Regulatory Motif Discovery in Magnetic Bacteria

Alastair M. Kilpatrick

Master of Science

School of Informatics

University of Edinburgh

2009

Abstract

This project investigates a number of algorithms for discovering regulatory motifs in the DNA of magnetic bacteria. The most promising of these is then implemented in Java and evaluated using a combination of synthetic data and previously characterised real-world data. The implemented algorithm is also used to study three previously uncharacterised datasets; a number of new potential motifs are identified from these datasets. Two extensions to the implemented algorithm are designed, implemented and evaluated.

Acknowledgements

Many thanks to my supervisors Dr. Stuart Aitken and Dr. Bruce Ward, for their valuable advice, support and encouragement. Also for answering my many questions, a few of which they had to answer more than once!

Thanks also to my parents and partner Elaine, for their love and support.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Alastair M. Kilpatrick*)

Table of Contents

Chapter 1

Introduction

1.1 Project Background

This research project is concerned with magnetic bacteria, members of a class of bacteria known as alphaproteobacteria. These bacteria are able to produce magnetic particles within their cells. The principle goal of the project is to investigate a number of algorithms for discovering regulatory motifs in the DNA of magnetic bacteria (we are primarily concerned with *Magnetospirillum magneticum* sp. strain AMB-1 bacteria). The most promising of these algorithms will be implemented using Java; this implementation will then be used to identify the upstream DNA sequences important in the production of magnetic particles. The implemented algorithm will also be evaluated.

This project is important for a number of reasons. Firstly, studies of regulatory sequences in bacteria are important in researching bacteria and the diseases caused by them (Lan, 2008). Although the AMB-1 species is well regarded by scientists for its lack of pathogenic activity, many similar Gram-negative bacteria are pathogenic, meaning they can cause disease. The implemented algorithm will potentially improve motif discovery in upstream DNA sequences; besides the AMB-1 species used in this project, it will be possible to apply the implemented algorithm to discover other regulatory sequences. This means that the implemented algorithm will have wider uses than simply the discovery of regulatory sequences for magnetite particle production. In the long term, it is also possible that the studying of the regulatory sequences in magnetic bacteria will have some commercial applications.

1.2 Magnetotactic Bacteria and *M. magneticum* sp. strain AMB-1

Magnetotactic bacteria are a class of bacteria which are unique in their capability to orient themselves along magnetic field lines (a process known as magnetotaxis) through the production of bacterial magnetic particles. The first peer-reviewed article on magnetotactic bacteria was published in 1975 and contained this observation, which would become one of the hallmarks of the *Magnetospirillum* genus. Although once considered a novelty, magnetotactic bacteria have been found to contribute to the global iron cycle by acquiring iron and converting it into particles of magnetite (an iron oxide, Fe_3O_4) or greigite (an iron sulphide, Fe_3S_4), which accumulate in intracellular structures known as magnetosomes (Matsunaga, *et al.*, 2005). Magnetotaxis is thought to occur because the magnetosome structures align themselves in chains within the bacterium; these chains act as a form of biological compass needle (Arakaki, *et al.*, 2008). Magnetosomes are enveloped in an organic lipid membrane; this allows them to be dispersed easily and evenly in aqueous solutions in comparison to artificial magnetites, making them ideal materials for biotechnological applications. A number of potential applications for magnetosomes have been suggested, including the discovery of proteins, genetic material and other biomolecules in the field of drug discovery and biomedicine (Saiyed, *et al*., 2003; Lan, 2008). It has also been proposed that magnetotactic bacteria could be used as micro-energy sources in nanotechnology applications (Leahy, 2006). One of the major problems in using magnetosomes commercially is the difficulty and cost of cultivating magnetic bacteria. By identifying the key genome sequences which control the production of magnetosomes, this genetic information could be introduced into bacteria which could be cultivated much more cheaply (e.g. *Escherichia coli*); this would make the use of magnetosomes more commercially viable.

Magnetospirillum magneticum sp. strain AMB-1 is a Gram-negative alphaproteobacterium which was first isolated in Tokyo, Japan. AMB-1 has been the subject of a large number of studies and is one of the better understood magnetotactic bacteria. Genome sequencing of AMB-1 was completed in 2005 by the Tokyo University of Agriculture and Technology, Japan (Matsunaga, *et al.*, 2005).

1.3 Gene Regulation Basics

In order to explain what is meant by a motif (or 'regulatory sequence'), we must consider the genome as a whole. The genome of an organism can be regarded as a collection of genes which together specify everything about that particular species. The total number of genes varies widely between species; the AMB-1 genome has 4,559 genes, while the human genome has around 20,000. We can consider the set of genes within a genome to be similar to a set of beads on a string. This is shown in Figure 1.1. (In practice, this is not quite the case and genes may sometimes overlap within the genome; however, our simple example is sufficient for now.)

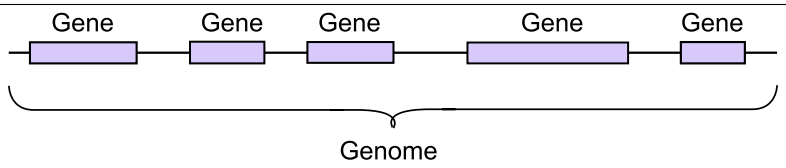

Figure 1.1: Genes within a genome

Consider now a single gene from within the genome. The function of most genes is to create a protein for some defined purpose. This is performed in two steps, transcription and translation. In the transcription step, the genetic code contained in the gene is transcribed into messenger RNA (mRNA). This mRNA is then translated to produce a protein, which provides a specific metabolic function. Figure 1.2 shows the transcription and translation process. Not every gene is transcribed (or switched on) at the same time; in order to activate a transcription, a regulation protein binds in the upstream region of the gene. By switching particular genes on and off, an organism can respond to different stresses or environments. Figure 1.3 illustrates a regulation protein binding upstream of a gene.

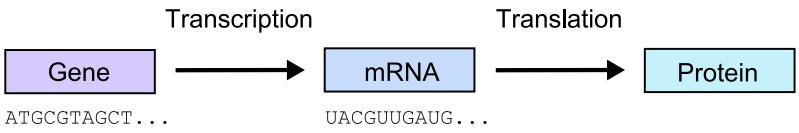

Figure 1.2: The transcription and translation process

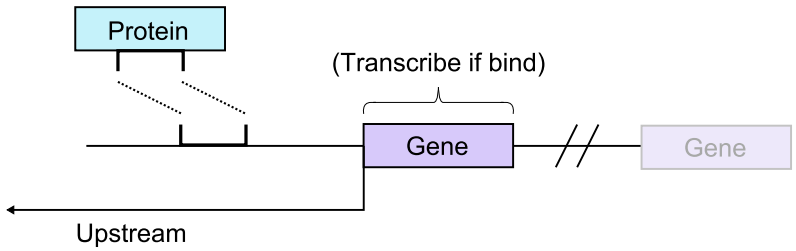

Figure 1.3: A regulation protein binding upstream of a gene, activating the transcription process

The upstream region is the non-coding DNA sequence directly preceding the start of the gene (in our analogy, this is the string just before the bead). In this research project, we regard the upstream region as being 200bp long. That is, for each gene, we consider the upstream region for that gene to be the 200 base pairs before the start codon of the gene. There is no rule defining the length of the upstream region; however, we expect that regulatory proteins usually bind between around 10 and 40 base pairs before the start codon of the gene (there are a number of exceptions to this 'rule of thumb'). Defining the upstream region to be the 200 base pairs before the start codon therefore ensures that we have the best chance of finding the regulation protein binding site.

If we have a number of genes which have a similar product, we expect that the upstream regulation protein binding sites for these genes should be reasonably similar (although likely subject to a small number of natural mutations) in terms of both pattern and length; we call this conserved binding site a 'motif'. It is these motifs which motivate much of the rest of the work in this research project. If we know the motif which activates the production of magnetosome particles, we can 'switch on' these genes whenever we like – the task in this project therefore is to discover these motifs. The prediction of regulatory motifs is an important task in the larger challenge of understanding the mechanisms that control the expression of genes. However, computational methods have offered some hope in this area and computational biologists have invested considerable effort in this area (Tompa, *et al.*, 2005).

Although the explanation of the regulation process given above seems relatively straightforward, in practice, there are a number of complications which make understanding of the regulation mechanism a considerable problem:

- Sometimes, more than one protein is required to activate the transcription for a single gene (see Figure 1.4), or the presence of another protein elsewhere up-

stream will block the transcription for a gene even if the regulation protein is in place.

- Sometimes the transcription of a gene may be controlled by a chain of genes, the first of which is required to activate in order for the protein regulating another gene to be produced (see Figure 1.5).
- The length of non-coding DNA sequences between genes can vary widely, between hundreds of bases and in some cases less than 10. This means that by considering the upstream sequence to be the preceding 200 bases, it is quite possible that we will 'back into' the previous gene and pick up motivic sequences from within that gene.
- This leads to the fourth problem, which is where a number of genes that are very close to one another (sometimes only a few bases apart) are activated by one binding site; when a protein binds to the binding site (known as an operon), all of the genes transcribe simultaneously (see Figure 1.6). The problem created here is that without an in-depth knowledge of the layout of the genome, it is hard to tell if a gene belongs to a group controlled by an operon. The 200bp upstream region for that gene will also tell us nothing about how that gene is regulated.

From all of this it is clear that the regulatory mechanism can be extremely complex.

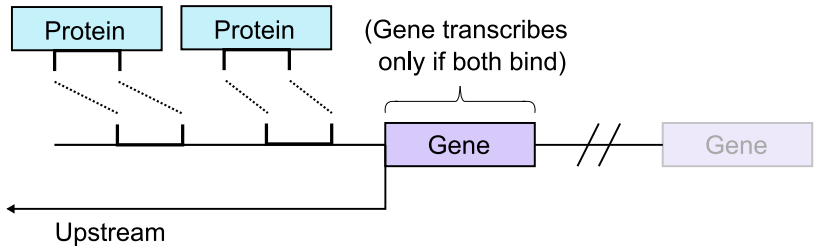

Figure 1.4: Multiple regulation proteins may be require to activate transcription

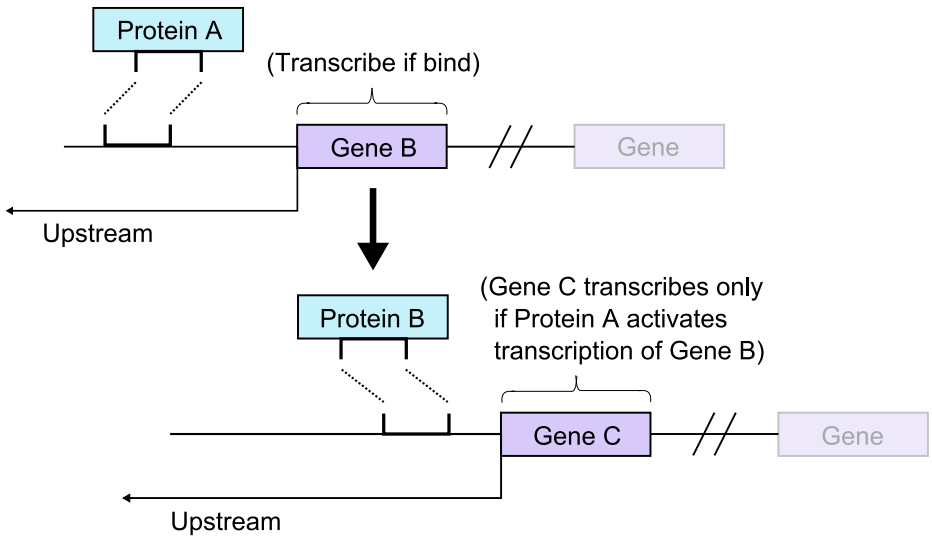

Figure 1.5: Transcription may depend on other genes producing proteins

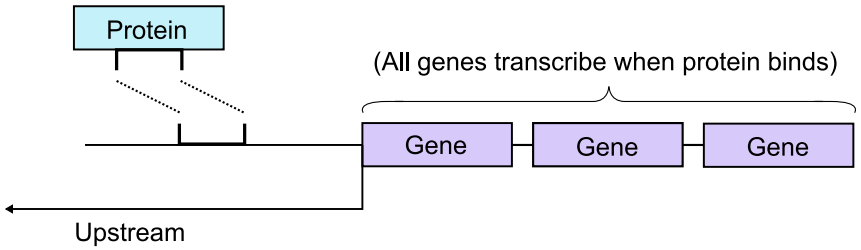

Figure 1.6: A gene controlled by an operon may be activated by a protein binding much further back in the genome

1.4 Sequence Logos

Sequence logos are graphical representations of nucleic acid sequences that will be used to illustrate concepts and results in this project. Each position in the sequence is represented by a stack of symbols (A, C, G or T) where each symbol represents a different nucleic acid (Adenine, Cytosine, Guanine or Thymine). The height of each symbol in a stack represents the probability of that symbol being in that position in the sequence; it follows that each stack sums to 1. Figure 1.7 shows a typical sequence logo as used in this project.

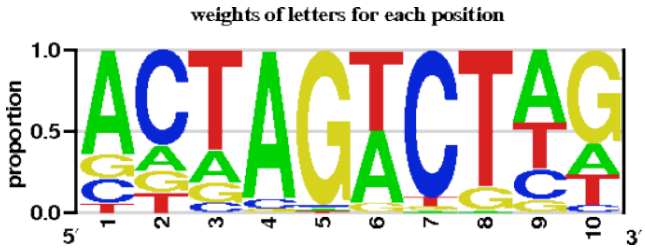

Figure 1.7: An example of a sequence logo

The sequence logos used in this project were created using the online enoLOGOS tool[1] (Workman, *et al.*, 2005).

1.5 Thesis Overview

The rest of this thesis is structured as follows. Chapter 2 presents a review of previous work carried out in this field, including related MSc projects. Chapter 3 presents a description of the theoretical and practical work carried out in this project, including the tests carried out using the implemented application. The results of these tests and a critical analysis of the results are presented in Chapter 4. Chapter 5 presents conclusions and indications of paths for further research.

[1] http://www.benoslab.pitt.edu/cgi-bin/enologos/enologos.cgi

Chapter 2

Literature Review

Since the first motif sequences were found in DNA in the early 1970s there have been many studies in this area, reflecting the amount of interest in finding similar 'motif' subsequences within DNA sequences (Pevzner and Sze, 2000). A number of these studies are reviewed here. Previous work on magnetic bacteria in particular will also be useful references for this study; we are interested in *Magnetospirillum magneticum* sp. strain AMB-1 in this study, however, this work could also be extended to the genomes of other magnetic bacteria.

2.1 Motif Discovery

Das and Dai (2007) survey a number of motif-finding algorithms, comparing approaches and evaluating results. Motif-finding algorithms are split into two distinct types (Das and Dai, 2007): firstly, algorithms which detect possible motifs by searching for statistically overrepresented patterns in a set of genes (i.e. patterns which occur more often than would be expected to occur by chance alone) and secondly, more complex algorithms which take advantage of comparing genomes from different similar species, also known as phylogenetic footprinting or comparative sequence analysis.

2.1.1 Probabilistic Algorithms – EM methods

Das and Dai note that the majority of probabilistic motif discovery algorithms use statistical techniques such as the Expectation Maximization (EM) algorithm and Gibbs

sampling as well as extensions of these techniques. The EM algorithm was first used by Lawrence and Reilly (1990) to search for discovering protein motifs; however, EM can also be applied to motif discovery in DNA sequences. Lawrence and Reilly claim that one advantage of using EM is that "the proposed EM algorithm has the ability to examine motifs [...] which may be nonadjacent in sequence"; that is, the discovered motif need not appear in exactly the same place in each of the sequences in the input dataset. The use of the EM algorithm for motif discovery was extended by Bailey and Elkan (1994a,b) into the MEME algorithm, which incorporates a number of novel features which improve the original.

Bailey and Elkan (1994a) describe a finite mixture model which can be used to represent the input data. The EM algorithm is used to discover the unknown parameters defining the model; given a certain motif width and a number of input sequences, the EM algorithm first estimates (the E step) the start location of the most significant motif within the sequence and then maximises (the M step) the expected likelihood of the data given the current estimates of the parameters. These two steps are repeated until either convergence is reached or a user-defined iteration limit is reached. There are a number of advantages in using this algorithm; one novel feature of the MEME algorithm is its ability to probabilistically 'erase' discovered motifs from the dataset – this allows the most statistically significant motif to be found, then 'erased', then the EM algorithm is run again to find the second most statistically significant motif, until a user-defined number of motifs have been found. If we believe that multiple motifs could be factors in transcription regulation, this is a good reason to choose the MEME algorithm. It is also claimed that the MEME algorithm has the ability to cope with local minima within the search space of alternative models (through multiple runs to find the maximum converged log likelihood), so theoretically we should obtain good results through the use of the MEME algorithm. There is, however, some requirement in that the width of the motif must be supplied by the user; there is at present no way to know the exact length of a motif before running the analysis. One way around this problem would be to set up a script or build a loop into the implemented algorithm, looping through a number of different motif widths. Some additional analysis would then need to be performed to evaluate the most likely motifs, based on some prior expectations regarding the motif widths, for instance, we would not expect the motif to be of width 2, or of width 200. Comparing the log likelihood between two models using different widths would not be a good measure of the 'correctness' of a motif

as the log likelihood of the model is bound to increase as the motif width increases. Another possible disadvantage proposed by Bailey and Elkan is that it is not possible to estimate the number of different motifs in a given dataset.

Bailey and Elkan (1995a,b) have also described several extensions to the MEME algorithm which increase its ability to find motifs in an unsupervised fashion by exploiting prior beliefs or background knowledge regarding the input data. These extensions include detection of palindromic regulatory sequences and a heuristic 'rule of thumb' for automatically determining motif widths. This heuristic allows the MEME algorithm to automatically discover a number of motifs of different widths within a dataset. These extensions are shown to give good results and have been the basis for the MEME algorithm since; this shows the benefit of using prior knowledge when using the MEME algorithm.

