "no")
    {
    exit;
    }
}

#PRINT LOG FILE
open (FILEOUT,"> submission.log");
print FILEOUT "\nSubmission Information\n";
print FILEOUT "User: $curator ($current_date)\n";
print FILEOUT "------------------------------------------------------------------------\n";
print FILEOUT "The values you have entered are:\n\n";
print FILEOUT "Organism: $fullname\n";
print FILEOUT "Source: $source\n";
print FILEOUT "Date: $date\n";
print FILEOUT "Species Code: $genome_code\n";
print FILEOUT "Taxonomic Classification: $taxon\n";
print FILEOUT "Source URL: $source_url\n";
print FILEOUT "Genome Size (Mb): $genome_size\n\n";
print FILEOUT "Are these values correct- yes/no:";
close FILEOUT;


$statement="INSERT INTO genomes VALUES ($total_genomes,\"$fullname\",\"$source\",
\"$date\",\"$genome_code\",NULL,\"$taxon\",$source_url,$genome_size,\"$curator\",\"$current_date\")\;";

$sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
$rv = $sth->execute or die "cant execute the query: $sth->errstr";
print "Done\n";

open (FILE,$ARGV[0]) or die "Error: cannot open FASTA file\n";

while (<FILE>)
        {
        chop($_);
```

```
        if (/^>(\S+)(.*)/)
                {
                $total_proteins++;
                $protein_id="$genome_code";
                $protein_id=sprintf("%s-%0.6d",$genome_code,$total_proteins);
                $old_id{$protein_id}=$1;
                $annotation{$protein_id}=$2;
                if (length($annotation{$protein_id}) > 250)
                        {
                        $annotation{$protein_id}=substr($annotation{$protein_id},0,250);
                        }
                $annotation{$protein_id}=~ s/\"//g;
                $annotation{$protein_id}=~ s/\'//g;
                }

        else
                {
                $sequences{$protein_id}=join("",$sequences{$protein_id},$_);
                }
        }

foreach $sequence (sort(keys(%sequences)))
        {

        $counter++;

        $percentage=sprintf("%5s",(($counter/$total_proteins)*100));
        print "$percentage\n";
        $peptide=$sequences{$sequence};
        $length=length($peptide);
        $peptide=~s/(.{60})/$1\n/g;
        $statement="INSERT INTO proteins VALUES (\"$sequence\",$total_genomes,
\"$old_id{$sequence}\",\"$length\",\"$peptide\",\"$annotation{$sequence}\",\"\")\;";

        $sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
        $rv = $sth->execute or die "cant execute the query: $sth->errstr";
        }

$statement="SELECT count(*) from proteins where genome_id=\"$total_genomes\"\;";
        $sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
        $rv = $sth->execute or die "cant execute the query: $sth->errstr";
        $total_proteins= ($sth->fetchrow_array())[0];
        print "Added $total_proteins\n";

$statement="UPDATE genomes SET total_genes=\"$total_proteins\" WHERE genome_id=\"$total_genomes\"\;";
print "$statement\n";
        $sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
        $rv = $sth->execute or die "cant execute the query: $sth->errstr";


$statement="UNLOCK TABLES\;";
$sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
$rv = $sth->execute or die "cant execute the query: $sth->errstr";
$dbh->disconnect;

print "Add Genome Complete\n";
```

add_families.pl

```
#!/ebi/mig/src/bin/perl
$|=1;
use DBI;
use DBD::mysql;
use Term::ReadLine;
use Term::ReadKey;
require 'consensus-annotate2.pl';

$term = new Term::ReadLine 'ProgramName';

$database="CGD";
$user="root";
$machine="maine.ebi.ac.uk";

system("clear");
print "-----------------------------------------------------------------------\n";
print "AddFamilies v1.0\n";
print "\n";
print "Adds Finished and Annotated MCL families to the Database\n";
print "-----------------------------------------------------------------------\n\n";

ReadMode 4;
$accesspassword = $term->readline("Please Enter Password for the database:");
ReadMode 0;
print "\n\n";

print "READING Species Descriptions\n";
open (PROC,"echo \"select proteins.protein_id, genomes.tax_class from proteins,
genomes where proteins.genome_id=genomes.genome_id\"| mysql -u cgg -h maine CGD|");

while (<PROC>)
        {
        chop($_);
        @array=split(" ",$_);
        $hash{$array[0]}=$array[1];
        }

close PROC;
open (PROC,"echo \"select accession, tax_class from swissproteins\" | mysql -u cgg -h maine CGD|");
while (<PROC>)
        {
        chop($_);
        @array=split(" ",$_);
        $hash{$array[0]}=$array[1];
        }

print "Done!\n";
print "Regenerating Families Table:";
$dbh = DBI->connect("DBI:mysql:$database:$machine", $user, $accesspassword) or die "Cannot Connect\n";

$statement="drop table families\;";
$sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
$rv = $sth->execute;

$statement="drop table family_members\;";
$sth = $dbh->prepare($statement) or die "Can't prepare $statement: $dbh->errstr\n";
$rv = $sth->execute;
```

```
$dbh->disconnect;
print "Done\n\n\n";
open (PROC,"|cat ../sql/families.sql | /sw/arch/bin/mysql -u root -h maine -p$accesspassword CGD");
close PROC;
open (PROC,"|cat ../sql/familymembers.sql | /sw/arch/bin/mysql -u root -h maine -p$accesspassword CGD");
close PROC;
open (PROC2,"|/sw/arch/bin/mysql -u root -h maine -p$accesspassword CGD");

$families=0;
open (FILE,$ARGV[0]);
while (<FILE>)
        {

        if (/^(\d+)\s+(\S+)\s+(.*)/)
                {
                push(@{$familieshash{$1}},$2);
                if ($3)
                        {
                        push(@{$annotations{$1}},uc($3));
                        }
                $count{$1}++;
                }
        }

foreach $cluster (sort numeric(keys(%familieshash)))
        {
        $archea=$bacteria=$eukaryota=$viruses=0;
        $average=0;
        $av_count=0;
        $go1="";
        $go2="";
        foreach $protein (@{$familieshash{$cluster}})
                {
                if ($go1 eq "")
                        {
                        $go1="protein1=\"$protein\"";
                        }
                else
                        {
                        $go1=join(" OR ",$go1,"protein1=\"$protein\"");
                        }

                if ($go2 eq "")
                        {
                        $go2="protein2=\"$protein\"";
                        }
                else
                        {
                        $go2=join(" OR ",$go2,"protein2=\"$protein\"");
                        }
                }