2.1.2 Probabilistic Algorithms – Gibbs sampling methods

The class of probabilistic algorithms also includes algorithms such as MotifSampler (Thijs, *et al.*, 2001), and AlignACE (Roth, *et al.*, 1998 and Hughes, *et al.*, 2000), which are based on the Gibbs sampling method; methods such as these have also been used extensively for motif discovery. The most basic Gibbs sampler method as developed by Lawrence *et al.* (1993) takes a Markov Chain Monte Carlo approach and assumes that there is at least one instance of a motif pattern in every input sequence; although Lawrence *et al.* describe a method for motif discovery in protein sequences, the method could be equally applied to DNA sequences. The Gibbs sampling method is built on the same statistical framework as the MEME algorithm; like the MEME algorithm, an iterative sampling method is used, indeed, Lawrence *et al.* note that the Gibbs sampling method can be viewed as a stochastic version of the EM algorithm used in MEME. As in the MEME algorithm, two data structures are maintained and evolved by the steps in the algorithm; these data structures hold probabilistic models of the residue (background) frequencies of the input sequence and the alignment (or motif) sequence, which is the most common pattern within the input sequence. In the 'update' step, one of the input sequences z is chosen at random; an alignment sequence and residue frequencies are then calculated from all the current positions in the remaining input sequences. In the 'sampling' step, every possible motif within z is considered as a possible motif; the probabilities of generating each possible motif within z are then

calculated and a weight assigned to each possible motif. Once each possible motif is weighted, one is chosen at random to be the new position used in the 'update' step. The central idea of the Gibbs sampling method is that the more accurate the alignment sequence calculated in the 'update' step, the more accurate the determination of its location within the input sequences in the 'sampling' step. Given random positions in the 'sampling' step, the alignment sequence will not favour any potential motif in particular; however once some correct position has been chosen by chance, the algorithm tends to favour further correct positions (Lawrence, *et al.*, 1993). The objective is to identify the "best" (most probable) alignment sequence by maximising the ratio of the corresponding alignment sequence probability to background probability (Liu, J.S., *et al.*, 1995; Werhli and Husmeier, 2007). The Gibbs sampling method described by Lawrence, *et al.* (1993) allows multiple motif sequences to be searched for simultaneously rather than sequentially. This has the advantage that information gained about one motif can be used to aid discovery of the others; this differs from the MEME algorithm which effectively 'erases' previously discovered motifs from the dataset and therefore must be sequential. This also has the advantage that any 'incorrect' motifs do not have a negative effect on other motifs, whereas any motif discovered by MEME will be probabilistically erased before continuing to search for other motifs, regardless of whether the discovered motif is 'correct' or not. The disadvantage discussed above of having to repeat the procedure for each possible motif width is found again in this method; Lawrence, *et al.* note that like the MEME algorithm it is not possible to directly compare models over a range of different plausible widths.

The AlignACE algorithm (Hughes, *et al.*, 2000) extends the basic Gibbs sampling method in a number of ways. A maximum *a priori* log likelihood score is used to judge alignments that are sampled during the run of the algorithm; this is used to gain a feeling of how overrepresented the alignment is within the test dataset. The algorithm also provides a measure which takes into account the sequence of the whole genome; obviously any alignments returned should be overrepresented within the test dataset and relatively much less common in the genes which make up the rest of the genome (Tompa, *et al.*, 2005). One interesting extension implemented by Hughes, *et al.* is the consideration of both strands of the input DNA sequence at each sampling step of the algorithm; that is, not only the single DNA strand given as an input to the algorithm, but also the complementary DNA strand.

Although algorithms based on discovering statistically overrepresented motifs such as

MEME and AlignACE are relatively simple, they have been shown to give good results. It is perhaps an indication of the success of the MEME algorithm that work is still being carried out on this algorithm 15 years after it was first proposed; current work is mainly based on the online MEME server, allowing users to upload sequences and have the results emailed to them. Proposed future directions include adding algorithms which remove sequences of low complexity and integrating online MEME with other algorithms to improve results (Bailey, *et al.*, 2006)

2.1.3 Other machine learning approaches

Besides these probabilistic models, there are a number of other machine learning approaches to the motif discovery problem. Genetic algorithms (Liu, F.F.M., *et al.*, 2004) and neural networks (Liu, D., *et al.*, 2006) have both been used for DNA motif discovery with some success. Combinatorial and graphical approaches have been used to try and simplify the problem of mutations between sequences; the iterative WINNOWER and SP-STAR algorithms proposed by Pevzner and Sze (2000) (also Liang, 2003) try to avoid local optima within the search space and converge to a global optimum; although these algorithms require further development they seem to give good results. Approaches based on aligning a number of potentially related sequences have also been used with some success; it is these approaches (usually some form of aligning algorithm such as Smith-Waterman or Needleman-Wunsch) which form the basis of bioinformatics tools such as BLAST. One relatively recent approach which seems to improve the results of motif discovery is the "ensemble approach" (Hu, *et al.*, 2005). This approach combines the predictions made by multiple runs of a number of different motif-discovery algorithms, clustering them and choosing the most likely results. In tests, the ensemble approach was shown to produce results at least as good as the individual component algorithms, even for longer sequences in higher order species. The advantages of an ensemble approach are clear; this approach could take advantage of good predictions given by the component algorithms and by checking them against others, deliver results with a high level of certainty.

2.1.4 Algorithms based on phylogenetic footprinting

There are also motif discovery methods which do not use any form of machine learning. Algorithms such as CONREAL (Berezikov, *et al.*, 2004) and PHYLONET (Wang and Stormo, 2005) take advantage of cross-species genome comparison; generally, potential motif sites which are well conserved over a number of orthologous promoter regions are good candidates for discovering regulatory motifs. Although these algorithms often produce good results, they have the disadvantage that they are generally much more complex in their implementation and usually require expert information regarding the degree of similarity between the species used. For example, if the species are too similar, the motif returned is obvious and uninformative; if the species are too distinct, the comparison fails as it is too difficult to find enough similarities to compare the genes.

2.1.5 Comparison of algorithms

A comparison of the available machine learning methods for motif discovery was carried out (Tompa, *et al.*, 2005) with the two main objectives of providing a benchmark of data sets for assessing different methods and in the process providing some assessment of the relative accuracy of the available methods. Methods that use external data such as phylogenetic footprinting were not included in the assessment; although these methods were considered to be important and effective for motif discovery, it was not deemed fair to include them in the comparison as the additional biological data required for these methods may not be available for the specific organism studied. A number of tools, including MEME, AlignACE and MotifSampler were tested using the TRANSFAC dataset (a commonly used database of known real transcription factors and their binding sites; Wingender, *et al.*, 1996) and the results presented and discussed. The results presented indicate that overall, the absolute correctness of the majority of the algorithms was very low, for example, it is claimed that the average correlation coefficient (a measure of how well the resultant motifs coincide with the known motifs in the dataset; Tompa, *et al.*, 2005) is 0.2, where a coefficient of 1 represents a perfect correlation between the result and the expected result and a coefficient of 0 indicates no correlation. However, there are a great number of reasons why the results appear to be poor, the most important being that the current knowledge of biological regulatory mechanisms is far from complete. This means that there is no

absolute standard against which to measure the accuracy of the methods (a point also noted by Das and Dai, 2007); for this reason, it is perhaps the case that the benchmark data sets used are a poor approximation of true biological data. In addition, many of the transcription binding sites in the TRANSFAC dataset are unusually long, up to 71bp in length; it is possible that the increase in length has a detrimental effect on the results returned and the way they are scored. Tompa *et al.* (2005) note that this fact may indicate a lack of precision in the experimental method used. It is also noted that although the algorithms were applied by experts, the amount of time invested by the experiment participants varied widely and so the results could perhaps be improved given more time to experiment with the data. These reasons and others mean that the relatively poor results should not be interpreted as meaning that the methods themselves are poor. Despite the results, some interesting observations on the results were made. Two versions of the MEME algorithm were run independently and the results were remarkably consistent. Of all the tools tested, the Weeder algorithm (Pavesi, *et al.*, 2004) gave significantly better results than the other algorithms; this is thought to be through the cautious use of the algorithm, only returning the strongest results and therefore not decreasing the accuracy score of the algorithm by returning results which may possibly be wrong. A summary of the discussed algorithms is presented in Table 2.1.

The conclusions made by Das and Dai (2007) in their survey are broadly the same as that by Tompa, *et al.* (2005). Again, it is noted that the incomplete understanding of the underlying biological mechanisms means that it is difficult to provide a good evaluation of various motif discovery methods. The main recommendation is that biologists should use a combination of complementary methods rather than relying on a single one, which should increase the success rate for motif discovery; this ensemble approach should also increase the certainty for the returned motifs. This reflects the success of the ensemble algorithm as discussed above.

Name	Method	Advantages and Disadvantages	References
MEME	Probabilistic - EM algorithm	**Advantages**: relatively simple, well understood, allows discovery of multiple motifs **Disadvantages:** requires user to know motif width, cannot estimate number of possible motifs, sequential motif discovery	Bailey and Elkan (1994a,b; 1995a-c) Bailey, et al. (2006)
AlignACE	Probabilistic - Gibbs sampling	**Advantages**: relatively simple, allows simultaneous discovery of multiple motifs **Disadvantages:** requires user to know motif width	Roth, et al. (1998) Hughes, et al. (2000) Lawrence, et al. (1993)
MotifSampler			Thijs, et al. (2001) Lawrence, et al. (1993)
WINNOWER	Graphical network	**Advantages:** generally converges to a global optimum **Disadvantages:** not as well understood, complex algorithms	Pevzner and Sze (2000)
SP-STAR			
CONREAL	Phylogenetic footprinting	**Advantages:** usually gives good results **Disadvantages:** complex algorithms, usually require expert knowledge	Berezikov, et al. (2004)
PHYLONET			Wang and Stormo (2005)
Weeder	Consensus-based clustering	**Advantages:** seemingly good results (but see main text) **Disadvantages:** complex algorithm	Pavesi, et al. (2004)

Table 2.1: A summary of motif discovery algorithms

2.2 Previous Applications of Motif Discovery Algorithms

Besides the comparison carried out by Tompa, *et al.* (2005), the motif discovery algorithms discussed above have also been used in a large number of real-world studies. Baker, *et al.* (1999) used MEME to discover motifs from an input dataset of around 200 divergent hydrogenases; the matrix representations of the discovered motifs used were subsequently used as an input to the MAST and Meta-MEME tools, which were used to score each motif and create a hidden Markov model of all the motifs respectively. In this study, MEME identified six motifs that were regarded as being functionally important to the studied proteins. Heikkinen, *et al.* (2008) also used MEME alongside Weeder and MotifSampler to identify transcriptional regulatory sites in the upstream regions of MicroRNA sequences extracted from the genomes of *Caenorhabditis elegans* and *Caenorhabditis briggsae*. A previously unknown motif was discovered and evaluation suggested that it may be involved in miRNA sequence regulation.

As noted earlier, the AlignACE algorithm was used by Hughes, *et al.* (2000) to study

groups of genes in the *Saccharomyces cerevisiae* genome. AlignACE returned a very large number of potential motifs. Further comparison and clustering of the potential motifs narrowed the results and many previously identified *cis*-regulatory elements were found. In addition, motifs which were previously unidentified were also described, one of which was verified by laboratory experiments. Grainger, *et al*. (2007) used AlignACE to search for sequence motifs present in a number of *Escherichia coli* DNA sequences. Previous analysis had identified 43 potential FNR (a global transcription regulator protein) binding site targets, 33 of which were identified and confirmed using AlignACE.

Although, as previously noted, phylogenetic footprinting methods require expert knowledge regarding the similarity between compared species, a number of studies have used phylogenetic footprinting algorithms such as CONREAL and PHYLONET. It is evident that frequently the algorithm is run by groups already possessing a large amount of knowledge about the studied organism. Liu, J., *et al*. (2008), having previously selected a number of similar *Shewanella* genomes for comparative analysis and determined a number of orthologous genes, used phylogenetic footprinting to identify all possible conserved palindromic motifs in each set of orthologous promoter regions. PHYLONET was then used to identify motifs that were common to multiple sets of orthologous promoter regions. In total, 209 unique DNA motifs were obtained, just over a third of which had some additional supporting evidence besides conservation in other genomes (e.g. matching to known transcription factor binding motifs).

2.3 Previous Work on *M. magneticum* sp. strain AMB-1

2.3.1 Genetic Analysis of AMB-1

The first comprehensive genetic analysis of AMB-1 was completed in 2005. Matsunaga *et al.* (2005) describe the genome of AMB-1, attempt to identify the specific genes required for magnetosome production and describe some of the potential applications of magnetosomes. Four major stages are hypothesized for the process of magnetosome production. It appears that the most important is a strictly controlled iron oxidation-reduction stage (see also Matsunaga and Okamura, 2003); the amb3335 gene (BAE52139.1: predicted ferric reductase) is singled out as being important in this process. In addition, the number of proteins expressed in the lipid membrane of

the magnetosome particles prompted an analysis of the genes encoding ferredoxin and cytochromes; these genes are thought to be involved in iron reduction.

2.3.2 Investigating Regulatory Networks in Alphaproteobacteria

An investigation into the regulation networks of a number of different alphaproteobacteria was carried out by Rodionov, *et al.* (2006). A genome comparison was performed and used to investigate the distribution of DNA motifs in the upstream regions of genes involved in iron and manganese homeostasis; the results of this were combined with a number of computational approaches in order to reconstruct metal regulatory networks in a large number of alphaproteobacteria with fully sequenced genomes. In the process, a list of 17 genes thought to be important in iron co-regulation in the AMB-1 genome are listed (Table S6[1]); these are used as an input to the motif discovery program SignalX (part of GenomeExplorer), which uses the Smith-Waterman algorithm to perform nucleotide sequence alignment[2] in a similar way to BLAST. A possible motif is presented using the results of SignalX. It is noted that despite the key role of iron processes in alphaproteobacteria (in general, not just in the *Magnetospirillum* genus), very few studies have been concerned with iron regulation in these bacteria. This is clearly one area which requires further study.

2.3.3 Identifying Regulatory Sequences in Magnetic Bacteria

Similar to the current project, the study by Zhou (2008) aimed to evaluate the current methods for discovery of regulatory motifs, develop an implementation of the most promising and then use this implementation to explore potential DNA motifs used in regulation of magnetosome production. Three methods of motif discovery were identified; firstly, a naïve method such as that suggested by Gelfand, *et al.* (1999), which attempts a statistically global alignment of a number of related genes and searches for an *l*-mer with the largest accumulative weight value. This is similar to the alignment algorithms discussed above; the naïve method suggested is not always applicable, however, since the location of the motif need not be the same for each gene (as

[1] http://www.ploscompbiol.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pcbi.0020163.st006 (Accessed 17th July 2009)

[2] http://bioinform.genetika.ru/projects/reconstruction/Similarity%20Search.htm (Accessed 24th July 2009)

discussed above). A second method, known as "palindrome search" is also identified; this method is based on the idea that most prokaryotes have biologically palindromic or quasi-palindromic regulatory sequences (i.e. the reverse complement of the sequences is the same as the original sequence). The final method is the MEME algorithm as described above.

The naïve method described by Gelfand *et al.* (1999) was chosen for implementation, although there does not seem to be a clear reason why this method was chosen over the others. The implementation was then modified to overcome the problem of motif location as described above; a palindrome heuristic was also incorporated into the algorithm. The implementation was then tested and evaluated; it was discovered that although there are no bugs in the coded implementation, the system has a number of flaws when searching for candidate regulatory motifs. Perhaps most important amongst these flaws is the fact that it is not guaranteed that all motifs are palindromic or even quasi-palindromic; it appears this heuristic has been deduced from a number of experimentally confirmed motifs and is certainly not a rule to which all regulatory motifs must adhere. This means that the implemented algorithm may not find all potential motifs in an input sequence. In addition, the user has a large number of parameters to input for while searching, most notably the l-mer parameter which determines the length of the motif. There is at present no way to know the length of a motif in advance, so the user must repeat the search for each different motif length.

Zhou's implementation and preprocessed data are no longer available, meaning that it is not possible to carry out a comparison between the implementation created in the current project and the implementation created by Zhou. This also means that the data preprocessing will have to be carried out anew; however, a detailed description of the required steps is included in Zhou's report.

While the study by Lan (2008) had roughly the same aims as that by Zhou (2008), a more bioinformatical approach was taken, using comparative genomics to investigate and predict regulatory motifs rather than a probabilistic model. An orthology analysis was used to identify potential regulatory sequences in AMB-1 by examining other well-understood alpha-proteobacteria, concentrating on the CtrA (cell transcriptional regulator) and Fur (ferric uptake regulator) metabolisms. Orthologs, or orthologous genes, are genes in different species that are similar to one another because both species evolved from a single common ancestor; these genes generally have a similar function and structure. As a result of these properties, orthology analysis can be used to discover

the function of unknown genes in one species from well-understood genes in another related species. Once the potential regulatory sequences were discovered, the possible motifs were predicted using motif discovery programs (mainly GenomeExplorer). Using these methods, two possible CtrA motifs were found which were similar to the CtrA motifs in *Caulobacter crescentus* (the 'related' species used). The study of the Fur-regulated metabolism was mainly to confirm the work carried out by Rodionov, *et al.* (2006) and the motif discovered was very similar to the motif published in that study.

2.4 Discussion

One conclusion that can be drawn from the algorithm surveys of Tompa *et al.* (2005) is that all the tested machine learning algorithms (with the exception of Weeder, see above) give similar performance levels overall, while on specific datasets, some algorithms may give better results than others. However, without testing each algorithm separately and some knowledge about the motifs to be discovered in that particular dataset, it is hard to predict which algorithm which will give the best performance; additionally, we would not normally be carrying out motif discovery if we knew the motifs in the dataset!

There are advantages and drawbacks for almost all of the algorithms discussed above, however, we can use some criteria to narrow down our choice. Perhaps most important in this project is the time constraint, it therefore makes sense to implement one of the probabilistic algorithms rather than one of the more complex phylogenetic footprinting algorithms. Probabilistic algorithms have the advantage that they are relatively simple and well understood; this would perhaps allow for more time experimenting with algorithm extensions following the implementation of the chosen basic algorithm. The time constraint also rules out the ensemble technique as developed by Hu, *et al.* (2005) as this would involve the implementation of more than one good algorithm, which is unlikely to be feasible given the allotted time to complete this project.