                $statement="select protein1,protein2,percen_identity from similarity
where ($go1) AND ($go2)\n";
                open (FETCH,"echo \'$statement\' | mysql -u cgg -h maine CGD_SIM|");
                while (<FETCH>)
                        {
                        chop($_);
                        @array=split(" ",$_);
                        $protein1=$array[0];
                        $protein2=$array[1];
                        $identity=$array[2];
```

```
                $identity=~s/\%//g;
                if (($protein1 ne "protein1") &&($protein1 ne $protein2))
                        {
                        $average+=$identity;
                        $av_count++;
                        }
                }
        if ($av_count != 0)
                {
                $average=$average/$av_count;
                }
        else
                {
                $average=100;
                }

        foreach $protein (@{$familieshash{$cluster}})
                {
                if ($hash{$protein} eq 'Archaea;')
                        {
                        $archea++;
                        }
                elsif ($hash{$protein} eq 'Bacteria;')
                        {
                        $bacteria++;
                        }
                elsif ($hash{$protein} eq 'Eukaryota;')
                        {
                        $eukaryota++;
                        }
                elsif ($hash{$protein} eq 'Viruses;')
                        {
                        $viruses++;
                        }
                else
                        {
                        print "Error!: $protein -> No Tax Class -> God help us!\n";
                        }
                }


        $families++;
        $family_size=$#{$familieshash{$cluster}}+1;
        $family_id=sprintf("TR-%10.10d",$families);

        print "$family_id\t($family_size members)\t$cluster\n";
        @temp=consensus_annotate(@{$annotations{$cluster}});
        print "$temp[0]\n";

        if ($temp[0])
                {
                $annotation=$temp[0];
                $annotation=~s/\'/ /g;
                $ann_score=$temp[1];
                }
        else
                {
                die "Error: No annotation returned\n";
                }

        $statement="INSERT INTO families VALUES(\"$family_id\",$family_size,\'$annotation\',
$ann_score,\"$average\",\"0.1\",$archea,$bacteria,$eukaryota,$viruses)\;";
```

```
                print PROC2 "$statement\n";


                foreach $protein (@{$familieshash{$cluster}})
                        {
                        $statement="INSERT INTO family_members VALUES(\"$protein\",\"$family_id\")\;";
                        print PROC2 "$statement\n";
                        }

        }
close PROC2;