The Gibbs sampling method used in algorithms such as AlignACE has the advantage that it deals easily with multiple motifs in a single run. This not only allows the algorithm to take advantage of the alignment of one motif and apply that information to gain knowledge about other motifs, but it is also claimed that this increases the speed of

the algorithm. One large problem faced by any algorithm in solving this kind of problem is the large dimensionality of the search space and the subsequent tendency for the algorithm to converge to a local rather than a global optimum; clearly we would like the algorithm to converge to the global optimum value. In the Gibbs sampling method, each dimension of the search space is explored one dimension at a time (Lawrence, *et al.*, 1993); the addition of stochastic sampling means the Gibbs sampling method is reasonably robust in dealing with multiple local optima in a single run. In contrast, the MEME algorithm must be run a number of times with different starting parameters in order to choose the best result, which may still not be guaranteed to be a global optimum solution. There are, however, a number of heuristic rules which can be used to ensure as much as possible that the solution returned is the optimum one.

Bailey and Elkan (1994b) claim that MEME has a number of advantages compared to the Gibbs sampling method. Firstly, MEME does not require input sequences to be classified in advance by a biologist as definitely containing a motif; the capability of MEME to deal with noise is quite robust. Secondly and perhaps more importantly, MEME searches the space of possible starting points for gradient descent optimization systematically, meaning that the algorithm always converges in a predictable way, usually requiring only a relatively small number of iterations. In contrast, algorithms employing the Gibbs sampling method intersperse the gradient search steps with stochastic jumps within the search space; this means that it is possible that these algorithms can spend an unpredictable number of iterations jumping around the search space, effectively leaving the algorithm on a plateau before converging, increasing the running time of the algorithm dramatically. This can be seen by considering the mechanism of the Gibbs sampling algorithm described above. The algorithm will only start to converge after some correct position has been chosen; however, the positions are chosen randomly and so it may take relatively few iterations to chance upon some correct motif, conversely, it may take a huge number of iterations.

One limitation of both MEME and Gibbs sampling algorithms is the constraint that all motifs searched for must have the same width; it is likely that this limitation is not consistent with the real world data on which any algorithm will be used. In addition, it is very difficult to compare results from searches with two different widths as the likelihood values returned by the algorithm are not comparable for runs with different widths. Another limitation affecting both types of algorithm is the phenomenon of

'shifting'[3], where the algorithm iterates to a solution which is a shifted version of the optimal pattern (this is illustrated in Figure 2.1). Lawrence, *et al.* (1993) suggest that this could be improved by inserting a comparison step after every n iterations, comparing the likelihood of the current pattern with shifted versions of that pattern and switching to one of the shifted versions if it has a higher likelihood.

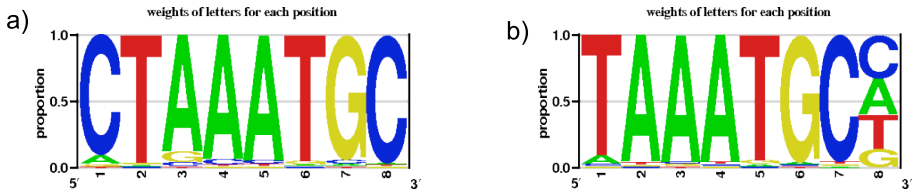

Figure 2.1: The effects of shifting. The motif to be discovered is CTAAATGC, as shown in a), however, the motif returned by the algorithm is TAAATGCx as in b), perhaps due to peculiarities of the background data. It appears as if the actual motif has been shifted one place to the left. It is expected that by shifting the motif discovered in b) one place to the right, we can increase the log likelihood of the result; solving this problem is likely to improve the success of an algorithm

Considering all the algorithms surveyed, it makes sense to implement and extend the MEME algorithm because of the advantages as described above. While there are some limitations it is has been proved that MEME produces good results and there is a relatively large amount of literature concerning the basic MEME algorithm and its extensions. A 'second choice' of algorithm would perhaps be the AlignACE algorithm due to its relative simplicity and similarity in structure to the MEME algorithm. There is the possibility of adding some extensions to the MEME algorithm implemented in this project; these could include an extension allowing for gaps in the motif (it appears that (quasi)-palindromic motifs especially may have small gaps in the motif), for example. The question of why a new implementation of MEME is required when the online MEME server is available could be asked, however, the source code for the current MEME algorithm is written in C and includes extensions added piece by piece over the past 15 years. By creating a new Java based implementation of MEME, it is hoped to test the intended hypothesis as easily as possible; it is also hoped that the resultant code will be extensible, allowing for additions as required.

[3]Lawrence, et al. (1993) use the term 'phase shifting'; following Bailey and Elkan (1994b), the term 'shifting', which does not carry any notion of periodicity, is used here.

The hypothesis that this project intends to test is that extensions to the implementation of the MEME algorithm will improve its performance for discovering regulatory sequence motifs in bacteria with respect to the original MEME algorithm. It is assumed that the basic MEME algorithm is capable of discovering DNA sequence motifs in general; this is believed to be the case because the MEME algorithm has been proved to work in the past and has achieved good results on bacteria of other species which has been confirmed experimentally.

Chapter 3

Regulatory Motif Discovery

This chapter presents a description of the theoretical and practical work carried out in this project. An explanation of the data preprocessing procedure is provided, followed by a theoretical explanation of the MEME algorithm and how it is implemented in this project. The implemented application, known as JMeme, is then tested in order to identify the characteristics in dealing with data, before being applied to new unknown data. Two extensions to JMeme are also described and implemented as JMemePlus.

3.1 Data Preprocessing

3.1.1 Introduction

Before any motif discovery can be performed, some preprocessing must be applied to the available raw data in order to extract the relevant data in a usable format. The following sections outline the genome data used, the tasks which must be carried out in order to create a dataset and how these tasks were completed.

Background to the Genome Sequence

As noted in Section 1.2, the first comprehensive analysis of the AMB-1 genome was completed by Matsunaga, *et al.* in 2005, at Tokyo University of Agriculture and Technology, Japan. The results of this study, along with other efforts, have been combined by the National Center for Biotechnology Information (NCBI) into an annotated file

containing the full nucleotide sequence and details about the structure, function and location of each gene within the genome. The full nucleotide sequence for the AMB-1 genome consists of 4,967,148 base pairs. Of this, 88% is coding DNA, containing instructions for producing proteins. The AMB-1 genome contains 4,611 genes, of which 4,559 are protein coding genes; it is these genes in which we are interested in this study. The genome can be freely downloaded from the NCBI website[1]; the downloaded sequence data file will be further discussed in Section 3.1.2.

Computing Tasks

Three main computing tasks have been identified as being necessary in order to create a usable dataset. Firstly, details of all the genes must be extracted from the downloaded data file. A subset of the data must then be created based on some user query. The data matching the user query must then be returned, preferably in the FASTA format as this is a widely-used text-based format for the representation of nucleotide sequences[2]. The computing tasks and their implementation will be further discussed in Section 3.1.3.

3.1.2 Data

The complete genome data for AMB-1 is contained within a GenBank database file (known as a .gbk file). GenBank is an annotated collection of all publicly available nucleotide sequences and their protein translations. Although it was not strictly necessary, the AMB-1 .gbk file was downloaded from the NCBI database; the large size of the file (around 10MB) meant that performing operations on the remote file would increase the program running time. By having the .gbk file available locally, the preprocessing step can be carried out more quickly. All .gbk files have the same defined syntax, which lends them to being read automatically by computer program; the file provides a number of useful tags, which can be used to extract the relevant information. The AMB-1 .gbk file contains the following information (see also Figure 3.1):

- The ‘gene’ tag indicates the start of information about a new gene. This is followed by two numbers, the first of which represents the start point of the gene

[1] http://www.ncbi.nlm.nih.gov/sites/entrez?Db=genome&Cmd=ShowDetailView&TermToSearch=19021 (Accessed 17th July 2009)

[2] http://www.ncbi.nlm.nih.gov/blast/fasta.shtml (Accessed 17th July 2009)

within the genome and the second represents the end point of the gene. In Figure 3.1, the first gene starts at base 239 and ends at base 1336; the last gene in the genome is shown to start at base 4,966,807 and end at base 4,967,148. The 'gene' tag may also indicate that a gene is said to be complementary using the 'complementary' tag (see Figure 3.2); the meaning of this will be discussed shortly.

- The '/locus_tag' tag indicates the number of the gene within the genome. In the AMB-1 genome, genes run from amb0001 to amb4559.
- The '/product' tag indicates the product, or predicted product, (usually as a protein name) of the gene. In the case of amb0001, the product is 'Predicted GTPase'. Similarly for amb4559, the product is 'Ferredoxin'.
- The '/protein_id' tag indicates the ID of the protein produced by the gene.
- After all the genes have been listed, the full genome sequence is provided, indicated by an 'ORIGIN' tag.

As noted above, some of the coding genes are said to be complementary. The .gbk file contains only one strand of the DNA double helix structure; when a gene is said to be complementary (as in Figure 3.2), this means that the coding gene is not in the strand given in the .gbk file (5'-3'), but the other (complementary) strand. However, the structure of DNA means that given one strand, we can easily find the other; this is a consequence of the way nucleotide bases link together (e.g. adenine always links to thymine and cytosine always links to guanine). If a gene in the .gbk file is complementary, this means that rather than coding from the 5' end to the 3' end of the genome as given in the .gbk file, the gene codes from the 3' end to the 5' end. It may help to consider a gene as reading from left to right (5'-3') within the genome (Figure 3.3). We would consider the 'upstream' sequence for this gene to be the 200bp before the start codon of the gene. For complementary genes, the gene codes in the opposite direction (3'-5'), therefore the 'upstream' sequence is the 200bp after the end point given in the .gbk file. This sequence therefore has to be complemented (as it is in the complementary strand) and reversed, to be consistent with the first case above.

```
LOCUS AP007255 4967148 bp DNA circular BCT 26-JUN-2008
DEFINITION Magnetospirillum magneticum AMB-1 DNA, complete genome.

[information omitted]

gene 239..1336
     /locus_tag="amb0001"
CDS  239..1336
     /locus_tag="amb0001"
     /codon_start=1
     /transl_table=11
     /product="Predicted GTPase"
     /protein_id="BAE48805.1"
     /db_xref="GI:82943941"
     /translation="MAELHLHGGRAVAAALTARLGELGLRPAEPGEFSRRAFLNGKLD
     LTRAEAIADLVDAETAAQRRQALRQLDGGLAGLVEGWRSALVRAMAHLEAVIDFADED
     IPDTLLEQSVGEVRSLRREMEVHLDERRNGERLRDGIHITILGAPNAGKSSLLNRLAG
     REAAIVSAQAGTTRDVIEVHLDLGGWPVIVADTAGLRDSACEIESEGVRRAADRAAKA
     DLRLCVFDGTLYPNLDAATLEMIDDATLVVLNKRDLMTGETPASINGRPVLTLSAKAG
     EGVDDLVAELARVVESRFAMGSAPVLTRERHRVAVAEAVAALSRFDPGLGIEMAAEDL
     RLAARSLGRITGRVDVEEILDVIFHEFCIGK"

[information omitted]

gene 4966807..4967148
     /locus_tag="amb4559"
CDS  4966807..4967148
     /locus_tag="amb4559"
     /codon_start=1
     /transl_table=11
     /product="Ferredoxin"
     /protein_id="BAE53363.1"
     /db_xref="GI:82948499"
     /translation="MSRLPGDPPDYFRVHVFICTNRRPDDNKRGSCAGRGSEALREHM
     KDAQKKLGLKDVRINSAGCLDRCGKGPVMVIYPEGIWYSFNSVADLDEILETHIVGGG
     RVERLMLAPNS"

ORIGIN
      1 tcttcctccc atgtccgaga ccatctatgc ccttgccagt gccgccggaa gggccggaat
     61 cgccgtctgg cggctgtcgg gcgagggcag tgggacagcg ctgtccgccc tgaccggcaa
    121 gcccctgccc gagccgcgcc gggcccggcg ggtccgtttg cgcgacgggg cgggggaggt
    181 gctggacgac gggctggtcc tgtggttccc cgcaccccat tcctttaccg gcgaggatgt

[information omitted]

4966921 gcgctgcgcg aacacatgaa ggacgcccag aagaagctcg gcctcaagga tgtacggatc
4966981 aactccgccg gctgcctcga ccgctgcggc aaggggccgg tgatggtgat ctatcccgag
4967041 ggcatctggt acagcttcaa tagcgtggcc gacctggatg aaattctcga gacgcatatc
4967101 gtcgggggcg gccgggtcga gcgtctgatg ctcgcgccca attcctga
//
```

Figure 3.1: Some of the information contained in a .gbk file

```
gene  complement(9900..11123)
      /locus_tag="amb0012"
CDS   complement(9900..11123)
      /locus_tag="amb0012"
      /note="related FAD-dependent oxidoreductase"
      /codon_start=1
      /transl_table=11
      /product="2-polyprenyl-6-methoxyphenol hydroxylase"
      /protein_id="BAE48816.1"
      /db_xref="GI:82943952"
      /translation="MSTSSLLRVDVLINGGGPVGALLAAVLGRAGIRVAVVEAAPPEI
      LGRPGSDARAIAIAYTARKVIAASGAWEGMKDEAGEILEIRVTDGGSPLFLHYDHQDI
      GDDPLGWIVPNPAIRRELLAALTRSPNVSLLAPARLGRLERTPHRVEAELEDGRVIHA
      ALAVAADGRGSALRQQAGIKVTRRDYHESGIVCIMAHEKPHHGIAHERFLPAGPFAIL
      PLAGNRSGIVWTESNGVAAAICAQDDEAFRAELAAKVGGFLGEIQVEGARFHHPLTLQ
      FAEAMIDHRLALIGDAAHGMHPIAGQGMNMGIRDVAALAEVIADALRLGLDPGGPDVL
      ERYQRWRRFDTMLMLGLTDGLDRLFSNDVELLKHVRRLGLAGVHQLGHTKRFFMRHAM
      GLVGDLPRLMQGKTL"
```

Figure 3.2: A complementary gene in the .gbk file

Non-complementary case: The blue sequence is the upstream sequence we are interested in (i.e. the 200bp before the start codon (green) of the gene).

Complementary case: The blue sequence is the upstream sequence we are interested in (i.e. the 200bp before the start codon (green) of the gene). However, the gene is in the complementary strand and reversed in comparison to the non complementary gene in the .gbk strand - we find the upstream sequence, then take the reverse and complement to make it consistent with the non-complementary cases.

Figure 3.3: Non-complementary and complementary genes

3.1.3 Processing

In order to create an input dataset comprised of a number of candidate upstream sequences, the required upstream sequences must be extracted from the genome. As the genome is made up of almost 5 million base pairs, it would be an incredibly time-consuming process to read through the genome and extract the required sequences manually. In addition to this, manually reading through the genome introduces the high possibility of human error. Clearly it is many times quicker and more accurate for the genome to be read by a computer program, which can automatically extract the required data.

The GBKReader class contains a number of methods that are used to automatically read the .gbk file and extract the data relevant to a query given by the user. There are three main steps in the process; firstly, the details for all genes are extracted from the .gbk file and stored in memory. After this has been completed, the details are searched according to the given query and a set of genes constructed. The final step in the process is to read the .gbk file once more, extract the required nucleotide sequences and write them in the FASTA format to a specified output file (Figure 3.4).

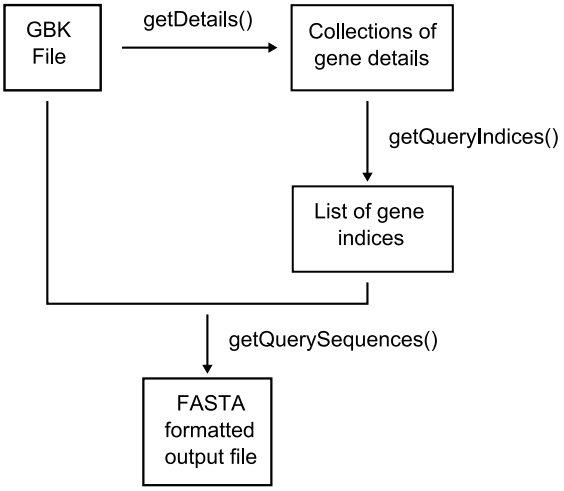

Figure 3.4: Process diagram of the GBKReader class

Data Extraction

The getDetails() method in the GBKReader class is used to extract the basic gene information from the .gbk file; this method uses the Scanner class, provided in the

java.util package (Barnes & Kolling, 2006: 465), to scan through the file. The Scanner object can read input files either word by word or line by line; it is the former mode which is used in the getDetails() method. To extract the gene details, we read through the .gbk file until we reach a 'CDS' token – this marks the start of a protein coding gene. We know that the next "word" represents the start and end points of the gene; we can make this assumption because .gbk files follow a standard format. The word representing the start and end points is split into two integers. Similarly, we read on for the locus, product, protein ID and complementary flag (a Boolean value denoting whether a gene is complementary or not). Whether a gene is complementary or not is stored in another ArrayList, named isComp. isComp takes on a Boolean value for each gene, 'true' if a gene is complementary and 'false' otherwise. Once we have all the details for the gene, a number of ArrayList objects are updated with the details and we move on to the next gene in the .gbk file (again, the ArrayList class is provided in the java.util package).

There are a number of reasons why it makes sense to store the details in ArrayList objects. By storing all the gene details in ArrayList objects in an ordered way, it is simple to return the information for a given gene *i* by calling the get() method on the ArrayList object. For example, the start point of a gene *i* can be obtained using the following code:

```
int startpoint = start.get(i);
```

In addition, the ArrayList class stores objects in the order in which they were added by definition (ibid.: 465), so assuming the data is added to the ArrayList objects in an ordered fashion, we do not need to worry about data for different genes being mixed up - calling get(*j*) on each ArrayList object will always return the details for gene *j*. We continue extracting information until we reach the 'ORIGIN' tag; this tag indicates the start of the nucleotide sequence so we have extracted all the gene information required and therefore do not need to read any further at this stage. The number of genes extracted from the .gbk file can be accessed using the size() method in a similar way; the size of each ArrayList should be the same.

Querying

Now that the basic information for each gene has been stored, we can search over this information to find the genes we are interested in. The getQueryIndices() method in

the GBKReader class builds a set of indices containing the index for each gene that matches the given query. If a gene matches the query, the index for that gene is stored in a TreeSet object. After we have searched through all the genes, we have a complete set of gene indices which allows us to retrieve the nucleotide sequences for the genes we are interested in.