sub numeric {$a<=>$b}
```

Bibliography

Adams, M. D., Celniker, S. E., Holt, R. A., Evans, C. A., Gocayne, J. D., Amanatides, P. G., Scherer, S. E., Li, P. W., Hoskins, R. A., Galle, R. F., George, R. A., Lewis, S. E., Richards, S., Ashburner, M., Henderson, S. N., Sutton, G. G., Wortman, J. R., Yandell, M. D., Zhang, Q., Chen, L. X., Brandon, R. C., Rogers, Y. H., Blazej, R. G., Champe, M., Pfeiffer, B. D., Wan, K. H., Doyle, C., Baxter, E. G., Helt, G., Nelson, C. R., Gabor, G. L., Abril, J. F., Agbayani, A., An, H. J., Andrews-Pfannkoch, C., Baldwin, D., Ballew, R. M., Basu, A., Baxendale, J., Bayraktaroglu, L., Beasley, E. M., Beeson, K. Y., Benos, P. V., Berman, B. P., Bhandari, D., Bolshakov, S., Borkova, D., Botchan, M. R., Bouck, J., Brokstein, P., Brottier, P., Burtis, K. C., Busam, D. A., Butler, H., Cadieu, E., Center, A., Chandra, I., Cherry, J. M., Cawley, S., Dahlke, C., Davenport, L. B., Davies, P., de Pablos, B., Delcher, A., Deng, Z., Mays, A. D., Dew, I., Dietz, S. M., Dodson, K., Doup, L. E., Downes, M., D (2000). The genome sequence of Drosophila melanogaster. *Science*, 287:2185–2195.

Allen, J. (1994). *Natural language understanding*. Benjamin Cummings, Redwood City, CA.

Altschul, S. F., Boguski, M. S., Gish, W., and Wootton, J. C. (1994). Issues in searching molecular sequence databases. *Nat Genet*, 6:119–129.

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

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

Andrade, M. A. and Bork, P. (2000). Automated extraction of information in molecular biology. *FEBS Lett*, 476:12–17.

Andrade, M. A. and Valencia, A. (1998). Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families. *Bioinformatics*, 14:600–607.

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

Anfinsen, C. B. and Corley, L. G. (1969). An active variant of staphylococcal nuclease containing norleucine in place of methionine. *J Biol Chem*, 244:5149–5152.

Apic, G., Gough, J., and Teichmann, S. A. (2001a). An insight into domain combinations. *Bioinformatics*, pages S83–S89.

Apic, G., Gough, J., and Teichmann, S. A. (2001b). Domain combinations in archaeal, eubacterial and eukaryotic proteomes. *J Mol Biol*, 310:311–325.

Apweiler, R., Attwood, T. K., Bairoch, A., Bateman, A., Birney, E., Biswas, M., Bucher, P., Cerutti, L., Corpet, F., Croning, M. D., Durbin, R., Falquet, L., Fleischmann, W., Gouzy, J., Hermjakob, H., Hulo, N., Jonassen, I., Kahn, D., Kanapin, A., Karavidopoulou, Y., Lopez, R., Marx, B., Mulder, N. J., Oinn, T. M., Pagni, M., and Servant, F. (2001). The InterPro database, an integrated documentation resource for protein families, domains and functional sites. *Nucleic Acids Res*, 29:37–40.

Ashburner, M. . (1993). FlyBase. *Genome News*, 13:19–20.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., and Sherlock, G. (2000). Gene ontology: tool for the unification of biology. *Nat Genet*, 25:25–29.

Attwood, T. K., Flower, D. R., Lewis, A. P., Mabey, J. E., Morgan, S. R., Scordis, P., Selley, J. N., and Wright, W. (1999). PRINTS prepares for the new millennium. *Nucleic Acids Res*, 27:220–225.

Bairoch, A. and Apweiler, R. (2000). The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res*, 28:45–48.

Bateman, A. (1997). The structure of a domain common to archaebacteria and the homocystinuria disease protein. *Trends Biochem Sci*, 22:12–13.

Bateman, A., Birney, E., Durbin, R., Eddy, S. R., Finn, R. D., and Sonnhammer, E. L. (1999). Pfam 3. *Nucleic Acids Res*, 27:260–262.

Bateman, A., Birney, E., Durbin, R., Eddy, S. R., Howe, K. L., and Sonnhammer, E. L. (2000). The Pfam protein families database. *Nucleic Acids Res*, 28:263–266.

Bell, S. D., Cairns, S. S., Robson, R. L., and Jackson, S. P. (1999). Transcriptional regulation of an archaeal operon in vivo and in vitro. *Mol Cell*, 4:971–982.

Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A., and Wheeler, D. L. (2002). GenBank. *Nucleic Acids Res*, 30:17–20.

Bernal, A., Ear, U., and Kyrpides, N. (2001). Genomes OnLine Database (GOLD): a monitor of genome projects world-wide. *Nucleic Acids Res*, 29:126–127.

Birney, E., Bateman, A., Clamp, M. E., and Hubbard, T. J. (2001). Mining the draft human genome. *Nature*, 409:827–828.

Blackstock, W. P. and Weir, M. P. (1999). Proteomics: quantitative and physical mapping of cellular proteins. *Trends Biotechnol*, 17:121–127.

Blaiseau, P. L., Isnard, A. D., Surdin-Kerjan, Y., and Thomas, D. (1997). Met31p and Met32p, two related zinc finger proteins, are involved in transcriptional regulation of yeast sulfur amino acid metabolism. *Mol Cell Biol*, 17:3640–3648.

Blaschke, C., Andrade, M. A., Ouzounis, C., and Valencia, A. (1999). Automatic extraction of biological information from scientific text: protein-protein interactions. *Proc Int Conf Intell Syst Mol Biol*, pages 60–67.

Blencowe, B. J., Bowman, J. A., McCracken, S., and Rosonina, E. (1999). SR-related proteins and the processing of messenger RNA precursors. *Biochem Cell Biol*, 77:277–291.

Blumenthal, T. (1998). Gene clusters and polycistronic transcription in eukaryotes. *Bioessays*, 20:480–487.

Boucher, L., Ouzounis, C. A., Enright, A. J., and Blencowe, B. J. (2001). A genome-wide survey of RS domain proteins. *RNA*, 7:1693–1701.

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

Brown, H., Sanger, F., and Kitai, R. (1955). The structure of pig and sheep insulins. *Biochem. J.*, 60:556–556.

Brown, N. P., Leroy, C., and Sander, C. (1998). MView: a web-compatible database search or multiple alignment viewer. *Bioinformatics*, 14:380–381.

Bulow, L. (1990). Preparation of artificial bifunctional enzymes by gene fusion. *Biochem Soc Symp*, 57:123–133.

Burns, D. M., Horn, V., Paluh, J., and Yanofsky, C. (1990). Evolution of the tryptophan synthetase of fungi. *J Biol Chem*, 265:2060–2069.

Caceres, J. F., Misteli, T., Screaton, G. R., Spector, D. L., and Krainer, A. R. (1997). Role of the modular domains of SR proteins in subnuclear localization and alternative splicing specificity. *J Cell Biol*, 138:225–238.

Carrilho, E., Ruiz-Martinez, M. C., Berka, J., Smirnov, I., Goetzinger, W., Miller, A. W., Brady, D., and Karger, B. L. (1996). Rapid DNA sequencing of more than 1000 bases per run by capillary electrophoresis using replaceable linear polyacrylamide solutions. *Anal Chem*, 68:3305–3313.