The GBKReader class allows the data to be searched in two different ways. If we know the genes we are interested in and have the identifiers (the locus tags) for these genes, we may use these identifiers as input for the search. For example, if we interested in genes amb3037 and amb4088, we can perform a search with query string "amb3037 amb4088"; this will create a set containing the ArrayList indices for the amb3037 and amb4088 genes. If we do not know the genes we are interested in, GBKReader allows a keyword search to be performed, checking query keywords against the product value for each gene. For example, a search with query "iron" will create a set containing the ArrayList indices for all genes that contain the word "iron" in their product description. Multiple keywords can be searched for in a similar fashion; a search for "iron ferrous" creates a set of indices for all genes that contain either "iron" or "ferrous" in their product description.

While it is possible to use another ArrayList object to store the indices, the properties of the TreeSet object (provided in the java.util package) allow more complex searches to be carried out. As noted above, ArrayList stores objects in the order in which they were added. This behaviour is fine for single query searches but for more complex searches, where we run through the stored details a number of times, ArrayList does not allow for ordering the genes in ascending order, leading to messy search results. In contrast, TreeSet stores objects in a specific order, in this case in ascending locus order. Another disadvantage ArrayList has is its allowance of duplicate values; again, for one search this may not be a problem but for more complex searches, such as the "iron ferrous" search above, it is possible that different searches return the same gene (i.e. one or more genes contain both keywords in their product description), leading to duplication in the search results and inevitably to confusion. TreeSet, based on the mathematical concept of a set, does not allow for duplicates and is therefore more suited to our needs.

Sequence Extraction

All that remains to be done is to extract the nucleotide sequences for the genes that we are interested in from the full genome sequence and output these sequences to a file. This is done by reading through the set of genes constructed in step 2; for each gene in the set, the start and end points are found and the relevant sequence (either the coding sequence or the 200bp upstream sequence) is output to a file.

The main task in this step is extracting the correct sequence from the full genome given in the .gbk file. As shown above, the .gbk file gives the full genome sequences as rows of 60bp. In theory, there should be no limit to the size of a String object in Java so it may be possible to read the entire nucleotide sequence into memory and easily extract the sequence required; in practice, however, this is not feasible due to memory considerations and the length of time it would take to construct such a string. Therefore, we must read through the nucleotide sequence (using the Scanner object in 'line by line' mode) so we can extract from the correct place, reading the correct number of rows.

As shown above, the rows are numbered according to the first base in each row (i.e. 1, 61, 121, 181 etc.). To find the line we should start from, we take the start point of the required sequence and divide by 60, rounding down to the nearest integer (floor function). We then multiply by 60 and add 1 to find the line at which we should begin extraction (Example 3.1).

Example 3.1

We want to extract the sequence starting at base 170.

$$\text{floor}(\tfrac{170}{60}) = 2$$

$$2 \times 60 = 120$$

$$120 + 1 = 121$$

we therefore begin our extraction at the line beginning with base 121.

One exception to this is when the start point for the sequence begins at a number which is exactly divisible by 60. In this case, after rounding down, we should subtract a further 1 from the number before multiplying by 60 and adding 1 (Example 3.2). It appears that the sequence extraction process used by Zhou (2008) did not account for this exception.

Example 3.2

We want to extract the sequence starting at base 3120.

$$\text{floor}(\tfrac{3120}{60}) = 52$$

$$52 \times 60 = 3120$$

$$3120 + 1 = 3121$$

We see that following the same operations as last time will not extract the first base in the sequence that we want. By subtracting an additional 1 after we have divided by 60 and rounded down, we will start extracting with the line beginning with base 3061, correctly including the first base:

$$\text{floor}(\tfrac{3120}{60}) = 52$$

$$(52 - 1) \times 60 = 3060$$

$$3060 + 1 = 3061$$

To find the number of lines to extract, we take the length of the required sequence, divide by 60, again round down to the nearest integer and add 2, to make sure that any overflow is covered (Example 3.3).

Example 3.3

We want to extract a 200bp sequence.

$$\text{floor}(\tfrac{200}{60}) = 3$$

$$3 + 2 = 5$$

we therefore need to extract 5 lines.

Now that we have a sequence of extracted lines, we just have to find out where in this sequence our required sequence starts. This can be found by subtracting the line number of the first extracted line from the start point of our required sequence and extracting the (end – start) bases from that point. We can now combine and apply all the above rules; Example 3.4 shows how we can apply these rules to extract an upstream sequence from the full genome contained in the .gbk file.

Example 3.4

We want to extract the sequence from base 1152 to base 1351 (i.e. the 200bp upstream sequence for the amb0002 gene).

Find the line (L) to start from:

$$\text{floor}(\tfrac{1152}{60}) = 19$$

$$(19 \times 60) + 1 = 1141$$

therefore L is the line beginning with base 1141.

Find the number of lines to extract:

$$\text{floor}(\tfrac{1152-1141}{60}) + 2 = 5$$

therefore we need to extract 5 lines.

Find the base in L at which our required sequence starts:

$$1152 - 1141 = 11$$

therefore our sequence starts at base 11 in line L

The getQuerySequences() method in the GBKReader class implements the above rules and searches through the full genome sequence and returns either the upstream or coding sequence for each gene in the query set. This is then output to a user-specified text file in the FASTA format (as detailed above). If the coding sequence is to be returned, each returned sequence is checked to make sure that the codon for that gene begins with 'ATG', 'GTG' or 'TTG'; if the coding sequence returned does not begin with one of these common codon sequences, the user is alerted so the sequences may be checked manually.

3.1.4 Using the GBKReader application

The GBKReader application may be run from the command line, using text files as both input and output. The application takes five arguments:

- input - a .gbk file containing the genome you wish to extract genes from.
- sequence flag - denotes whether the coding [-c] or 200bp upstream [-u] sequence for each gene should be returned.
- search flag - whether the query string is for a keyword search [-k] or locus search [-l].
- query string - a string containing the term to be searched for.

- output - a file to write the output to (typically a .txt file).

The following examples illustrate how the GBKReader application can be used.

Example 3.5
To extract the coding sequence for gene amb0001 from file AP007255.gbk and write it to out.txt:

```
java GBKReader AP007255.gbk -c -l "amb0001" out.txt
```

Example 3.6
To extract the 200bp upstream sequences for genes amb0001 and amb4559 from file AP007255.gbk and write them to out.txt:

```
java GBKReader AP007255.gbk -u -l "amb0001 amb4559" out.txt
```

Example 3.7
To extract the coding sequence for every gene containing the word "iron" in its 'product' tag, from file AP007255.gbk and write it to out.txt:

```
java GBKReader AP007255.gbk -c -k "iron" out.txt
```

Example 3.8
To extract the 200bp upstream sequence for every gene containing the words "iron" or "ferredoxin" in its 'product' tag, from file AP007255.gbk and write it to out.txt

```
java GBKReader AP007255.gbk -u -k "iron ferredoxin" out.txt
```

3.2 The Finite Mixture Model

Having extracted from the full genome a number of upstream sequences which we believe to contain at least one regulatory motif, we now face the problem of applying a method which will seek out any motifs present in the dataset. As noted in Section 2.1.1, Bailey and Elkan (1994a,b) propose the use of a mixture model to probabilistically model the dataset. The proposed model consists of two generative components; one component (the "motif" model) describes a set of similar subsequences of fixed width, while the other (the "background" model) describes every other position in the

sequences. It is assumed that each position in the input dataset has been sampled from either the motif model or the background model. The parameters of each model are initially unknown: to fit the model to the data, the Expectation Maximization (EM) technique is used to find the maximum likelihood estimates for the unknown parameters from some initial estimates. After the EM technique has been applied to the mixture model, we have maximum likelihood estimates for the parameters of both models; that for the motif model is the most statistically significant motif in the dataset. The finite mixture model we will use to represent the dataset requires three inputs. Firstly, a dataset (referred to as Y) of a number (N) of DNA sequences, through which we will search for possible motifs; secondly, a set motif width (referred to as W) to search with and finally, a fixed 'alphabet' $A = \{a_1,a_2,\ldots,a_L\}$ (in our case $A = \{\text{A, C, G, T}\}$) from which each character in the dataset is drawn. The mixture model does not model each sequence in Y directly; instead, the dataset is conceptually split into all possible (n) overlapping subsequences of width W – we will use this dataset (referred to as X) as input to the model (Example 3.9 shows how an input dataset may be split into n overlapping W-width subsequences). Although this means that the model does not strictly model the original dataset, it greatly simplifies much of the later calculation and is a reasonable approximation in practice.

Example 3.9:
We have a dataset Y of $N = 4$ input sequences:

Y_1 = {C, G, T, T, G, C, G, G, C, G, T, G}
Y_2 = {C, T, G, G, G, C, C, G, C, C, T, G}
Y_3 = {C, C, C, C, G, G, C, A, G, C, G, C}
Y_4 = {C, G, T, C, A, A, C, C, C, C, A, T}

Taking our example motif width $W = 5$, we can divide Y into a list (X) of every possible overlapping width-5 sequence:

X_1 = {C, G, T, T, G}
X_2 = {G, T, T, G, C}
X_3 = {T, T, G, C, G}
...
X_{32} = {C, C, C, A, T}

As previously explained, the mixture model consists of a motif and a background component. The motif component specifies that each position in a subsequence which is part of a motif is generated by an independent random variable describing a multinomial trial with parameter $f_i = (f_{i1},\ldots,f_{iL})$. That is, the probability of a certain letter a_j appearing in position i in the motif is f_{ij} (Bailey and Elkan, 1994b). The background component specifies that each position in a subsequence which is not part of a motif is generated independently, again by a multinomial random variable with parameter $f_0 = (f_{01},\ldots,f_{0L})$. This means that a chosen X sequence which is not a motif is a sequence of W independently generated samples from the single multinomial distribution f_0 . The parameters f_{ij} for $i = 0,\ldots,W$ and $j = 1,\ldots,L$ must be estimated from the data. For reasons which will become clear shortly, we shall refer to the motif model as $\theta_1 = (f_1,f_2,\ldots,f_W)$ and the background model as $\theta_2 = f_0$. To model the input data, we assume that nature chooses either the motif model θ_1 (with probability λ_1), or the background model θ_2 (with probability $\lambda_2 = 1 - \lambda_1$) and then a sequence of length W is generated according to the distribution governing the chosen model; it is assumed that every X value arises from one of the models, of which we know the distributional form but not the parameters.

3.3 Using Expectation Maximization

3.3.1 The General EM Algorithm

Now that we have some initial estimates for the parameters of the model, we can apply the Expectation Maximization (EM) algorithm in order to find the values of the parameters which maximise the likelihood of the data. The iterative procedure of the EM algorithm finds values for $\lambda = (\lambda_1,\lambda_2)$ and $\theta = (\theta_1,\theta_2)$ which locally maximise the likelihood of the data, given the finite mixture model described above.

The general EM algorithm is described by Bishop (2006: 440-1):

Given a joint distribution $p(X,Z|\theta)$, over observed variables X and latent variables Z, governed by parameters θ, the goal is to maximise the likelihood function $p(X|\theta)$ with respect to θ.

1. Choose an initial setting for the parameters θ^{old}.
2. **E-step** Evaluate $p(Z|X,\theta^{old})$.
3. **M-step** Evaluate θ^{new} given by:

$$\theta^{new} = \underset{\theta}{argmax}\, Q(\theta,\theta^{old}),$$

where

$$Q(\theta,\theta^{old}) = \sum_{Z} p(Z|X,\theta^{old})\, log\, p(X,Z|\theta).$$

4. Check for either convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let:

$$\theta^{old} \leftarrow \theta^{new}$$

and return to step 2.

Figure 3.5: The general EM algorithm

We can apply this same method to the current problem. The general EM algorithm makes use of the concept of hidden data (or 'latent variables'). It is clear that in this case, the hidden data (Z) that we must learn is the knowledge of which model (θ_1 or θ_2) each X_i value has arisen from. The EM algorithm iteratively maximises the expected log likelihood by repeatedly applying the E-step (in which we find the expected value of the log likelihood of the model) and the M-step (in which we maximise the param-

eters given the current log likelihood) of the algorithm. We continue iterating until we are sure that some convergence has been reached (or a set number of iterations have been carried out).

Bailey and Elkan (1994b) use the following notation to represent the hidden data Z:

> $Z = (Z_1, Z_2, ..., Z_n)$, where n is the number of samples (i.e. the number of samples in X)
>
> $Z_i = (Z_{i1}, Z_{i2})$, where 1 and 2 represent the motif and background models respectively
>
> $Z_{ij} = 1$, if X_i is from group j; 0, otherwise.

Behind this flurry of notation lies the idea that if $Z_{ij} = 1$, then X_i has the distribution $p(X_i|\theta_j)$. At the start, the values of Z are unknown and are treated by EM as missing data, along with the parameters λ and θ of the mixture model.

3.3.2 Finding some intial estimations of the model parameters

As explained above, the EM algorithm must be started with some initial estimation of the model parameters θ. Although theoretically these can be anything (within reason), due to the high dimensionality of the search space, we would like the estimation to be as good as possible to avoid getting 'stuck' in a local maximum that is far from the 'real' solution. In order to get as good an estimation as possible, the intial estimate of θ must be made from the input data. Example 3.10 shows how θ may easily be estimated from the X dataset.

Example 3.10:
Given the dataset $X = \{X_1,\ldots,X_{32}\}$ from Example 3.9, we wish to estimate f_{ij} for $i = 0,\ldots,W$ and $j = 1,\ldots,L$.

Estimating f_0 requires only a simple calculation of the frequencies of each letter in the Y dataset; for each letter we therefore count the occurrences of that letter and normalise using the total of all the letters. We end up with:

$f_0 = (\frac{4}{48}, \frac{21}{48}, \frac{16}{48}, \frac{7}{48})$

Estimating f_i is done in a similar fashion for each position using the X dataset. Therefore to calculate f_{1A}, we count the number of times 'A' occurs in position 1 in the X dataset, we then normalise by dividing by n (the number of sequences in the X dataset - in our running example this is $n = 32$); our estimate for f_{1A} is therefore $\frac{3}{32}$. This is done similarly for each position and letter. Our estimation of f_i follows (each column represents a position, each row a letter):

$$f_i = \begin{pmatrix} 3/32 & 3/32 & 3/32 & 4/32 & 4/32 \\ 14/32 & 13/32 & 15/32 & 14/32 & 13/32 \\ 11/32 & 12/32 & 11/32 & 11/32 & 12/32 \\ 4/32 & 4/32 & 3/32 & 3/32 & 3/32 \end{pmatrix}$$

NB: It should be noted that the results for f_0 and f_i are skewed due to the small nature of the example dataset. Using a larger real-world dataset smoothes the estimates so they are all roughly the same (this depends on the proportion of each letter within the DNA sequences – we assume this is roughly even for each letter). Strictly, we should estimate f_0 from dataset X; however this smoothing means that we can carry out the initial estimation much faster from Y with roughly the same results.

3.3.3 Finding the log likelihood of the model

What we are interested in is maximising the log likelihood of the data, given the model. To do this we must first calculate the log likelihood of the data given the parameters, following Bailey and Elkan (1994b):

It should be clear from the definition of Z that the prior probability that a particular Z_{ij} = 1 is simply λ_j:

$$p(Z_{ij} = 1|\theta,\lambda) = \lambda_j,\ 1 \leq i \leq n.$$

It is also clear that for any given i, all Z_{ij} are 0 except for one, since a sample can only belong to one group. It follows that the conditional densities of a single X_i can be written (Bailey and Elkan, 1994b):

$$p(X_i|Z_i,\theta,\lambda) = \prod_{j=1}^{2} p(X_i|\theta_j)^{Z_{ij}}$$

and similarly for its missing data Z_i:

$$p(Z_i|\theta,\lambda) = \prod_{j=1}^{2} \lambda_j^{Z_{ij}}.$$

By these two definitions and the definition of conditional probability, the joint density of a sample and its missing data can be written as:

$$\begin{aligned} p(X_i,Z_i|\theta,\lambda) &= p(X_i|Z_i,\theta,\lambda)p(Z_{ij}|\theta,\lambda) \\ &= \prod_{j=1}^{2} [p(X_i|\theta_j)\lambda_j]^{Z_{ij}}. \end{aligned}$$

If we assume that each X_i is independent, we can write the joint density of the data and all the missing information as:

$$\begin{aligned} p(X,Z|\theta,\lambda) &= \prod_{i=1}^{n} p(X_i,Z_i|\theta,\lambda) \\ &= \prod_{i=1}^{n}\prod_{j=1}^{2} [p(X_i|\theta_j)\lambda_j]^{Z_{ij}}. \end{aligned}$$

While our assumption of independence between each X_i makes the mathematics considerably simpler, it is incorrect; as we have seen, the samples in X are overlapping subsequences of width W taken from Y. We will therefore have to make sure that we take this independence assumption into account; how this is done in practice is explained shortly.

The likelihood of our unknown parameters θ and λ given the joint distribution of our input data X and the missing data Z is defined as:

$$L(\theta,\lambda|X,Z) = p(X,Z|\theta,\lambda). \tag{3.1}$$

The log likelihood is therefore:

$$log\,L(\theta,\lambda|X,Z) = \sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}\,log(p(X_i|\theta_j)\lambda_j). \tag{3.2}$$

The EM algorithm iteratively maximises the expected log likelihood over the conditional distribution of the missing data Z given the (observed) input data X and the current estimates of parameters θ and λ. This is done by repeatedly applying the E-step and then the M-step as detailed below, until some convergence condition is met.