Casjens, S. (2000). Borrelia genomes in the year 2000. *J Mol Microbiol Biotechnol*, 2:401–410.

Chang, C. and Meyerowitz, E. M. (2001). Eukaryotes have "two-component" signal transducers. *Res Microbiol*, 145:481–486.

Cho, R. J., Campbell, M. J., Winzeler, E. A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T. G., Gabrielian, A. E., Landsman, D., Lockhart, D. J., and Davis, R. W. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell*, 2:65–73.

Chothia, C. (1992). One thousand families for the molecular biologist. *Nature*, 357:543–544.

Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P. O., and Herskowitz, I. (1998). The transcriptional program of sporulation in budding yeast. *Science*, 282:699–705.

Claverie, J. M. and States, D. J. (1993). Information enhancement methods for large scale sequence analysis. *Comput. Chem.*, 17:191–201.

Corpet, F., Servant, F., Gouzy, J., and Kahn, D. (2000). ProDom and ProDom-CG: tools for protein domain analysis and whole genome comparisons. *Nucleic Acids Res*, 28:267–269.

Coulson, R. M., Enright, A. J., and Ouzounis, C. A. (2001). Transcription-associated protein families are primarily taxon-specific. *Bioinformatics*, 17:95–97.

Dandekar, T., Snel, B., Huynen, M., and Bork, P. (1998). Conservation of gene order: a fingerprint of proteins that physically interact. *Trends Biochem Sci*, 23:324–328.

Dayhoff, M. O. (1976). The origin and evolution of protein superfamilies. *Fed Proc*, 35:2132–2138.

Dayhoff, M. O. (1978). Altas of Protein Sequence and Structure. *Natl. Biomed. Res. Found., Washington DC.*, 5:Suppl. 3–Suppl. 3.

Dembo, A. and Karlin, S. (1991). Strong limit theorems of empirical functionals for large exceedances of partial sums of i. *Annals of Probability*, 19:1737–1755.

Denis-Duphil, M. (1989). Pyrimidine biosynthesis in Saccharomyces cerevisiae: the ura2 cluster gene, its multifunctional enzyme product, and other structural or regulatory genes involved in de novo UMP synthesis. *Biochem Cell Biol*, 67:612–631.

DeRisi, J. L., Iyer, V. R., and Brown, P. O. (1997). Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science*, 278:680–686.

Doolittle, R. F. (1985). The genealogy of some recently evolved vertebrate proteins. *Trends Biochem. Sci.*, 10:233–237.

Doolittle, R. F. (1995). The multiplicity of domains in proteins. *Annu Rev Biochem*, 64:287–314.

Doolittle, R. F. and Bork, P. (1993). Evolutionarily mobile modules in proteins. *Sci Am*, 269:50–56.

Doolittle, W. F. (2000). The nature of the universal ancestor and the evolution of the proteome. *Curr Opin Struct Biol*, 10:355–358.

Dowell, R. D., Jokerst, R. M., Day, A., Eddy, S. R., and Stein, L. (2001). The Distributed Annotation System. *BMC Bioinformatics*, 2:7–7.

Duncan, K., Edwards, R. M., and Coggins, J. R. (1987). The pentafunctional arom enzyme of Saccharomyces cerevisiae is a mosaic of monofunctional domains. *Biochem J*, 246:375–386.

Duncan, K., Edwards, R. M., and Coggins, J. R. (1988). The Saccharomyces cerevisiae ARO1 gene. *FEBS Lett*, 241:83–88.

Dwight, S. S., Harris, M. A., Dolinski, K., Ball, C. A., Binkley, G., Christie, K. R., Fisk, D. G., Issel-Tarver, L., Schroeder, M., Sherlock, G., Sethuraman, A., Weng, S., Botstein, D., and Cherry, J. M. (2002). Saccharomyces Genome Database (SGD) provides secondary gene annotation using the Gene Ontology (GO). *Nucleic Acids Res*, 30:69–72.

Eades, P. (1984). A heuristic for graph drawing. *Congressum Numerantium*, 41:149–160.

Eddy, S. R. (1995). Multiple alignment using hidden Markov models. *Proc Int Conf Intell Syst Mol Biol*, 3:114–120.

Eddy, S. R. (1998). Profile hidden Markov models. *Bioinformatics*, 14:755–763.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A*, 95:14863–14868.

Eisenberg, D., Marcotte, E. M., Xenarios, I., and Yeates, T. O. (2000). Protein function in the post-genomic era. *Nature*, 405:823–826.

Enright, A. J., Iliopoulos, I., Kyrpides, N. C., and Ouzounis, C. A. (1999). Protein interaction maps for complete genomes based on gene fusion events. *Nature*, 402:86–90.

Enright, A. J. and Ouzounis, C. A. (2000). GeneRAGE: a robust algorithm for sequence clustering and domain detection. *Bioinformatics*, 16:451–457.

Enright, A. J. and Ouzounis, C. A. (2001a). BioLayout-an automatic graph layout algorithm for similarity visualization. *Bioinformatics*, 17:853–854.

Enright, A. J. and Ouzounis, C. A. (2001b). Functional associations of proteins in entire genomes via exhaustive detection of gene fusion events. *Genome Biology*, 2:341–347.

Enright, A. J. and Ouzounis, C. A. (2001c). *Protein-protein interactions: a molecular cloning manual*, chapter Computational detection of genome-wide protein interaction, pages 471–500. Cold Spring Harbour Laboratory Press, Cold Spring Harbour, NY.

Enright, A. J., van Dongen, S., and Ouzounis, C. A. (2002). An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 7:–.

Etzold, T. and Argos, P. (1993). SRS--an indexing and retrieval tool for flat file data libraries. *Comput Appl Biosci*, 9:49–57.

Etzold, T., Ulyanov, A., and Argos, P. (1996). SRS: information retrieval system for molecular biology data banks. *Methods Enzymol*, 266:114–128.

Falquet, L., Pagni, M., Bucher, P., Hulo, N., Sigrist, C. J., Hofmann, K., and Bairoch, A. (2002). The PROSITE database, its status in 2002. *Nucleic Acids Res*, 30:235–238.

Fitch, W. M. (1973). Aspects of molecular evolution. *Annu Rev Genet*, 7:343–380.

Fitch, W. M. and Margoliash, E. (1967). Construction of phylogenetic trees. *Science*, 155:279–284.

Fleischmann, R. D., Adams, M. D., White, O., Clayton, R. A., Kirkness, E. F., Kerlavage, A. R., Bult, C. J., Tomb, J. F., Dougherty, B. A., and Merrick, J. M. (1995). Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science*, 269:496–512.

Fruchterman, T. M. and Rheingold, E. M. (1991). Graph drawing by force directed placement. *Softw. Exp. Pract.*, 21:1129–1164.

Fukuda, K., Tamura, A., Tsunoda, T., and Takagi, T. (1998). Toward information extraction: identifying protein names from biological papers. *Pac Symp Biocomput*, pages 707–718.

Galperin, M. Y. and Koonin, E. V. (1999). Searching for drug targets in microbial genomes. *Curr Opin Biotechnol*, 10:571–578.

Galperin, M. Y. and Koonin, E. V. (2000). Who's your neighbor? New computational approaches for functional genomics. *Nat Biotechnol*, 18:609–613.