3.3.4 E-step

The E-step of the EM algorithm finds the expected value of the log likelihood (3.2) over the values of the missing data Z and the current values of the parameters $\theta = \theta^{(0)}$ and $\lambda = \lambda^{(0)}$. Following Bailey and Elkan (1994b), we can use the fact that the expected value of a sum of random variables is equal to the sum of their individual expectations to simplify the calculation. This gives:

$$\begin{aligned}
\underset{(Z|X,\theta^{(0)},\lambda^{(0)})}{E}[log\,L(\theta,\lambda|X,Z)] &= \underset{(Z|X,\theta^{(0)},\lambda^{(0)})}{E}[\sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}\,log(p(X_i|\theta_j)\lambda_j)] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{2} \underset{(Z|X,\theta^{(0)},\lambda^{(0)})}{E}[Z_{ij}\,log(p(X_i|\theta_j)\lambda_j)] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{2} E[Z_{ij}|X,\theta^{(0)},\lambda^{(0)}]log(p(X_i|\theta_j)\lambda_j). \qquad (3.3)
\end{aligned}$$

Now we are only required to calculate the expected value of Z_{ij}. This can be found using the definition of expectation, Bayes' rule and our previous definitions for Z. Defining $Z_{ij}^{(0)} = E[Z_{ij}|X,\theta^{(0)},\lambda^{(0)}]$, we have[3]:

[3]This definition is equivalent to the two-component mixture (TCM) model described by Bailey and Elkan (1994a, 1995a)

$$\begin{aligned} Z_{ij}^{(0)} &= E[Z_{ij}|X,\theta^{(0)},\lambda^{(0)}] \\ &= 1\cdot p(Z_{ij}=1|X_i,\theta^{(0)},\lambda^{(0)}) + 0\cdot p(Z_{ij}=0|X_i,\theta^{(0)},\lambda^{(0)}) \\ &= \frac{p(X_i|Z_{ij}=1,\theta^{(0)},\lambda^{(0)})\cdot p(Z_{ij}=1|\theta,\lambda^{(0)})}{p(X_i|\theta^{(0)},\lambda^{(0)})} \\ &= \frac{p(X_i|\theta_j^{(0)})\lambda_j^{(0)}}{\sum_{k=1}^{2} p(X_i|\theta_k^{(0)})\lambda_k^{(0)}}, \qquad i=1,...,n,\ j=1,2. \end{aligned} \tag{3.4}$$

Substituting (3.4) into equation (3.3) and rearranging gives:

$$\begin{aligned} E[logL(\theta,\lambda|X,Z)] &= \sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}^{(0)}\, log(p(X_i|\theta_j)\lambda_j) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}^{(0)}\, log\, p(X_i|\theta_j) + \sum_{i=1}^{n}\sum_{j=1}^{2} Z_{ij}^{(0)}\, log\lambda_j \end{aligned} \tag{3.5}$$

Now we only need the values for $p(X_i|\theta)$ to calculate the expected log likelihood. Bailey and Elkan (1994a) state that MEME assumes the distributions for model 1 (motif) and 2 (background) are:

$$p(X_i|\theta_1) = \prod_{j=1}^{W}\prod_{k=1}^{L} f_{jk}^{I(k,X_{ij})}$$

and

$$p(X_i|\theta_2) = \prod_{j=1}^{W}\prod_{k=1}^{L} f_{0k}^{I(k,X_{ij})}.$$

where X_{ij} is the letter in the jth position of sample X_i and $I(k,a)$ is an indicator function which is 1 if and only if $a = a_k$. Although this notation looks complex, we can intuitively think of this as multiplying only the probabilities required by the sample X_i as the rest are omitted from the calculation due to the indicator function.

One additional calculation is performed after the values for Z have been calculated; the z_{ij} values for each sequence are normalised (or 'smoothed') to sum to at most 1 over any window of size W. This is done following Bailey and Elkan (1994a, 1995c):

$$\sum_{j=k}^{k+W-1} z_{ij} \leq 1,$$

for i = 1,...,N and k = 1,...,l_i - W (where l_i is the length of sequence i). This 'smoothing' is done because otherwise there is a tendency for the algorithm to converge to motif models that have long repeated sequences such as "AAAAAA" or "ATATAT" because the overlapping samples in X are not independent; the smoothing counters the faulty assumption we made that each X sequence is independent.[4]

3.3.5 M-step

The M-step of the EM algorithm maximises (3.5) over θ and λ in order to find the next estimates for them. It is clear from (3.5) that maximising over λ involves only the second term:

$$\lambda_j^{(1)} = \underset{\lambda}{argmax} \sum_{i=1}^{n} \sum_{j=1}^{2} Z_{ij}^{(0)} log\lambda_j$$

which has the solution

$$\lambda_j^{(1)} = \sum_{i=1}^{n} \frac{Z_{ij}^{(0)}}{n}, \qquad j = 1,2.$$

Maximising over θ is slightly more complex. We must solve

$$\theta_j^{(1)} = \underset{\theta}{argmax} \sum_{i=1}^{n} Z_{ij}^{(0)} p(X_i|\theta_j)$$

for j = 1,2. Bailey and Elkan define:

$$c_{0k} = \sum_{i=1}^{n} \sum_{j=1}^{2} Z_{i2}^{(0)} I(k, X_{ij})$$

and

[4] Bailey and Elkan (1995c) also include an extra 'squashing' algorithm, to be applied before the smoothing algorithm, which normalises z so that any z_{ij} is not greater than 1. It is, however, unclear why this is included as each z_{ij} value should be less than 1 by definition (3.4).

$$c_{jk} = \sum_{i=1}^{n} Z_{i1}^{(0)} I(k, X_{ij}),$$

for k = A, C, G, T, and j = 1,...,W. Again, the indicator function is used as a mathematical convenience; the variables we are not interested in are omitted from the calculation. c_{0k} represents the expected number of times letter a_k appears in positions generated by the background model and c_{jk} for j = 1,...,W is the expected number of times letter a_k appears at position j in occurrences of the motif. θ is reestimated using these count variables:

$$\begin{aligned} \theta_j^{(1)} &= (\hat{f}_0, \hat{f}_1, ..., \hat{f}_W) \\ &\therefore \\ \hat{f}_{jk} &= \frac{c_{jk}}{\sum_{k=1}^{L} c_{jk}} \end{aligned}$$

for j = 0,...,W and k = A, C, G, T. In practice, we must add a small 'pseudocount' to our estimation as maximum likelihood estimation of a multinomial random variable is subject to problems if any letter frequency becomes zero. We therefore use:

$$\hat{f}_{jk} = \frac{c_{jk} + \beta}{\sum_{k=1}^{L} (c_{jk} + \beta)}.$$

β is known as a *Laplace estimator* and is equivalent to using a prior Dirichlet distribution. The value of β must be chosen carefully: if it is too large, the effect of β will become much larger than that of the counts c, this leads to a decrease in the 'crispness' of found motifs. If β is very small, the algorithm is likely to make more definite decisions regarding which letter appears in which position; this may seem like a desired effect, however it is possible that the more definite decision taken is wrong! Bailey and Elkan (1994a) state that β should in some way represent the proportion of the dataset made up of each character. We assume that each character appears roughly equally throughout the dataset; for convenience, β has therefore been initially chosen to be 0.25 in this study.

Now that we have new estimates for θ and λ, we can use these as inputs to the E-step and continue the process until some convergence criteria has been met. After the

model has converged, the value of θ is our estimation for both the background and motif models; it is assumed that the converged motif model represents a motif in the input dataset.

3.4 From EM to MEME

As mentioned in Section 2.1.1, the MEME algorithm contains a number of novel features which extend and improve the basic EM algorithm for motif discovery. Although there has been much work on various heuristics to improve motif discovery since MEME was first described, in this study we are interested only in the features whch distinguish the basic MEME algorithm from the EM algorithm. These are an ability to discover multiple motifs due to an 'erasing' prior distribution and an ability to cope with local minima in the search space.

3.4.1 The 'Erasing' Prior Distribution

As we have already seen, the modelled dataset X contains one entry for each possible motif in the input dataset Y. In order for the MEME algorithm to find more than one 'most significant' motif, we must find a method which can remove previously discovered motifs from the dataset. The solution to this problem proposed by Bailey and Elkan (1995a,b) is to associate an 'erasing' prior distribution factor V_i with each X_i value. Each V_i value is initially set to 1, indicating that no 'erasing' has occurred. After a run of EM has converged to give us our first (most statistically significant) motif the V_i values are decreased according to the probability that X_i is part of the first motif. It is clear that the previously found motifs are effectively 'erased' from the dataset by the multiplicative effect of the prior distribution; this allows us to reestimate θ and rerun the EM algorithm to find the next most statistically significant motif.

To find multiple, different non-overlapping motifs in the input dataset, MEME uses a greedy search, incorporating information about previously discovered motifs into the model to avoid rediscovering the same motif. After the model has converged, the Z_{i1} variables represent the probability that X_i is an instance of the discovered motif. Intuitively, we would like to decrease the prior distribution according to this probability, so that a sample X_i which has a high probability of being an occurrence of a motif is

effectively cancelled out by the prior distribution V. In order to calculate the values of the prior distribution V, Bailey and Elkan (1995b) define another set of variables U which encode the positions in the input dataset that are not part of previously discovered motifs. Similarly to the calculation of Z_{ij}, we compute and store the expected values of U_{ij}; before we run the EM algorithm for the first time, each $U_{ij} = 1$. These values are updated according to the formula:

$$U_{i,j}^{(p)} = U_{i,j}^{(p-1)}(1 - \max_{k=j-W+1,\ldots,j} Z_{i,k}^{(c)}),$$

where $Z_{i,k}^{(c)}$ is the converged estimate of the missing information. It is clear from the formula that we change the estimate of X_{ij} not being part of the discovered motif by multiplying it by the probability of it not being contained in an occurrence of the previously discovered motif; we estimate this using the most probable motif of width W that would overlap it. The maximum Z_{ij} is used because occurrences of the motif cannot overlap themselves; it is in this way that we account for our assumption of independence discussed earlier. We calculate V using the following formula:

$$p(V_{i,j} = 1) = \min_{k=j,\ldots,j+W-1} p(U_{i,k} = 1),$$

that is, we estimate the probability of a motif occurrence not overlapping an occurrence of any previously discovered motif as the minimum of the probability of each position within the new motif occurrence not being part of an occurrence of a previously discovered motif (Bailey and Elkan, 1995b). The minimum is used here as, again, the probability of adjacent X_i samples not being contained in previously discovered occurrences is clearly not independent.

Now that we have calculated our prior distribution V, we simply slot this in when we reestimate Z in the E-step of the EM algorithm. We calculate Z as before and then multiply by our prior distribution to get:

$$\hat{Z}_{ij} = V_{ij} \cdot Z_{ij}$$

and use $\hat{Z}_{ij}$ instead of Z_{ij} in the M-step of the EM algorithm as described above. At this point it should be made clear that no actual erasing of the dataset (X or Y) is performed; the dataset always remains constant throughout. It is the effect of the 'erasing' prior

distribution which cancels out previously discovered motifs (Bailey and Elkan, 1994b; 1995a,b).

3.4.2 Coping with Local Minima

One of the previously mentioned problems facing any motif discovery algorithm is the large dimensionality of the search space (see Section 2.4) and therefore the high possibility of the algorithm converging to a local rather than a global optimum; clearly, we would prefer to converge to a global optimum in order to get the most likely solution. Bailey and Elkan (1994a,b) describe one method for increasing the chance of gaining a globally optimal solution. The EM algorithm is sensitive to its initial parameters, therefore by running EM a number of times and choosing the parameters which converge to the model with the highest log likelihood, we can guarantee as far as possible that we will reach a globally optimal solution. It is claimed (Bailey and Elkan, 1994b) that starting the EM algorithm with an intial estimation for λ_1 that is within a factor of 2 of the correct value is usually sufficient to converge to an optimal solution (however, it should be noted that the results supporting this claim are not shown). A lower limit for the initial value of λ_1 is given as $\frac{\sqrt{N}}{n}$ (recall that N is the number of sequences in the input dataset), the upper limit is given as $\frac{1}{2W}$; therefore, we run the EM algorithm a number of times, starting with $\lambda_1 = \frac{\sqrt{N}}{n}$, letting the algorithm converge and noting the log likelihood of the solution. λ_1 is then doubled and the procedure carried out again, repeating while λ_1 is less than $\frac{1}{2W}$. After every run of the algorithm has been completed, the algorithm is run one final time with the initial parameters that give the solution with the highest log likelihood; the solution for this run is then reported as the solution most likely to be the global optimum.

3.4.3 The Complete MEME Algorithm

Now that we have discussed the characteristic features of the MEME algorithm, the complete algorithm can be presented (Figure 3.6).

```
procedure MEME (
    Y       //dataset of sequences
    W       //motif width
    NPASSES)      //number of distinct motifs to search for
    Estimate initial θ from Y
    for i = 1 to NPASSES do
        for λ(0) = √N/n to 1/2W by × 2 do
            Run EM to convergence using current parameters
            Calculate log likelihood of converged model
        end
        Run EM to convergence using best parameters found above
        Print the converged motif
        'Erase' that motif from the dataset by updating U, V
    end
end
```

Figure 3.6: The complete MEME algorithm

3.5 Implementation of MEME

The JMeme application implemented in this work contains a number of methods that are used to implement the MEME algorithm as described above. There are two main processes handled by the JMeme application: setting up the data structures and carrying out the EM calculations. These processes and the methods used to perform them are described below.

3.5.1 Setup

The setup() method in the JMeme class calls a number of methods in order to successfully set up the data structures required for the EM algorithm. The first method to be called, addSequences(), imports a number of sequences in FASTA format (see Section 3.1.1) from a text file; these are placed in a temporary store. The next method, initialiseY(), creates a new integer array and adds the sequences from the temporary store to the array. Although the sequences could be stored as strings, it makes sense to store the sequences as integer arrays as this simplifies a great deal of the array manipulation required later by the EM algorithm. Now that we know how many sequences there are and how many possible motifs each contains, we create and initialise two arrays for U and V using the initialiseUandV() method; these are created now as they will not have to be reset throughout the running of the program, unlike the array used to store the Z

values. The final setup method, updateF(), creates an array (z) to store the values of Z_{i1} and performs the estimation of θ from the data as described in Example 3.10.

It should perhaps be noted at this point that it is unclear if the algorithm described by Bailey and Elkan (1994a,b) can be used with input sequences of different length. Setting up the program would be considerably easier if it is assumed that each input sequence is of different length; however, this assumption has not been made in the programming of JMeme to allow more flexibility in creating input datasets.

3.5.2 Carrying out EM

Each pass of the MEME algorithm is carried out by the pass() method; from here other methods are repeatedly called in order to use the EM algorithm. The pass() method firstly calculates the limits of λ_1 as described earlier. The run() method is then called with the current value of λ_1 as an argument. After the run() method finishes, the log likelihood of the model is calculated and stored along with the λ_1 value used. λ_1 is then doubled and the process repeated. After run() has finished for the final time, the λ_1 parameter is set to the value that converged to the model with the highest log likelihood and the run() method called once more to return the optimal solution. Once the optimal solution has been printed, the arrays U and V are updated to reflect the newly discovered motif.

The run() method does a large amount of the work in the JMeme application. Firstly, updateF() is called once more so that we are sure we are starting from the same settings for each run of the EM algorithm. Then, while the algorithm has not converged, the eStep() and mStep() methods are called repeatedly to carry out the EM calculations. Following Bailey and Elkan (1994a), the algorithm is said to have converged when the change in θ (calculated using the Euclidean distance) falls below a threshold of 10^{-6} or 4000 iterations have been carried out. (Although Bailey and Elkan (1994a) use a default of 1000 iterations, this has been raised to give the algorithm the best possible chance of converging without reaching the maximum number of iterations; it was deemed that the possible slight increase in processing time was not problematic.)

The eStep() method finds the expected value of the log likelihood over the conditional distribution of the missing data as described above. The array z is then updated using the current probability values for each possible motif and then smoothed, as described

above (see Section 3.3.4); once the z array has been smoothed, the 'erasing' prior distribution V is applied as described above (see Section 3.4.1).

The mStep() method maximises the expected value of the log likelihood over the conditional distribution of the missing data by separately maximising λ and θ as described above. The calculateLambda1() method carries out the update of λ_1; this is a simple summation of all the values in z, before dividing by the total number of possible motifs n. The two methods calculateC0() and calculateC1() are used to calculate the c values described above; the reestimateF() method then uses the updated c values to reestimate θ.

3.5.3 Using the JMeme application

The JMeme application may be run from the command line, using standard FASTA formatted text files as an input. The application takes three arguments:

- width - the width of motifs to search for; this should be at least 6.
- input - a FASTA formatted .txt file containing a number of nucleotide sequences to be searched for motifs
- passes - the number of passes to be made by JMeme; typically this is 1, but this can be increased to search for multiple motifs

The following examples illustrate how the JMeme application may be used.

Example 3.10

Running JMeme from the command line to search for one motif of width 8 in the file ctaaatgc10-2.txt:

```
java JMeme 8 ctaaatgc10-2.txt 1
```

Example 3.11

Sample output from the application after the above command has been entered:

```
JMeme
A.M.Kilpatrick 2009

Adding input sequences...
Finished adding sequences.

============
MEME Pass 1.
============

Maximum converged LL is: -21369.945968216583.
Initial lambda for that run was 0.0016.
Optimal solution found.

Optimal Solution: Pass 1
lambda1:
0.010074436192016997
Log likelihood: -21369.945968216583

f0:
A 0.2544
C 0.2477
G 0.2444
T 0.2535
f1:
A 0.0729 0.0201 0.7451 0.9204 0.9061 0.0733 0.0690 0.1932
C 0.8445 0.1055 0.2131 0.0171 0.0579 0.1209 0.0135 0.6642
G 0.0209 0.0163 0.0267 0.0351 0.0196 0.0152 0.9020 0.0417
T 0.0617 0.8581 0.0151 0.0274 0.0164 0.7906 0.0155 0.1009

End of solution.

================
```

3.6 Characterisation of JMeme

In order to identify the characteristics of the JMeme application, a number of tests were set up, using both synthetic and known real-world data. Using synthetic data (that is, where one or more known motifs were inserted into otherwise random 'background' data) allows the performance of the application to be measured while altering a single specific variable. Using real-world datasets where motifs have been previously proposed also allows some measure of the performance of the application against existing motif discovery methods. These tests are important, as knowledge gained from both the synthetic and real-world characterised tests allows a strategy for dealing with new uncharacterised data to be incorporated.

3.6.1 Synthetic Data

Dataset Creation

A small application to create synthetic data, CreateTestData was written in Java. This application can be used to automatically create FASTA formatted datasets, containing a user-defined number of nucleotide sequences, a number of which (again, user-defined) contain a motif chosen by the user. Like other FASTA formatted files, these datasets can be used as input to the JMeme application. There are two main steps in the creation process; firstly, given a sequence length (s) and a motif (of length m) to be inserted, a random nucleotide sequence of length s - m is created using the Random class, provided in the java.util package (Barnes and Kolling, 2006: 465). The second step is then to insert the motif at a random point in the newly created sequence and write the sequence in the FASTA format to a specified output file.

Using the CreateTestData application

The CreateTestData application may be run from the command line, using a text file for the output. The application takes five arguments:

- number of sequences - the total number of sequences to be created.
- number of motif sequences - the number of sequences to create that contain the motif.
- sequence length - the length of the sequences to be created.
- motif string - the motif to be included in the sequences
- output - a file to write the output to (typically a .txt file)

The following example illustrates how the CreateTestData application can be used.

Example 3.12

To create a dataset containing 20 sequences of length 200bp, of which 8 contain the motif GAGTTCAACTC and write it to the file out.txt:

```
java CreateTestData 20 8 200 gagttcaactc out.txt
```

Tests

Tests were set up using synthetic datasets in order to answer six important questions regarding the performance of JMeme; this allows some information to be gained on the basic characteristics of the JMeme application. As mentioned above, the use of synthetic data allows measurements to be taken, varying one aspect of the input data while keeping the rest as constant as possible. Using synthetic data also means that the motif to be searched for in the data is already known; this allows some scoring criterion to be used so that numeric comparisons can be made between two different runs of the application in various tests. The questions and the formulated tests performed are described below; the results of the tests, the scoring criteria and an analysis of the results will be presented in Chapter 4.

Does the number of input sequences have an effect on performance? A test was carried out to discover the effect of the number of sequences in the input dataset on the performance of JMeme. The test was carried out for five different motif widths, using five different motifs for each different motif width. For each motif, five datasets were constructed, containing 10, 20, 30, 40 and 50 sequences. Each test sequence was 200bp long; this length was chosen so that the results of the test would reasonably represent the results that could be expected when using new unknown data (it was decided in advance that the new unknown sequences would be 200bp long - see Section 1.3). The test datasets were constructed by using the CreateTestData application to create a dataset consisting of 50 sequences, each containing a specific motif; this would be our 50 sequence dataset. 10 sequences were then deleted from the file, which was then saved again as the 40 sequence dataset. This process was repeated until all the required datasets had been constructed. This may seem like a strange method for constructing the required datasets, however, using this process allows the datasets to be as constant as possible while changing the number of sequences; it would have been possible to simply run CreateTestData five times to create different sized datasets, but by using the chosen method we keep the random 'background' data literally the same, rather than simply regenerating from the same distribution. For each tested motif width, the performance of the application was recorded for each motif; these results were then averaged to give a single value for each motif width and dataset size.

Does the random 'background data' have an effect on performance? The effect of regenerating the background data was explored in the second test; this test aimed to answer the question of how the performance of JMeme was affected by variation in the randomly generated background data. Again, datasets containing 10, 20, 30, 40 and 50 sequences were constructed; this time, the CreateTestData application was used to regenerate the random background while keeping the chosen motif the same each time. The tests were carried out using 20 datasets for each number of sequences with a single width 8 motif; while it could not be assumed that this single test would be representative of all possible motifs for all possible widths, the results would provide a useful guideline which could then be used to judge the results of other tests. For each size of dataset, the performance of the application was recorded and the mean and variance in the results calculated. This test provides some idea of how sensitive the application is to background data; results which showed very little variation in the performance of the application while using different background sequences would indicate that the application worked well to discover motifs without relying on random quirks of the generated background data.

Does the proportion of input sequences containing the motif have an effect on performance? Both the above tests have assumed that any motif in the input dataset will be present in every sequence in that dataset. This is clearly unlikely to be the case given the current understanding of the exact role of each gene and the complex mechanism by which magnetosome particles are produced. It would therefore be wise to carry out a test in which the proportion of input sequences containing a motif was varied, in order to discover how this variation would affect the performance of the application. Once again, datasets containing 10, 20, 30, 40 and 50 sequences were constructed using the CreateTestData application; the application was used to insert motifs into varying proportions of the dataset. For each dataset size and proportion, 20 datasets were created. Again, this test was only carried out with a single width 8 motif, the same comments made above apply to this test also. For each size of dataset, the performance of the application was recorded for a number of different dataset proportions, from 100% (i.e. all sequences contain the motif) to 0% (i.e. all sequences are random). The results of the 0% tests would give some idea of how many correct motif positions could be predicted by chance; clearly, any test result which is about the same as could be predicted by chance alone is not significant.

Does performance decrease when searching for multiple motifs? The tests described above have explored the characteristics of JMeme for a single motif, however, as noted previously, one of the main features of the MEME algorithm is its ability to deal with multiple motifs through probabilistic erasing. We should therefore explore how JMeme deals with multiple motifs. For five different motif widths, datasets were created containing 3, 4 and 5 different, non-overlapping motifs. In order to ensure that each motif was non-overlapping, the datasets were created containing only one motif using CreateTestData and then other motifs added manually. Each dataset contained 20 input sequences, each of which contained each motif. The number of input sequences was chosen to be close to likely dataset sizes that would be used with the real-world data. Again, we cannot assume that the real-world input sequences all contain every motif and further, it is unlikely that, in datasets containing multiple motifs, each motif is the same width. However, the results gained from this test will be useful in helping to build a strategy for dealing with new unknown data which may contain multiple motifs. For each number of motifs contained in the dataset, the performance of JMeme in discovering all contained motifs was recorded and then averaged in order to gain some idea of any performance differences between finding the most statistically significant motif and other less significant motifs.

How does JMeme handle gapped motifs? As mentioned in Section 2.4, previous research has shown that regulatory motifs may not simply consist of a single unbroken sequence, but two sequences separated by a small gap (this appears to be the cases especially for (quasi)-palindromic motifs). If we consider the upstream sequence as a 3-dimensional structure, it is somewhat easier to see how this can be the case. It is possible that the upstream sequence we are interested in loops around before carrying on; in this case the regulating protein binds to either side of the loop but not to the loop itself, causing a gap in the motif. In order to test how any extensions to JMeme improve its ability to discover gapped motifs, we must first explore its current ability to deal with gapped motifs. A number of datasets were built, some containing 10 input sequences, the others containing 20. These datasets were constructed by running the CreateTestData application a number of times keeping both sides of the gapped motif the same each time, but the 'gap' section different (see Example 3.13); the results were combined by copying each of the created files into another 'master' file. Again, the performance of JMeme in discovering the motif and the gap was recorded for each dataset.

Example 3.13
Creating a dataset with the gapped motif TATAC - - - - GTATA. Note that the start and end sequences remain the same, while the 'gap' sequence changes between each run.

```
java CreateTestData 5 5 200 tatacacgtgtata 1.txt
java CreateTestData 5 5 200 tataccgtagtata 2.txt
java CreateTestData 5 5 200 tatacgtacgtata 3.txt
java CreateTestData 5 5 200 tatactacggtata 4.txt
copy 1.txt + 2.txt + 3.txt + 4.txt tatac----gtata.txt
```

Do we need to know the 'correct' motif width? One of the most important factors in working with unknown data is that the width of any motifs contained in the data is also unknown. We should therefore carry out some tests to see the effect of running JMeme with the 'wrong' motif width, for example, given a dataset containing a width-8 motif, we should explore the effects of using JMeme to search for a motif of different width, perhaps 10 or 11. This will give some indication of the effects we are likely to see when it comes to analysing real-world unknown data and will help to build a strategy for working with this kind of data. In order to explore the effects, a dataset containing a width-8 motif was created and a number of tests carried out in JMeme using different values for the motif width parameter. Unlike the majority of the other tests, this was not a test that could be measured numerically, however, the discovered effects were noted as it is likely that they will play a part in our strategy for working with unknown data.

Discussion

Although the tests on synthetic data described above are not exhaustive, they should give some idea of the characterisation of the basic JMeme application. The main factor in performing the above tests is the large combination of parameters which grows with each test; it would be good to perform each test several times, varying each and every possible parameter, however, this is simply not possible given the time constraints of the project. In most cases, the tests were carried out with data which was deemed to be representative of data to be used in real-world tests. Clearly, tests with more extreme data to explore the limitations of the JMeme application would be desirable given more time, however it was decided to spend more time working with known real-world data and devising a strategy for working with uncharacterised real-world data. The tests

carried out with previously characterised real-world data are described below.

3.6.2 Real-world Characterised Data: CtrA Metabolism

As mentioned in Section 2.3.3, the study by Lan (2008) concentrated on the cell transcriptional regulator (CtrA) metabolism of *Magnetospirillum magneticum* and identified two new potential motifs by carrying out an orthology analysis between *M. magneticum* and *Caulobacter crescentus* (a closely related alphaproteobacteria): a width-6 ungapped motif and a longer width-15 gapped motif were identified. The same data was used in this study in order to further test the JMeme application. The aim of this test was not only to confirm that JMeme was working as expected but to validate the potential motif sequences identified by Lan (2008); the results of the tests and an analysis of the results will be presented in Section 4.1.2.

Dataset Creation

Two real-world datasets (referred to as BLAST and LAN) were created in order to further test the implemented JMeme application using the CtrA metabolism. This is one area which has been studied previously, therefore results gained here should be confirmed by existing knowledge.

BLAST In order to discover the genes involved in CtrA metabolism in AMB-1, a list of genes known to be involved in CtrA metabolism in a closely related species of alphaproteobacteria (*Caulobacter crescentus*) was used (Laub, *et al.*, 2002). A BLAST search was carried out for each gene in the list; this returned a number of similar genes in the AMB-1 species. Although some genes returned a large number of matches, only the best results were kept. A similarity score of 1E-8 was used as a threshold; genes that scored less than this (i.e. that were better matches) were kept and the rest discarded. After this process was carried out for each gene in the *C. crescentus* list, 119 genes deemed to be involved in CtrA metabolism in AMB-1 were remaining; these genes were extracted from the AMB-1 .gbk file using the GBKReader application (see Section 3.1.4). Lists of the genes included in all of the real-world datasets used are presented in Appendix A.

LAN As a confirmation of the above method, the genes from the BLAST dataset were compared to the list of genes built using comparative genomics by Lan (2008). Almost all of the genes in that dataset had been found by the BLAST search above; of these, 20 were used by Lan (2008) to explore the CtrA metabolism. The dataset of 20 genes used by Lan (2008) was reconstructed using the GBKReader application so it could be used as an input to JMeme.

Tests

The JMeme application was run with the LAN dataset over a range of different possible motif widths. It was decided not to use the BLAST dataset because of the large number of sequences in the dataset. Although it is possible that the sequences in the BLAST dataset that are additional to those in the LAN dataset may support any motifs found in the LAN dataset, there are the disadvantages of a much longer required computation time and a high number of potential 'noise' sequences; therefore, only the LAN dataset was used in order to get the best results. It was expected that runs with shorter widths would discover the ungapped motif, with additional runs over similar widths validating the result. Runs over a range of longer widths were performed in order to discover the longer gapped motif. Additional runs over similar widths were performed in order to validate the result; this would be important as Lan (2008) notes that the signal for the longer gapped motif is weaker than that for the ungapped motif and it therefore will likely be harder to find.

3.6.3 Real-world Characterised Data: Iron Metabolism

The study by Lan (2008) also aimed to confirm the work carried out by Rodionov, *et al.* (2006) on the iron regulation metabolism in *M. magneticum* (see Section 2.3.3). Lan (2008) obtained a width-21 motif which was very similar to the width-19 motif proposed by Rodionov, *et al.* (2006). Again, the aim of this test was to validate this result and further test the JMeme application to make sure it was working as expected; the results, analysis and a potential new motif will be presented in Section 4.1.3.

Dataset Creation and Tests

As noted in Section 2.3.2, Rodionov, *et al*. (2006) provide a list of 17 genes thought to be important in regulation of iron production; the upstream regions of these 17 genes were used for the study by Lan (2008). The same genes were extracted from the AMB-1 .gbk file using the GBKReader application; these form the ROD dataset. This dataset has been analysed previously and one possible motif discovered; it therefore provides a starting point for motif discovery. It is possible that different motifs or even no motifs may be found with the JMeme application as the previously discovered possible motif was found using the GenomeExplorer application.

Again, the JMeme application was run over a range of different possible motif widths in order to discover the proposed motif; it was hoped that additional runs would again validate the result as it is likely that the signal for the motif is quite weak.

3.7 A Strategy For Analysing Real-world Uncharacterised Data

Knowledge gained from the tests described above was used to devise a strategy for working with real-world unknown data. This is important because it is this type of data that we are most interested in; in order to gain meaningful results and use the available time most efficiently, a structured way of working with this data is required.

One important piece of information which is unknown in this situation is the motif width. Conducting tests with synthetic data in which the correct width of the motif (the 'target' width) is different to the width parameter given to JMeme (the 'tested' width) indicated that, in general, motifs which had strong results when the tested width was correct remained strong when the tested width was incorrect. When the value of the tested width used was smaller than the target width, it was found that a number of positions in the target motif were cut off from the result motif; when the tested width is increased to a value larger than the target width, the result motif generally contains the target motif and a number of weaker 'noisy' positions (see also Section 4.1.1). From these results, it is clear that, in order to find the 'correct' motif width, a range of widths must be tested; sequences which appear strong over a number of different widths are more likely to be significant.

When the tested motif width is significantly smaller than the target, it was found that multiple passes of JMeme tended to find different fragments of the target motif on each pass. Assuming each fragment is more significant than the background data this result should not be surprising; as one fragment is probabilistically erased, subsequent passes of JMeme should find the remaining fragments of the target motif. It should therefore be noted that the returned motifs may not themselves be motifs, but smaller fragments of a longer motif. It follows from this that the effect of shifting (explained in Section 2.4) plays a very important part in motif discovery when more than one pass of JMeme is used. For example, consider a target motif of width 12, of which, perhaps due to some quirk in the background data, only the first 8 positions were picked up correctly by the first pass. Before the second pass, these positions will be probabilistically erased, leaving the last 4 positions, which may be picked up by the second pass of JMeme. Clearly, the effect of this hazard could be reduced by reducing the effect of shifting on the first pass, however, it must be kept in mind when working with unknown data as obviously we do not have any prior information about the length of any motifs contained within the dataset. Testing multiple passes of JMeme appeared to show that the results of initial passes were more significant (the results of this test are presented in Section 4.1.1); while this does not mean that results gained on subsequent passes are insignificant, it may indicate that disputes between results gained on two different passes may be resolved by choosing the more significant earlier pass.

Exploration into the effect of the motif length on the performance of JMeme showed that a width of around 6bp is a lower limit when discovering motifs; motifs shorter than this tended to get lost in the background data and the number of correct positions drops to a level around that which could be expected by chance alone (the results of this test are presented in Section 4.1.1). From these results, it makes sense to set a motif width of 6bp as the lower limit for testing different motif widths.

The described strategies can be summarised:

- Test a range of motif widths, with a width of 6bp as the lower limit. Any motifs which are strongly conserved over a range of different motif widths are likely to be more significant.
- Be aware that longer motifs can be returned as a number of shorter motifs, either as initial passes over different tested motif widths or subsequent passes over the same tested motif width.

- It may be possible to join a number of shorter motifs together depending on the strength of the results; bearing in mind that motifs may be (quasi-)palindromic may help.
- In general, earlier passes of JMeme tend to be more significant, however, further passes may help to confirm results.

3.8 Motif Discovery in Uncharacterised Data

3.8.1 Dataset Creation

Three datasets were constructed to be used for the discovery of motifs important in magnetosome regulation.

RAST The RAST (Rapid Annotation using Subsystem Technology) online tool[5] (Aziz, *et al.*, 2008) can be used to automatically annotate and classify genes within an uploaded genome. This procedure was carried out with the AMB-1 genome. Genes contributing to the *Fe_stress subsystem were regarded as being possibly important in magnetosome production; these genes would be a possible dataset. It was discovered that the RAST annotation is performed automatically by machine and therefore does not use the same locus tags as found in the GenBank file; the *Fe_stress genes were manually mapped from the RAST analysis to the GenBank file in order to find the corresponding locus tag for each gene. Having mapped a list of locus tags, the GBKReader application was used to create a dataset of the 23 genes returned by RAST.

From an initial scan of the gene products, it appears that the genes selected by RAST may be regulator genes, that is, genes whose product is a regulation protein for genes which produce magnetosomes (see Figure 1.5)

IRON, FD Two additional datasets were constructed using the GBKReader, performing a naive search for genes with either ‘iron’, ‘ferric’, ‘ferrous’ or ‘Fe^{2+}’ as part of its product (IRON), or with ‘ferredoxin’ (iron-sulphur proteins involved in transferring electrons in a number of metabolic functions) as part of its product (FD). The IRON dataset contains the 25 genes found, while the FD dataset contains 34 genes.

[5] http://rast.nmpdr.org/ (Accessed 21st July 2009)

Datasets were constructed as described in Section 3.1.4). Constructing the RAST dataset consisted of identifying genes contributing to the *Fe_stress subsystem in *M. magneticum* using the RAST online genome annotation tool and creating a dataset of those genes using the GBKReader application. The IRON and FD datasets were constructed using a naïve keyword search in the GBKReader application.

3.8.2 Tests

For each of the three datasets, the JMeme application was run over a range of possible different motif widths following the strategy for dealing with uncharacterised data as described in Section 3.7; different numbers of passes were also tested for each dataset. The results for each dataset were noted and well-conserved sequences recorded as being possible motifs. The results and an analysis will be presented in Section 4.1.4.

3.9 Implemented Extensions: JMemePlus

Two extensions to JMeme were designed and implemented to test whether extensions to the JMeme algorithm improved the performance for discovering regulatory sequences in real-world data. The extensions implemented were a different method for choosing the optimal run of the EM algorithm and a method to incorporate prior beliefs when searching for regulatory sequences. These additions were added to create a new application, referred to as JMemePlus. A brief analysis of the extensions will be presented in Chapter 4.