Gonnet, G. H., Cohen, M. A., and Benner, S. A. (1992). Exhaustive matching of the entire protein sequence database. *Science*, 256:1443–1445.

Gotoh, O. (1982). An improved algorithm for matching biological sequences. *J Mol Biol*, 162:705–708.

Gracy, J. and Argos, P. (1998a). Automated protein sequence database classification. *Bioinformatics*, 14:174–187.

Gracy, J. and Argos, P. (1998b). Automated protein sequence database classification. *Bioinformatics*, 14:164–173.

Gray, M. W. (1993). Origin and evolution of organelle genomes. *Curr Opin Genet Dev*, 3:884–890.

Gribskov, M., McLachlan, A. D., and Eisenberg, D. (1987). Profile analysis: detection of distantly related proteins. *Proc Natl Acad Sci U S A*, 84:4355–4358.

Gumbel, E. (1958). *Statistics of Extremes.* Columbia University Press, New York, NY.

Gusfield, D. (1997). *Algorithms on strings, trees and sequences*. Cambridge University Press, Cambridge, UK.

Haeckel, E. (1874). *Anthropogenie oder Entwickelungsgeschichte des Menschen.* Engelmann, Leipzig.

Heger, A. and Holm, L. (2000). Towards a covering set of protein family profiles. *Prog Biophys Mol Biol*, 73:321–337.

Hegyi, H. and Bork, P. (1997). On the classification and evolution of protein modules. *J Protein Chem*, 16:545–551.

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

Helliwell, S. B., Howald, I., Barbet, N., and Hall, M. N. (1998). TOR2 is part of two related signaling pathways coordinating cell growth in Saccharomyces cerevisiae. *Genetics*, 148:99–112.

Henikoff, S., Greene, E. A., Pietrokovski, S., Bork, P., Attwood, T. K., and Hood, L. (1997). Gene families: the taxonomy of protein paralogs and chimeras. *Science*, 278:609–614.

Henikoff, S. and Henikoff, J. G. (1991). Automated assembly of protein blocks for database searching. *Nucleic Acids Res*, 19:6565–6572.

Henikoff, S. and Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A*, 89:10915–10919.

Himmelreich, R., Plagens, H., Hilbert, H., Reiner, B., and Herrmann, R. (1997). Comparative analysis of the genomes of the bacteria Mycoplasma pneumoniae and Mycoplasma genitalium. *Nucleic Acids Res*, 25:701–712.

Hirschberg, D. S. (1975). A linear space algorithm for computing maximal common subsequences. *Comm. A.C.M.*, 18:341–343.

Hirschberg, D. S. (1977). Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24:664–675.

Holm, L. and Sander, C. (1996). Mapping the protein universe. *Science*, 273:595–603.

Holm, L. and Sander, C. (1997). New structure--novel fold? *Structure*, 5:165–171.

Holm, L. and Sander, C. (1998). Removing near-neighbour redundancy from large protein sequence collections. *Bioinformatics*, 14:423–429.

Hubbard, T., Barker, D., Birney, E., Cameron, G., Chen, Y., Clark, L., Cox, T., Cuff, J., Curwen, V., Down, T., Durbin, R., Eyras, E., Gilbert, J., Hammond, M., Huminiecki, L., Kasprzyk, A., Lehvaslaiho, H., Lijnzaad, P., Melsopp, C., Mongin, E., Pettett, R., Pocock, M., Potter, S., Rust, A., Schmidt, E., Searle, S., Slater, G., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Stupka, E., Ureta-Vidal, A., Vastrik, I., and Clamp, M. (2002). The Ensembl genome database project. *Nucleic Acids Res*, 30:38–41.

Huynen, M., Snel, B., Lathe, W., and Bork, P. (2000). Exploitation of gene context. *Curr Opin Struct Biol*, 10:366–370.

Huynen, M. A. and Bork, P. (1998). Measuring genome evolution. *Proc Natl Acad Sci U S A*, 95:5849–5856.

Iliopoulos, I., Enright, A. J., and Ouzounis, C. A. (2001a). Textquest: document clustering of Medline abstracts for concept discovery in molecular biology. *Pac Symp Biocomput*, pages 384–395.

Iliopoulos, I., Tsoka, S., Andrade, M. A., Janssen, P., Audit, B., Tramontano, A., Valencia, A., Leroy, C., Sander, C., and Ouzounis, C. A. (2001b). Genome sequences and great expectations. *Genome Biol*, 2:INTERACTIONS0001–INTERACTIONS0001.

Ito, T., Tashiro, K., Muta, S., Ozawa, R., Chiba, T., Nishizawa, M., Yamamoto, K., Kuhara, S., and Sakaki, Y. (2000). Toward a protein-protein interaction map of the budding yeast: A comprehensive system to examine two-hybrid interactions in all possible combinations between the yeast proteins. *Proc Natl Acad Sci U S A*, 97:1143–1147.

Janin, J., Miller, S., and Chothia, C. (1988). Surface, subunit interfaces and interior of oligomeric proteins. *J Mol Biol*, 204:155–164.

Janssen, P. J., Audit, B., and Ouzounis, C. A. (2001). Strain-specific genes of Helicobacter pylori: distribution, function and dynamics. *Nucleic Acids Res*, 29:4395–4404.

Jones, S. and Thornton, J. M. (1995). Protein-protein interactions: a review of protein dimer structures. *Prog Biophys Mol Biol*, 63:31–65.

Jones, S. and Thornton, J. M. (1996). Principles of protein-protein interactions. *Proc Natl Acad Sci U S A*, 93:13–20.

Kamada, T. and Kawai, S. (1989). An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15.

Kanehisa, M. and Goto, S. (2000). KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res*, 28:27–30.

Karlin, S. and Altschul, S. F. (1990). Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc Natl Acad Sci U S A*, 87:2264–2268.

Kawarabayasi, Y., Hino, Y., Horikawa, H., Yamazaki, S., Haikawa, Y., Jin-no, K., Takahashi, M., Sekine, M., Baba, S., Ankai, A., Kosugi, H., Hosoyama, A., Fukui, S., Nagai, Y., Nishijima, K., Nakazawa, H., Takamiya, M., Masuda, S., Funahashi, T., Tanaka, T., Kudoh, Y., Yamazaki, J., Kushida, N., Oguchi, A., and Kikuchi, H. (1999). Complete genome sequence of an aerobic hyper-thermophilic crenarchaeon, Aeropyrum pernix K1. *DNA Res*, 6:83–101, 145.

Koonin, E. V. (1997). Evidence for a family of archaeal ATPases. *Science*, 275:1489–1490.

Koonin, E. V., Aravind, L., and Kondrashov, A. S. (2000). The impact of comparative genomics on our understanding of evolution. *Cell*, 101:573–576.

Koonin, E. V., Mushegian, A. R., and Bork, P. (1996). Non-orthologous gene displacement. *Trends Genet*, 12:334–336.

Kraulis, P. J. (1991). Molscript: A program to produce both detailed and schematic plots of protein structures. *J. Appl. Cryst.*, 24:946–950.

Krogh, A., Brown, M., Mian, I. S., Sjlander, K., and Haussler, D. (1994). Hidden Markov models in computational biology. *J Mol Biol*, 235:1501–1531.

Kyrpides, N., Overbeek, R., and Ouzounis, C. (1999). Universal protein families and the functional content of the last universal common ancestor. *J Mol Evol*, 49:413–423.

Kyrpides, N. C., Olsen, G. J., Klenk, H. P., White, O., and Woese, C. R. (1996). Methanococcus jannaschii genome: revisited. *Microb Comp Genomics*, 1:329–338.

Kyrpides, N. C. and Ouzounis, C. A. (1998). Errors in genome reviews. *Science*, 281:1457–1457.

Kyrpides, N. C. and Ouzounis, C. A. (1999). Transcription in archaea. *Proc Natl Acad Sci U S A*, 96:8545–8550.

Kyrpides, N. C. and Woese, C. R. (1998). Tetratrico-peptide-repeat proteins in the archaeon Methanococcus jannaschii. *Trends Biochem Sci*, 23:245–247.

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., Funke, R., Gage, D., Harris, K., Heaford, A., Howland, J., Kann, L., Lehoczky, J., LeVine, R., McEwan, P., McKernan, K., Meldrim, J., Mesirov, J. P., Miranda, C., Morris, W., Naylor, J., Raymond, C., Rosetti, M., Santos, R., Sheridan, A., Sougnez, C., Stange-Thomann, N., Stojanovic, N., Subramanian, A., Wyman, D., Rogers, J., Sulston, J., Ainscough, R., Beck, S., Bentley, D., Burton, J., Clee, C., Carter, N., Coulson, A., Deadman, R., Deloukas, P., Dunham, A., Dunham, I., Durbin, R., French, L., Grafham, D., Gregory, S., Hubbard, T., Humphray, S., Hunt, A., Jones, M., Lloyd, C., McMurray, A., Matthews, L., Mercer, S., Milne, S., Mullikin, J. C., Mungall, A., Plumb, R., Ross, M., Shownkeen, R., Sims, S., Waterston, R. H., Wilson, R. K., Hillier, L. W., McPherson, J. D., Marra, M. A., Mardis, E. R., Fulton, L. A., Chinwalla, A. T., Pepin, K. H., Gish, W. R., Chi (2001). Initial sequencing and analysis of the human genome. *Nature*, 409:860–921.

Latchman, D. S. (1997). How can we use our growing understanding of gene transcription to discover effective new medicines? *Curr Opin Biotechnol*, 8:713–717.

Lawrence, M. C. and Colman, P. M. (1993). Shape complementarity at protein/protein interfaces. *J Mol Biol*, 234:946–950.

Lee, B. and Richards, F. M. (1971). The interpretation of protein structures: Estimation of static accessibility. *J. Mol. Biol.*, 55:379–400.

Lescure, A., Gautheret, D., Carbon, P., and Krol, A. (1999). Novel selenoproteins identified in silico and in vivo by using a conserved RNA structural motif. *J Biol Chem*, 274:38147–38154.

Lim, A. L. and Powers-Lee, S. G. (1996). Requirement for the carboxyl-terminal domain of Saccharomyces cerevisiae carbamoyl-phosphate synthetase. *J Biol Chem*, 271:11400–11409.

Linial, M., Linial, N., Tishby, N., and Yona, G. (1997). Global self-organization of all known protein sequences reveals inherent biological signatures. *J Mol Biol*, 268:539–556.

Linnaeus, C. V. (1735). *Systema Naturae*.

Lipman, D. J. and Pearson, W. R. (1985). Rapid and sensitive protein similarity searches. *Science*, 227:1435–1441.

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 Res*, 28:257–259.

MacBeath, G. and Schreiber, S. L. (2000). Printing proteins as microarrays for high-throughput function determination. *Science*, 289:1760–1763.

Marcotte, E. M., Pellegrini, M., Ng, H. L., Rice, D. W., Yeates, T. O., and Eisenberg, D. (1999a). Detecting protein function and protein-protein interactions from genome sequences. *Science*, 285:751–753.

Marcotte, E. M., Pellegrini, M., Thompson, M. J., Yeates, T. O., and Eisenberg, D. (1999b). A combined algorithm for genome-wide prediction of protein function. *Nature*, 402:83–86.

McGarvey, P. B., Huang, H., Barker, W. C., Orcutt, B. C., Garavelli, J. S., Srinivasarao, G. Y., Yeh, L. S., Xiao, C., and Wu, C. H. (2000). PIR: a new resource for bioinformatics. *Bioinformatics*, 16:290–291.

McGinnis, W., Garber, R. L., Wirz, J., Kuroiwa, A., and Gehring, W. J. (1984). A homologous protein-coding sequence in Drosophila homeotic genes and its conservation in other metazoans. *Cell*, 37:403–408.

Mellor, J. C., Yanai, I., Clodfelter, K. H., Mintseris, J., and DeLisi, C. (2002). Predictome: a database of putative functional links between proteins. *Nucleic Acids Res*, 30:306–309.

Mendelsohn, A. R. and Brent, R. (1999). Protein interaction methods--toward an endgame. *Science*, 284:1948–1950.

Merritt, E. A. and Murphy, M. E. P. (1994). Raster3D Version 2. *Acta. Cryst.*, 50:869–873.

Mewes, H. W., Albermann, K., Bhr, M., Frishman, D., Gleissner, A., Hani, J., Heumann, K., Kleine, K., Maierl, A., Oliver, S. G., Pfeiffer, F., Zollner, A., Mewes, H. W., Albermann, K., Bhr, M., Frishman, D., Gleissner, A., Hani, J., Heumann, K., Kleine, K., Maierl, A., Oliver, S. G., Pfeiffer, F., and Zollner, A. (1997). Overview of the yeast genome. *Nature*, 387:7–65.

Michalski, R., Bratko, I., and Kubat, M. (1998). *Machine learning and data mining*. Wiley.

Mizuta, T. R., Fukita, Y., Miyoshi, T., Shimizu, A., and Honjo, T. (1993). Isolation of cDNA encoding a binding protein specific to 5'-phosphorylated single-stranded DNA with G-rich sequences. *Nucleic Acids Res*, 21:1761–1766.

Moskovitz, J., Berlett, B. S., Poston, J. M., and Stadtman, E. R. (1997). The yeast peptide-methionine sulfoxide reductase functions as an antioxidant in vivo. *Proc Natl Acad Sci U S A*, 94:9585–9589.

Mott, R. (1992). Maximum-likelihood estimation of the statistical distribution of smith-waterman local sequence similarity scores. *Bull. Math. Biol.*, 54:59–75.

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. *J Mol Biol*, 48:443–453.

Nuttall, G. (1904). *Blood Immunity and Blood Relationship*. Cambridge University Press, Cambridge, UK.

Ohno, S. (1970). *Evolution by gene duplication*. Springer-Verlag, Berlin.

Olsen, G. J., Woese, C. R., and Woese, C. R. (1997). Archaeal genomics: an overview. *Cell*, 89:991–994.

Orengo, C. A., Bray, J. E., Buchan, D. W., Harrison, A., Lee, D., Pearl, F. M., Sillitoe, I., Todd, A. E., and Thornton, J. M. (2002). The CATH protein family database: A resource for structural and functional annotation of genomes. *Proteomics*, 2:11–21.

Ouzounis, C., Casari, G., Sander, C., Tamames, J., and Valencia, A. (1996). Computational comparisons of model genomes. *Trends Biotechnol*, 14:280–285.

Ouzounis, C. and Kyrpides, N. (1996). The emergence of major cellular processes in evolution. *FEBS Lett*, 390:119–123.

Ouzounis, C., Valencia, A., Tamames, J., Bork, P., and Sander, C. (1995). *3rd European Conference on Artificial Life*, chapter The functional composition of living machines as a design principle for artificial organisms., pages 843–851. Springer, Granada, Spain.