3.9.1 Choosing Optimal EM Models

Following Bailey and Elkan (1994a,b; 1995a,b,c), the original JMeme algorithm ran through a number of different values for λ_1, choosing the converged motif as the model with the highest converged log likelihood. During the tests on synthetic data, it was noted that on second and subsequent passes of JMeme, motifs were returned where every letter for every position had a value of 0.25, that is, no letter was more likely than any other for any position within the converged motif. Three possible reasons for this outcome were proposed. Firstly, it may be the case that too few iterations

were being performed in the EM algorithm part of the application and therefore the algorithm did not have a chance to make the motif converge to a true value. Secondly, it could also be the case that the effect of the erasing prior distribution effectively cancels out everything and makes this '0.25 motif' more likely than anything else. Finally, it could be the case that the β parameter requires adjustment (recall that too high a value for β would cancel out any trends in the data). Additional tests were carried out to try and explain this problem. The first situation can easily be tested by forcing JMeme to perform additional iterations, rather than stopping after θ appears to have converged (it could be the case that θ was converging exceptionally slowly). The JMeme application was altered to force 100,000 iterations of the data, however, this was found to have no effect on the converged motif; in addition, this substantially increased the computation time. This result should perhaps not be surprising as one advantage of the MEME algorithm proposed by Bailey and Elkan (see Section 2.4) is that the algorithm converges in a known and predictable way, therefore a large forced increase in the number of iterations is not likely to make much of a difference to the final result. Nevertheless, it was still deemed to be important to rule this possibility out. Given that the number of iterations was not too low, it seems more likely that the erasing prior distribution or the β parameter is a factor in this problem. However, it was noted that '0.25 motifs' were more likely to be returned on the second and subsequent passes of JMeme, indicating that it is more likely to be a side-effect of the erasing prior distribution. Forcing JMeme to return suboptimal solutions showed that on these subsequent passes, motifs were occasionally being found, but with a lower log likelihood; in order to continue with the test, these suboptimal solutions were used in motif discovery. It is, however, desirable for the application to automatically return these solutions, even when the solution has a lower log likelihood. One solution to this problem would be to calculate the information content of the converged motif model and use that as a score to choose between the different runs; after the runs of EM were complete, the information content would be calculated and the motif model with the highest information content returned. This seems a reasonable solution, as we would expect '0.25 motifs' to have a much lower information content than the other 'more interesting' motifs.

Calculating the Information Content of a Motif Model

Given an array of probabilities of each letter for the background model and each position in the motif model, we can calculate the information for each letter for each column. Firstly, the information for each column i is calculated[6]:

$$IC_i = \sum_{j=\{A,C,G,T\}} f_{ij} log_2 \left(\frac{f_{ij}}{f_{0j}} \right), \tag{3.6}$$

for $i = 1,...,W$. We calculate the information for each letter i in column j by multiplying:

$$Inf_{ij} = f_{ij} \times IC_i.$$

The total information content for a motif model IC_{total} can be calculated simply by summing the information over every letter and column:

$$IC_{total} = \sum_{i,j} Inf_{ij}.$$

The inclusion of the background model f_0 in (3.6) allows us to gain some idea of how 'surprised' we should be to see a letter in a certain position, for instance, if the letter G is particularly prevalent in the background model, we should be less surprised to see the letter G in the motif; it follows that we gain more information the greater the difference between the motif and the background model.

Implementation

The implementation of the extension described above is fairly straightforward. A method, getIC(), was written to return the information content IC_{total} of the current motif model. This may be called at any point in the program, however, it is most useful to us once we have run the EM algorithm to convergence. Changing JMeme to choose the model with the highest information content rather than the highest log likelihood is mainly swapping the calculateLL() method, which returns the log likelihood,

[6]Information can either be calculated using log_2, giving an answer in bits, or ln, which gives an answer in 'nats' (Bishop, 2006: 50). The only difference is a factor of $ln(2)$. To compare with the information scores of MEME online, we use the former calculation here.

with the getIC() method; apart from these changes, the mechanism for choosing the model remains the same.

3.9.2 Incorporating Prior Beliefs

The MEME algorithm as described by Bailey and Elkan (1994b, 1995b) contains no way of using prior information to help guide the algorithm to a result. Although basic knowledge about motifs can help (for instance, it might be known that some motifs are shorter than others) guide interpretations of the results, there is no method by which one could use prior information of the kind *"First two positions are AT, next three positions are unknown, next position is C, next four positions are unknown, last position is T"*. The concept of the 'energy' of a model given a prior belief has been used in work carried out by Werhli and Husmeier (2007); although it is used in that study to reconstruct regulatory networks with Bayesian networks, it seems reasonable that the same concept could be applied in this study, using the energy of a motif as a score on which to base further iterations of EM.

Using the Energy of a Motif

Given a belief B similar to the example above, which we believe to be similar to the motif we are searching for, we create an array of $4 \times W$ elements. In this array, a value of 1 in a column represents the letter we believe to be in that position, all other elements are 0. For instance, a belief of (A,C,G,G,T) is represented:

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and a belief of (A,A,*unknown*, *unknown*,T) may be represented:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

We also have a motif X, which we wish to calculate the energy of, similarly represented as a 0/1 array.

We define the energy of motif X, $E(X)$[7], as:

$$E(X) = \sum_{i,j} \left| B_{ij} - X_{ij} \right|.$$

We define γ as a measure of how confident we are in this belief (a value between 0 for no confidence and 1 for complete confidence).

We define a normalising constant $C(\gamma)$ as:

$$C(\gamma) = 4^{(W-1)}(3e^{-2\gamma} + 1).$$

Using these definitions, we can define the probability of a motif X, given a belief B, $p(X|B)$, as:

$$p(X|B) = \frac{e^{-\gamma E(X)}}{C(\gamma)}.$$

We now have a value that we can calculate for each X_i sample, which we intuitively think of as getting larger the closer X_i is to our belief B. Since the EM algorithm is based on increasing the log likelihood, it would make sense to introduce γ (that is, the parameter for our confidence in the belief B) into the equation for marginal likelihood (3.1) to give:

$$L(\theta, \lambda, \gamma | X, Z) = p(X, Z | \theta, \lambda, \gamma).$$

However, decomposing the right hand side as before proves problematic as we have to define $p(X,Z|\theta,\gamma)$, which is not trivial. Due to the problems with this method, an alternative way of incorporating our belief was designed, keeping our equation for marginal likelihood the same but modifying the way that the count variables c are calculated when we reestimate θ_1. This is intuitively similar, in that we are modifying the model by incorporating our beliefs, but we avoid complicated decompositions of the type seen above. Recall that we originally calculated c_{jk} using only Z values:

$$c_{jk} = \sum_{i=1}^{n} Z_{i1}^{(0)} I(k, X_{ij}) \qquad k = \{A, C, G, T\},\ j = 1, ..., W.$$

[7]Not to be confused with the *expectation* of X

This calculation is modified to incorporate our belief B:

$$c_{jk} = \sum_{i=1}^{n} Z_{i1}^{(0)} p(X_i|B) I(k, X_{ij}) \qquad k = \{A, C, G, T\},\ j = 1, ..., W,$$

and we continue the EM process as before.

Implementation

Again, the implementation of this extension is fairly straightforward. The $p(X_i|B)$ values for each X_i are calculated in the setup procedure; our belief about the nature of the motif does not change and similarly the X_i samples do not change, therefore we can assume that these probabilities are constant throughout one pass of the application. Once these have been calculated, it is a reasonably simple task to slot these probabilities in the recalculation of c_{jk}.

3.9.3 Using JMemePlus

The JMemePlus application may be run from the command line, using standard FASTA formatted text files as an input. The application takes five arguments:

- width - the width of motifs to search for; this should be at least 6.
- input - a FASTA formatted .txt file containing a number of nucleotide sequences to be searched for motifs
- passes - the number of passes to be made by JMeme; typically this is 1, but this can be increased to search for multiple motifs
- beta - the value of the β parameter; typically, a value of 0.25 can be used, however, when using beliefs this must be decreased (see Section 4.2.2)
- belief - a string representing some prior belief about the motif, or an empty string.

The following examples illustrate how the JMemePlus application may be used.

Example 3.14

Running JMemePlus from the command line to search for one motif of width 8 in the file test.txt, with a β value of 0.25 and no prior belief:

```
java JMemePlus 8 test.txt 1 0.25 "" 
```

Example 3.15

Running JMemePlus from the command line to search for one motif of width 8 in the file test.txt, with a β value of 3.8E-6 and a prior belief of 'xTAAATGC':

```
java JMemePlus 8 test.txt 1 3.8E-6 "xTAAATGC"
```

Chapter 4

Results and Evaluation

This chapter presents the results of the tests described in Chapter 3. The results are also critically evaluated and compared to other related work. An evaluation of the extensions implemented in JMemePlus is also presented.

4.1 Test Results and Evaluation

4.1.1 Synthetic Data

As mentioned in Section 3.6, the tests using synthetic data aimed to answer a number of questions regarding the basic performance characteristics of the JMeme application. The results of these tests are presented here. The advantage of using synthetic data for testing is that we know in advance the motif that should ideally be returned and therefore can apply some scoring rules in order to make some numeric comparisons between runs of the JMeme application. In general, the rules for scoring returned motifs are as follows:

- The position matrix representing the returned motif is studied; for each position in the returned motif, the letter with the highest proportion is regarded as being the letter for that position.
- Each correct position receives a score of 1 (a position where the returned motif matches the target motif is regarded as being a correct position). There is an exception to this rule when two letters are equally likely (as often occurs in the

test for multiple motifs); in this case, the position receives a score of 0.25. A score of 0.25 rather than 0.5 was chosen as it was thought a large number of 50/50 cases could sum to give a misleadingly high score; using 0.25 gives some credit for narrowing the choice to two letters but still penalises the inability to make a decision either way.

- Following Bailey and Elkan (1994b), returned motifs are shifted in order to align as best as possible with the target motif. For example, given the target motif CTAAATGC and the returned motif TAAATGCC (as shown in Figure 2.1), without shifting, only three positions are correct. When the returned motif is shifted one position to the right, seven positions are correct.
- After shifting, the number of correct positions is recorded and divided by the motif width in order to give a percentage figure of correct positions. It is this figure that will mainly be used to judge the quality of the returned motif.

Does the number of input sequences have an effect on performance?

This test was carried out as described in Section 3.6.1; the results of this test are shown in Figure 4.1. It can be seen that in general there is an upward trend, showing that increasing the number of input sequences also increases the average percentage of correct positions in the returned motif. However, there are two tested motif widths which appear to be exceptions to this trend. The first exception is a tested motif width of 4bp; in this test, the number of correct positions stayed around 30% and never reached higher than 40%. In contrast, for other tested motif widths, the average percentage of positions is almost always over 60%. This result seems to imply that shorter motifs are harder to find within the background noise. Indeed, it seems that the results of the test with motif width 4 are only as good as could be achieved through guesswork alone. It is interesting to note that the online MEME server had similar problems with finding the correct motif; this seems to confirm the belief that motifs shorter than around 6bp are very difficult to find because of their short length. While the results of the test with motif width 6 are much better in general than those with motif width 4, it is also a noticeable exception to the general trend. Although the average percentage of correct positions jumps to over 90% with 30 input sequences, adding more input sequences reduces the percentage of correct positions to around 80% with 50 input sequences. There does not seem to be an obvious explanation to this exception, although it is pos-

sible that after the test with 30 input sequences, input sequences were added which contained noisy versions of the target motif by chance. This would explain the falling performance rate for this tested motif width while in general, increasing the number of input sequences increases the average percentage of correct positions. Generalising from the data, we might expect that increasing the number of input sequences would not increase the performance of JMeme forever; we would expect there to be a point where adding extra input sequences makes no difference to the performance. An upper limit of 50 sequences was chosen as in practice the number of input sequences would not normally exceed this figure.

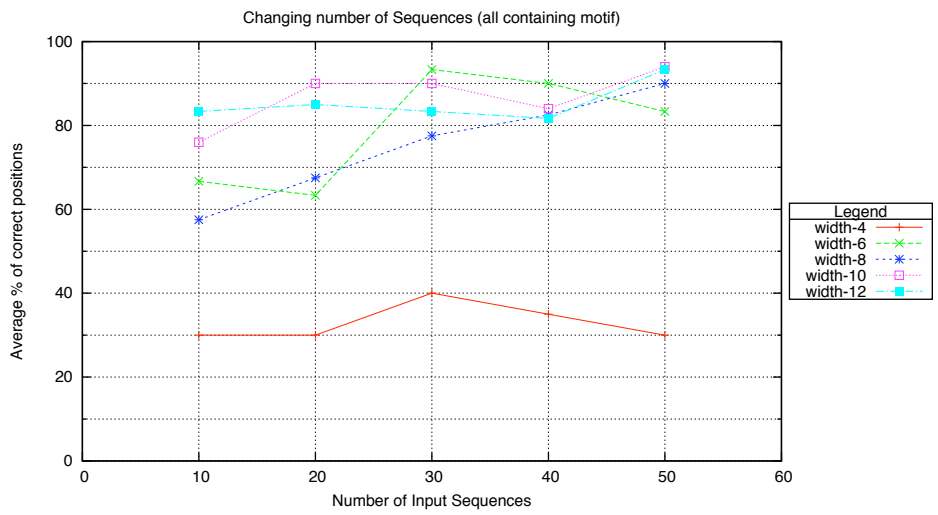

Figure 4.1: Testing the effect of increasing the number of input sequences

Does the 'random background' data have an effect on performance?

This test was carried out as described in Section 3.61; the mean results of this test are shown in Figure 4.2, the variance results shown in Figure 4.3. The mean results of this test seem to show that in general there is a slight trend to increase the percentage of correct positions as the size of the input dataset increases; this would be expected given the results of the previous experiment. Again, simply increasing the number of sequences would not be expected to increase the performance of JMeme forever; increases in the number of input sequences also increases the amount of computation time taken by JMeme. Studying the variation graph, it can be seen that, in general,

the variance decreases as the number of input sequences increases. Taken together, the mean and variance graphs suggest that by increasing the number of input sequences, a higher average percentage of correct positions can be achieved and that this average is more consistent over a number of different datasets (i.e. there is a lower variance). This result should be expected, as it seems reasonable that (in the case of synthetic data, at least) increasing the number of input sequences also increases the number of occurrences of the motif within the dataset.

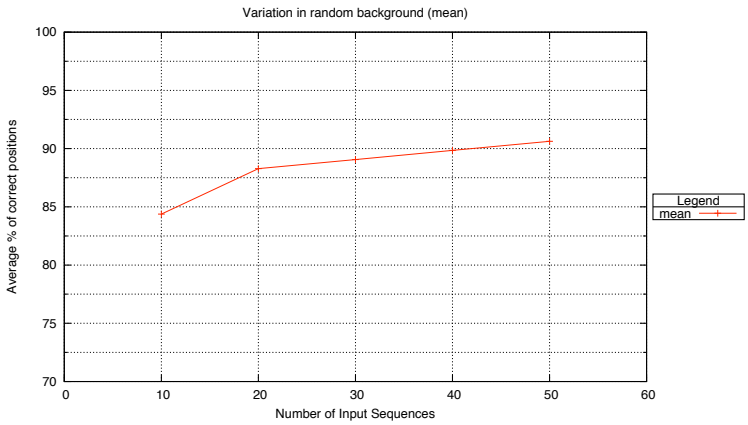

Figure 4.2: Testing the mean number of correct positions while changing the background data

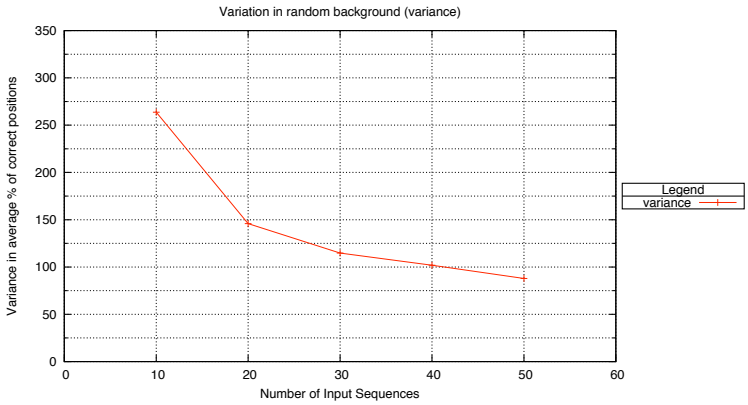

Figure 4.3: Testing the variance in the mean number of correct positions while changing the background data

Does the proportion of input sequences containing the motif have an effect on performance?

Again, this test was carried out as described in Section 3.6.1; the results of this test are shown in Figure 4.4. It can be seen from Figure 4.4 that there is a clear trend of decreasing percentage correct positions when the proportion of input sequences containing the motif is also decreased. Again, this result is as expected as it seems reasonable that the JMeme application will not be able to discover a motif if there are no motifs and similarly, increasing the proportion of input sequences containing the motif makes it more likely that a motif will be discovered, until eventually all the input sequences contain the motif and we get similar results to those gained in the first test above. Once again, there are a number of outlying points, notably for the dataset containing 10 sequences. It could be that there is not enough data to base any predictions on; it could also be due to quirks in the dataset, although this is unlikely given the number of tests performed. As may be expected, the figures for the dataset containing 50 sequences are generally the best and as the total number of sequences in the data decreases, performance decreases slightly regardless of the proportion of input sequences containing the motif. One point of note is the values for datasets where 0% of the sequences contain the motif; the JMeme application was run and the results aligned to give the best score. These results show that the random background gives a result which can be aligned to give on average just over 30% correct positions, even though there are no motifs present in the dataset. From this result, we can state that test results which only give 30% correct positions are not significant as this baseline value can be achieved even when no motifs are present in the dataset. Bailey and Elkan (1994b) claim that, using MEME, "experiments show that even when only 20% of the sequences in a dataset contain a motif, the motif can often still be characterised well". It is, however, unclear what is meant by "characterised well". From the results shown in Figure 4.4, with input datasets where 20% of the sequences contain the motif, on average, JMeme finds at least 50% of the motif positions correctly, a figure which rises to around 62% when 50 input data sequences are used. When less than 20% of the input sequences contain the motif, the performance of JMeme tails off to around 30%. Although it is clear that there is something significant happening once 20% of the input sequences contain a motif, without a clear description of what Bailey and Elkan mean by "characterised well", it is hard to compare JMeme to the algorithm described by Bailey and Elkan (1994b). At present it appears that JMeme is reasonably sensitive

to 'noise' sequences; clearly, there is a decrease in performance when more noise sequences are added. This therefore means that we would get better results if we knew that the input sequences definitely contained a motif, however, this is one piece of information which is not likely to be available before motif discovery is carried out.

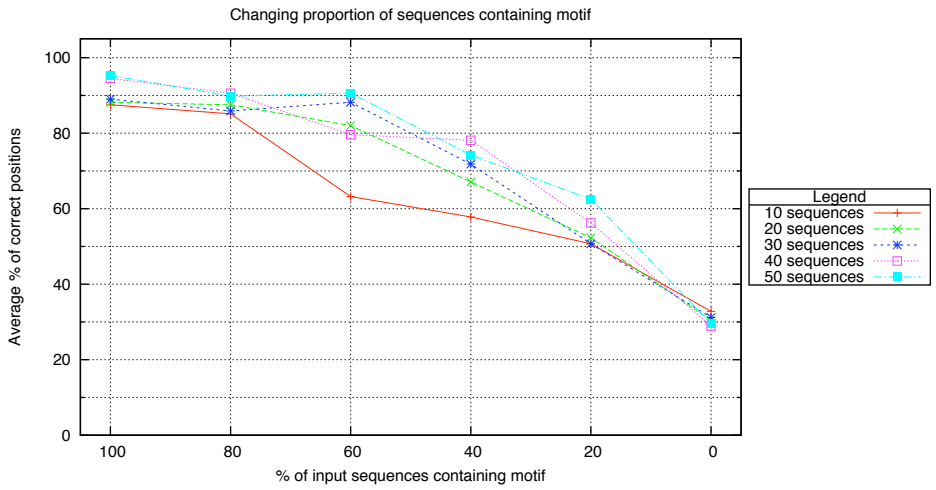

Figure 4.4: Testing the effect of changing the proportion of input sequences containing the motif

Does performance decrease when searching for multiple motifs?