Ouzounis, C. A. and Karp, P. D. (2000). Global properties of the metabolic map of Escherichia coli. *Genome Res*, 10:568–576.

Overbeek, R., Fonstein, M., D'Souza, M., Pusch, G. D., and Maltsev, N. (1999). The use of gene clusters to infer functional coupling. *Proc Natl Acad Sci U S A*, 96:2896–2901.

Park, J. and Teichmann, S. A. (1998). DIVCLUS: an automatic method in the GEANFAMMER package that finds homologous domains in single- and multi-domain proteins. *Bioinformatics*, 14:144–150.

Park, J., Teichmann, S. A., Hubbard, T., and Chothia, C. (1997). Intermediate sequences increase the detection of homology between sequences. *J Mol Biol*, 273:349–354.

Patthy, L. (1985). Evolution of the proteases of blood coagulation and fibrinolysis by assembly from modules. *Cell*, 41:657–663.

Pearson, W. R. (1990). Rapid and sensitive sequence comparison with FASTP and FASTA. *Methods Enzymol*, 183:63–98.

Pearson, W. R. (1996). Effective protein sequence comparison. *Methods Enzymol*, 266:227–258.

Pearson, W. R. and Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A*, 85:2444–2448.

Pearson, W. R. and Miller, W. (1992). Dynamic programming algorithms for biological sequence comparison. *Methods Enzymol*, 210:575–601.

Pellegrini, M., Marcotte, E. M., Thompson, M. J., Eisenberg, D., and Yeates, T. O. (1999). Assigning protein functions by comparative genome analysis: protein phylogenetic profiles. *Proc Natl Acad Sci U S A*, 96:4285–4288.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14:130–137.

Promponas, V. J., Enright, A. J., Tsoka, S., Kreil, D. P., Leroy, C., Hamodrakas, S., Sander, C., and Ouzounis, C. A. (2000). CAST: an iterative algorithm for the complexity analysis of sequence tracts. *Bioinformatics*, 16:915–922.

Retief, J. D. (2000). Phylogenetic analysis using PHYLIP. *Methods Mol Biol*, 132:243–258.

Riechmann, J. L., Heard, J., Martin, G., Reuber, L., Jiang, C., Keddie, J., Adam, L., Pineda, O., Ratcliffe, O. J., Samaha, R. R., Creelman, R., Pilgrim, M., Broun, P., Zhang, J. Z., Ghandehari, D., Sherman, B. K., Yu, G., Riechmann, J. L., Heard, J., Martin, G., Reuber, L., Jiang, C., Keddie, J., Adam, L., Pineda, O., Ratcliffe, O. J., Samaha, R. R., Creelman, R., Pilgrim, M., and Broun, P. (2000). Arabidopsis transcription factors: genome-wide comparative analysis among eukaryotes. *Science*, 290:2105–2110.

Rivera, M. C., Jain, R., Moore, J. E., and Lake, J. A. (1998). Genomic evidence for two functionally distinct gene classes. *Proc Natl Acad Sci U S A*, 95:6239–6244.

Robison, K., Gilbert, W., and Church, G. M. (1994). Large-scale bacterial gene discovery by similarity search. *Nature Genet.*, 7:205–214.

Rocchio, J. (1971). *The SMART retrieval system*, chapter Relevance feedback information retrieval, page 313. Prentice-Hall, Englewood Cliffs, NJ.

Rongo, C., Broihier, H. T., Moore, L., Doren, M. V., Forbes, A., and Lehmann, R. (1997). Germ plasm assembly and germ cell migration in Drosophila. *Cold Spring Harb Symp Quant Biol*, 62:1–11.

Ross, C. M., Kaplan, J. B., Winkler, M. E., and Nichols, B. P. (1990). An evolutionary comparison of Acinetobacter calcoaceticus trpF with trpF genes of several organisms. *Mol Biol Evol*, 7:74–81.

Rossman, M. G., Moras, D., and Olsen, K. W. (1974). Chemical and biological evolution of nucleotide binding proteins. *Nature*, 250:194–199.

Rubin, G. M., Yandell, M. D., Wortman, J. R., Miklos, G. L. G., Nelson, C. R., Hariharan, I. K., Fortini, M. E., Li, P. W., Apweiler, R., Fleischmann, W., Cherry, J. M., Henikoff, S., Skupski, M. P., Misra, S., Ashburner, M., Birney, E., Boguski, M. S., Brody, T., Brokstein, P., Celniker, S. E., Chervitz, S. A., Coates, D., Cravchik, A., Gabrielian, A., Galle, R. F., Gelbart, W. M., George, R. A., Goldstein, L. S., Gong, F., Guan, P., Harris, N. L., Hay, B. A., Hoskins, R. A., Li, J., Li, Z., Hynes, R. O., Jones, S. J., Kuehl, P. M., Lemaitre, B., Littleton, J. T., Morrison, D. K., Mungall, C., O'Farrell, P. H., Pickeral, O. K., Shue, C., Vosshall, L. B., Zhang, J., Zhao, Q., Zheng, X. H., and Lewis, S. (2000). Comparative genomics of the eukaryotes. *Science*, 287:2204–2215.

Sali, A. (1999). Functional links between proteins. *Nature*, 402:23, 25–23, 26.

Salton, G. (1970). Automatic text analysis. *Science*, 168:335–343.

Sanger, F. (1981). Determination of nucleotide sequences in DNA. *Science*, 214:1205–1210.

Sanger, F. and Coulson, A. R. (1975). A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J Mol Biol*, 94:441–448.

Scharf, M., Schneider, R., Casari, G., Bork, P., Valencia, A., Ouzounis, C., and Sander, C. (1994). GeneQuiz: a workbench for sequence analysis. *Proc Int Conf Intell Syst Mol Biol*, 2:348–353.

Schena, M., Shalon, D., Davis, R. W., and Brown, P. O. (1995). Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470.

Schultz, J., Copley, R. R., Doerks, T., Ponting, C. P., and Bork, P. (2000). SMART: a web-based tool for the study of genetically mobile domains. *Nucleic Acids Res*, 28:231–234.

Schultz, J., Milpetz, F., Bork, P., and Ponting, C. P. (1998). SMART, a simple modular architecture research tool: identification of signaling domains. *Proc Natl Acad Sci U S A*, 95:5857–5864.

Scott, M. P. and Weiner, A. J. (1984). Structural relationships among genes that control development: sequence homology between the Antennapedia, Ultrabithorax, and fushi tarazu loci of Drosophila. *Proc Natl Acad Sci U S A*, 81:4115–4119.

Sellers, P. H. (1974). On the theory and computation of evolutionary distances. *SIAM J. Appl. Math.*, 26:787–793.

Shigenobu, S., Watanabe, H., Hattori, M., Sakaki, Y., and Ishikawa, H. (2000). Genome sequence of the endocellular bacterial symbiont of aphids Buchnera sp. *Nature*, 407:81–86.

Smith, S. (1994). The animal fatty acid synthase: one gene, one polypeptide, seven enzymes. *FASEB J*, 8:1248–1259.

Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *J Mol Biol*, 147:195–197.

Smith, T. F. and Zhang, X. (1997). The challenges of genome sequence annotation or: the devil is in the details. *Nat Biotechnol*, 15:1222–1223.

Snel, B., Bork, P., and Huynen, M. A. (1999). Genome phylogeny based on gene content. *Nat Genet*, 21:108–110.

Sonnhammer, E. L., Eddy, S. R., Birney, E., Bateman, A., and Durbin, R. (1998). Pfam: multiple sequence alignments and HMM-profiles of protein domains. *Nucleic Acids Res*, 26:320–322.

Sonnhammer, E. L. and Kahn, D. (1994). Modular arrangement of proteins as inferred from analysis of homology. *Protein Sci*, 3:482–492.

Stock, A. M., Robinson, V. L., and Goudreau, P. N. (2000). Two-component signal transduction. *Annu Rev Biochem*, 69:183–215.

Stoesser, G., Baker, W., van den Broek, A., Camon, E., Garcia-Pastor, M., Kanz, C., Kulikova, T., Leinonen, R., Lin, Q., Lombard, V., Lopez, R., Redaschi, N., Stoehr, P., Tuli, M. A., Tzouvara, K., and Vaughan, R. (2002). The EMBL Nucleotide Sequence Database. *Nucleic Acids Res*, 30:21–26.

Stoldt, V., Rademacher, F., Kehren, V., Ernst, J. F., Pearce, D. A., and Sherman, F. (1996). Review: the Cct eukaryotic chaperonin subunits of Saccharomyces cerevisiae and other yeasts. *Yeast*, 12:523–529.

Struhl, K. (1999). Fundamentally different logic of gene regulation in eukaryotes and prokaryotes. *Cell*, 98:1–4.

Sussman, J. L., Lin, D., Jiang, J., Manning, N. O., Prilusky, J., Ritter, O., and Abola, E. E. (1998). Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules. *Acta Crystallogr D Biol Crystallogr*, 54:1078–1084.

Syvanen, M. (1984). Conserved regions in mammalian beta-globins: could they arise by cross-species gene exchange? *J Theor Biol*, 107:685–696.

Tan, S. and Richmond, T. J. (1998). Eukaryotic transcription factors. *Curr Opin Struct Biol*, 8:41–48.

Tateno, Y., Imanishi, T., Miyazaki, S., Fukami-Kobayashi, K., Saitou, N., Sugawara, H., and Gojobori, T. (2002). DNA Data Bank of Japan (DDBJ) for genome scale research in life science. *Nucleic Acids Res*, 30:27–30.

Tatusov, R. L., Koonin, E. V., and Lipman, D. J. (1997). A genomic perspective on protein families. *Science*, 278:631–637.

Thomas, J., Milward, D., Ouzounis, C., Pulman, S., and Carroll, M. (2000). Automatic extraction of protein interactions from scientific abstracts. *Pac Symp Biocomput*, pages 541–552.

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

Tsoka, S. and Ouzounis, C. A. (2000a). Prediction of protein interactions: metabolic enzymes are frequently involved in gene fusion. *Nat Genet*, 26:141–142.

Tsoka, S. and Ouzounis, C. A. (2000b). Recent developments and future directions in computational genomics. *FEBS Lett*, 480:42–48.

Uetz, P., Giot, L., Cagney, G., Mansfield, T. A., Judson, R. S., Knight, J. R., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B., Conover, D., Kalbfleisch, T., Vijayadamodar, G., Yang, M., Johnston, M., Fields, S., and Rothberg, J. M. (2000). A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature*, 403:623–627.

van Dongen, S. (2000a). *Graph clustering by flow simulation.* Ph.D. Thesis, University of Utrecht, The Netherlands.

van Dongen, S. (2000b). *A new cluster algorithm for graphs.* Center for Mathematics and Computer Science (CWI), Amsterdam, Report No. INS-R0010.

van Dongen, S. (2000c). *Performance criteria for graph clustering and Markov clustering experiments.* Center for Mathematics and Computer Science (CWI), Amsterdam, Report No. INS-R0012.

van Dongen, S. (2000d). *A stochastic uncoupling process for graphs.* Center for Mathematics and Computer Science (CWI), Amsterdam, Report No. INS-R0011.

van Eeden, F. and Johnston, D. S. (1999). The polarisation of the anterior-posterior and dorsal-ventral axes during Drosophila oogenesis. *Curr Opin Genet Dev*, 9:396–404.

Venter, J. C., Smith, H. O., and Hood, L. (1996). A new strategy for genome sequencing. *Nature*, 381:364–366.

Voorhees, E. M. (1998). *WordNet: an electronic lexical database*, chapter Using WordNet for text retrieval, page 285. MIT Press, Cambridge, MA.

Wales, M. E. and Wild, J. R. (1991). Analysis of structure-function relationships by formation of chimeric enzymes produced by gene fusion. *Methods Enzymol*, 202:687–706.

Ward, J. M. (2001). Identification of novel families of membrane proteins from the model plant Arabidopsis thaliana. *Bioinformatics*, 17:560–563.

Welch, G. R. and Easterby, J. S. (1994). Metabolic channeling versus free diffusion: transition-time analysis. *Trends Biochem Sci*, 19:193–197.

Wilbur, W. J. and Yang, Y. (1996). An analysis of statistical term strength and its use in the indexing and retrieval of molecular biology texts. *Comput. Biol. Med.*, 26:209–209.

Willet, R. (1988). Recent trends in heirarchic document clustering: a critical review. *Information Processing and Management*, 25:577–577.

Woese, C. R. and Fox, G. E. (1977). Phylogenetic structure of the prokaryotic domain: the primary kingdoms. *Proc Natl Acad Sci U S A*, 74:5088–5090.

Woese, C. R., Magrum, L. J., and Fox, G. E. (1978). Archaebacteria. *J Mol Evol*, 11:245–251.

Wolf, Y. I., Grishin, N. V., and Koonin, E. V. (2000). Estimating the number of protein folds and families from complete genome data. *J Mol Biol*, 299:897–905.

Wolfe, K. H. and Shields, D. C. (1997). Molecular evidence for an ancient duplication of the entire yeast genome. *Nature*, 387:708–713.

Wootton, J. C. (1994). Sequences with unusual amino acid compositions. *Curr. Opin. Struct. Biol.*, 4:413–421.

Wootton, J. C. and Federhen, S. (1993). Statistics of local complexity in amino acid sequences and sequence databases. *Comput. Chem.*, 17:149–163.

Xenarios, I., Salwanski, L., Duan, X. J., Higney, P., Kim, S. M., and Eisenberg, D. (2002). DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions. *Nucleic Acids Res*, 30:303–305.

Yanai, I., Derti, A., and DeLisi, C. (2001). Genes linked by fusion events are generally of the same functional category: a systematic analysis of 30 microbial genomes. *Proc Natl Acad Sci U S A*, 98:7940–7945.

Yeh, K. C., Wu, S. H., Murphy, J. T., and Lagarias, J. C. (1997). A cyanobacterial phytochrome two-component light sensory system. *Science*, 277:1505–1508.

Zhang, J. J., Zhao, Y., Chait, B. T., Lathem, W. W., Ritzi, M., Knippers, R., and Darnell, J. E. (1998). Ser727-dependent recruitment of MCM5 by Stat1alpha in IFN-gamma-induced transcriptional activation. *EMBO J*, 17:6963–6971.

Zuckerkandl, E., Jones, R., and Pauling, L. (1965). *Evolving Genes and Proteins*, chapter Evolutionary divergence and convergence in proteins., pages 97–166. Academic Press, New York.

Zuckerkandl, E. and Pauling, L. (1962). *Horizons in Biochemistry*, chapter Molecular disease, evolution and genic heterogeneity., pages 189–225. Academic Press, New York.