This test was carried out as described in Section 3.6.1; the overall results are shown in Figure 4.5. It can be seen that in general as the number of motifs in the dataset increases, the average percentage of correct positions decreases. One notable exception to this trend is the width-8 dataset, where the percentage of correct positions increases slightly with increasing numbers of motifs. Given the trends for other tested motif widths, this is likely to be an anomaly in the data; perhaps more testing would clarify this trend. As noted above, a slight exception was made to the scoring procedure in this test; positions which were equally split between two letters were awarded a score of 0.25 if one of the letters was the correct one for that position. As well as the general trend for average scores to decrease with increasing numbers of motifs contained within the dataset, another trend appeared to emerge during this test. Until now, JMeme has run the EM algorithm using a number of different settings for the λ pa-

rameter, returning the converged motif with the highest log likelihood score (following Bailey and Elkan, 1994b). However, in this test it was discovered that JMeme was returning motifs where every letter for every position had a value of 0.25, that is, no letter was more likely than any other for any position within the converged motif. Forcing JMeme to return suboptimal solutions showed that on these subsequent passes, motifs were occasionally being found, but with a lower log likelihood; in order to continue with the test, these suboptimal solutions were used in motif discovery. One proposed solution to this problem is to judge the different runs of JMeme using the information content of the motif rather than the log likelihood. While this may provide different solutions where the originally returned solutions *were* optimal, it would very likely solve the problem of returning '0.25 motifs' as we would expect these motifs to have a much lower information content than the suboptimal motifs used in this test. This solution was implemented as an extension to the JMeme application (see Section 3.9). As explained in Section 3.7, it can be seen that shifting on the first pass of JMeme has a large influence on the subsequent passes; it would therefore be helpful to reduce as much as possible the effect of shifting in order for subsequent passes to converge to the best results.

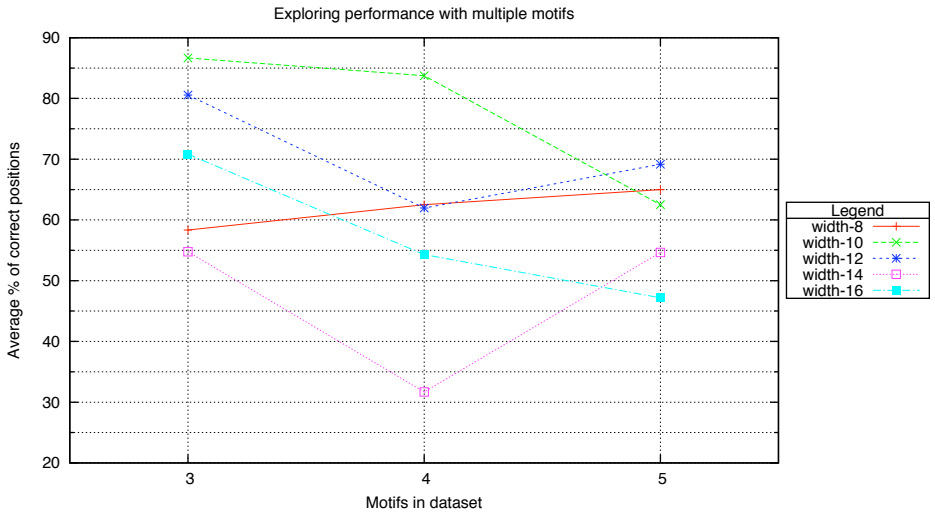

Figure 4.5: Exploring performance changes for different numbers of motifs

How does JMeme handle gapped motifs?

This test was carried out as described in Section 3.6.1. Motifs were not scored in this test; the purpose was simply to explore how JMeme handles gaps within motifs. In general, it was found that JMeme handles gapped motifs reasonably well: Figure 4.6 shows two examples of gapped motifs found with JMeme. It can be seen that in general JMeme gives strong results for the side sequences while the gap sequence is much less strong. This result should be expected as it seems reasonable that, given an ability to discover the strong side sequences, the gap should be much weaker, if not simply a 'noise' signal. Again, the effect of shifting can be seen in some of the results; all of the motifs tested here were symmetrical in structure (although not necessarily palindromic) in order to explore the effect of shifting as far as possible. It must be noted, however, that real-world gapped motifs need not be symmetrical (nor, for that matter, palindromic) so we cannot use a stronger signal at one side of a gap to infer that any shifting has occurred on the other side. Bailey and Elkan do not make a distinction between gapped and ungapped motifs in their evaluation of the MEME algorithm; we therefore cannot compare the results returned by JMeme.

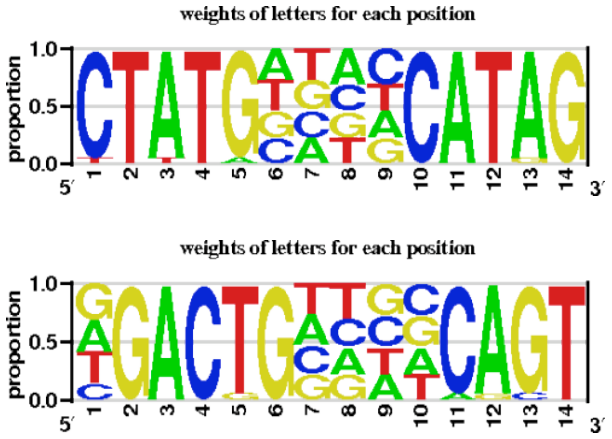

Figure 4.6: Exploring how JMeme handles gapped motifs. The first logo shows a strong signal of 5 positions at each side a 'gap' of 4 positions in the middle where no letter has a clear majority. The second logo illustrates shifting in gapped motifs.

Do we need to know the correct motif width?

This final test was carried out as described in Section 3.6.1; the effects noted are described in Section 3.7, but repeated briefly here. It was found that, in general, motifs which were strongly characterised when the correct test width was used remained so

when the incorrect width was used. When the tested width was smaller than the target motif, it was found that some positions in the target motif were cut off from the result motif; when the tested width was longer than the target motif, the result motif generally contains the target motif and a number of weaker noisy positions. These are illustrated in Figure 4.7. It therefore makes sense to test a number of similar motif widths when dealing with unknown data; positions which are well conserved over a number of different widths are more likely to be significant.

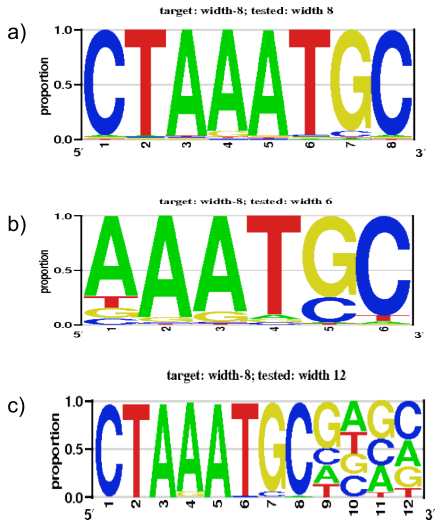

Figure 4.7: a) The full target width-8 motif; b) Using a shorter test width tends to cut positions off and we find only a section of the full motif; c) Using a longer test width finds the full target motif and also a number of 'noise' positions

4.1.2 Real-world Characterised Data: CtrA Metabolism

Tests were carried out to identify and validate the potential CtrA motifs proposed by Lan (2008) as described in Section 3.6.2. Lan (2008) proposes two CtrA motifs, one of which is consistent with the confirmed CtrA motif in *Caulobacter crescentus* (5'-TTAA-N7-TTAA-3') and one further motif which appeared to have a strong signal (5'-GGAATT-3').

Results gained in the tests carried out appear to confirm the presence of the gapped motif (TTAA-N7-TTAA); Figure 4.8 shows the logo representing the width-15 test on the LAN dataset. Several other larger widths were also tested in order to confirm this result and the majority of the tests pointed towards this sequence being a reasonably strong motif. Looking again at Figure 4.8, it can be seen that following the first TTAA

sequence, there is a further TTT fragment which seems to be just as strong as the first TTAA sequence. Following this fragment, there are four positions which can be classed as 'noise' (the probability for each letter is reasonably similar), before the second TTAA sequence. This seems to imply that the TTT fragment is also significant in the motif. We can therefore state that it appears that 5'-TTAATTT-N4-TTAA-3' is a potential CtrA motif in *Magnetospirillum magneticum*. This is still consistent with the motif discovered by Lan (2008) and the previously confirmed CtrA motif in *C. crescentus*.

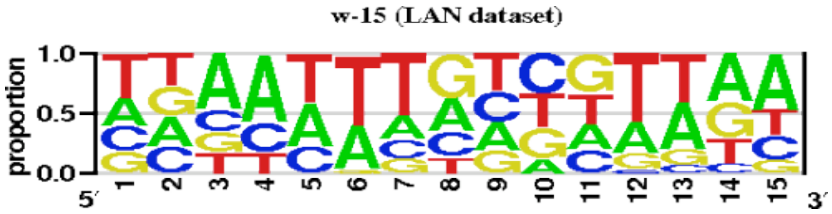

Figure 4.8: Result of the width-15 test on the LAN dataset

The test results for the shorter ungapped motif were similar to those proposed by Lan (2008); Figure 4.9 shows the logo representing the width-6 test on the LAN dataset. This result can be interpreted as a shifted version of the proposed motif, however, additional tests on several larger widths also failed to find the exact motif proposed by Lan. As a comparative test, the same dataset was used as an input to the online MEME suite. The online MEME tool found the motif as proposed by Lan (2008), therefore we can be reasonably sure that this is indeed the correct motif and that for some reason JMeme is unable to find this particular motif. In an attempt to find this particular motif with JMeme, an additional dataset, LAN7, was created, containing the 7 genes shown to contain the motif by the online MEME tool. The result of the width-6 test on the LAN7 dataset is shown in Figure 4.10. It can be seen that 5 out of the 6 positions match the proposed motif; again, however, additional tests with larger widths failed to find the first position.

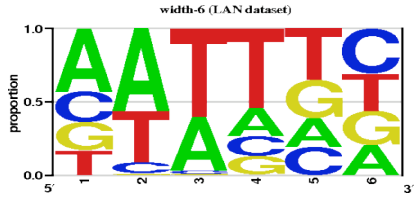

Figure 4.9: Result of the width-6 test on the LAN dataset

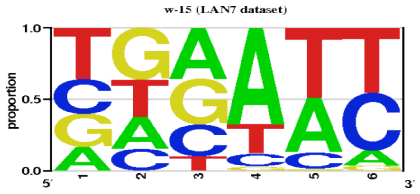

Figure 4.10: Result of the width-6 test on the LAN7 dataset

No CtrA motifs have previously been confirmed in *M. magneticum* experimentally, however, we can conclude from the above tests that the longer gapped motif proposed by Lan (2008) appears to be confirmed by JMeme. JMeme seems to show that there is an additional motif fragment which may be significant, however, even this fragment conforms to the motif proposed by Lan (2008) and is very similar to the confirmed CtrA motif in *C. crescentus*. Although Lan (2008) states that the signal for the shorter ungapped motif is stronger, JMeme fails to find the exact proposed motif, even when the dataset is known to contain the motif in 100% of the input sequences. One possible explanation for this is simply that the motif is too short and that this makes it harder for JMeme to find amongst the background sequences; in contrast, it may be the case that the online MEME tool contains extra heuristics for dealing with relatively short motifs (recall that development of this tool has been ongoing for around 15 years). As we have seen previously (see Figure 4.1), the average percentage of correct positions for width-6 motifs with 20 input sequences is slightly over 60%, as performance tails off noticably for shorter motifs. This figure also assumes that every sequence in the input dataset contains an occurrence of the motif, which is not the case here; as noted above, only 7 sequences (35%) contain the motif and again we have seen previously that we can expect around 65% correct positions when this is the case with 20 input sequences (see Figure 4.4). Given these results using synthetic data, the accuracy of JMeme in this test is as good (66% correct positions in the test on the LAN dataset) or better than could be expected (83% correct positions in the test on the LAN7 dataset).

4.1.3 Real-world Characterised Data: Iron Metabolism

Tests were carried out to identify and confirm the work carried out by Rodionov, *et al.* (2006) on the iron regulation metabolism in *Magnetospirillum magneticum* as described in Section 3.6.3. As mentioned above, Rodionov, *et al.* (2006) published a

width-19 regulatory motif (Supplementary Figure 1[1]) (5'-AATGCGAGTCA_TCGCATT-3'). The study by Lan (2008) aimed to confirm this result and identified a width-21 motif (5'-AATTGCGAGTAACTCGCAATA-3') which was very similar to the previously published motif (although not *identical*, as stated by Lan).

The results gained for widths 21, 22 and 23 are presented in Figure 4.11. It can be seen that, like the longer CtrA motif, the overall signal is not particularly strong, however the fact that small fragments within the motif are conserved over a number of different tested widths can lead us to conclude that these sections are significant, even if we are not sure of some of the other positions. Figure 4.12 shows how the results of Rodionov, *et al.* (2006), Lan (2008) and the three motif results from Figure 4.11 can be aligned.

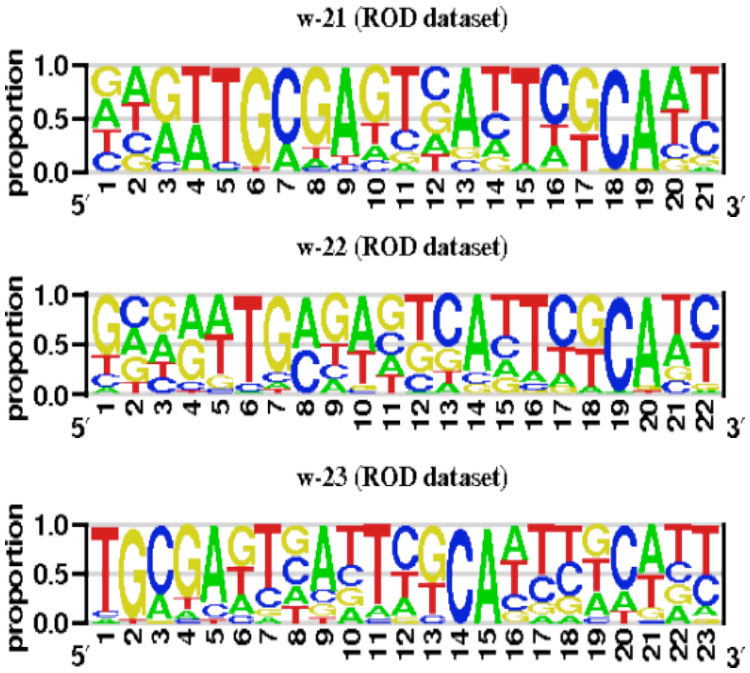

Figure 4.11: Result of the width-21, -22 and -23 tests on the ROD dataset

```
(1)         A A T G C G A G T C A _ T C G C A T T
(2)       A A T T G C G A G T A A C T C G C A A T A
(3)     G A G T T G C G A G T C A T T C G C A A T
(4)   G C G A A T G A G A G T C A T T C G C A T
(5)             T G C G A G T G A T T C G C A A T G C A T T
===
pos  1   3   5   7   9   11  13  15  17  19  21  23  25  27
```

Figure 4.12: The iron regulatory motifs identified by Rodionov, *et al.* (2006) (1), Lan (2008) (2) and Figure 4.11 (3, 4, 5) are aligned.

[1] http://www.ploscompbiol.org/article/fetchSingleRepresentation.action?uri=info:doi/10.1371/journal.pcbi.0020163.sg001 (Accessed 10th August 2009)

On examining Figure 4.12, it can be seen that the result for the width-21 run of JMeme closely resembles the previously proposed motifs (the width-21 result returned by JMeme matched 83% of the positions proposed by Lan (2008) and 79% of the positions proposed by Rodionov, *et al.* (2006)) and that there is a 17bp motif section (positions 6 to 22 above) that is very well conserved over all of the proposed motifs. This suggests that we can be reasonably certain about the strength of this part of the motif, although it is possible that there are extra positions on either side. It can be seen that using only the results from this study, position 6 is uncertain (A or likely C); however, when these results are combined with previously identified potential motifs, it is very likely that this position is C. Similarly, decisions can be made about other positions by taking the letter most likely at each position (given each result is equally possible).

It is therefore concluded that 5'-TGCGAGTCATTCGCAAT-3' is a potential iron regulatory motif in *M. magneticum*, taking all the results above into account.

It is easily conceivable that there are a number of different ways in which this result may be interpreted. The original Rodionov, *et al.* (2006) proposed motif is almost exactly palindromic about position 13 in Figure 4.12: 5'-AATGCGAGT-C-A_TCGCATT-3'. Lan (2008) notes that the motif proposed in that study is also almost palindromic; this should not be all that surprising, given its similarity to the previously proposed motif. The motif proposed in this study, above, is also quasi-palindromic around the same position: 5'-TGCGAGT-C-ATTCGCAAT-3'. The sequences TGCGA and TCGCA in particular are perhaps noteworthy. However, it has been proposed that iron regulatory motifs in *Escherichia coli* can be interpreted in a number of different ways (Escolar, *et al.*, 1999), so it is quite possible that some other structural pattern can be discerned from the proposed motif. Again, it must be stressed that the above proposed motif is just that and as such may not be completely accurate.

4.1.4 Real-world Uncharacterised Data

Tests were carried out on the three datasets of uncharacterised real-world data, as described in Section 3.8, with the aim of finding other well-conserved sequences which may be significant in the iron regulation mechanism. The results for each dataset and proposed potential sequences are presented here.

RAST

The results of the first passes for width-9, -10 and -11 are shown in Figure 4.13; Figure 4.14 shows how these results can be aligned.

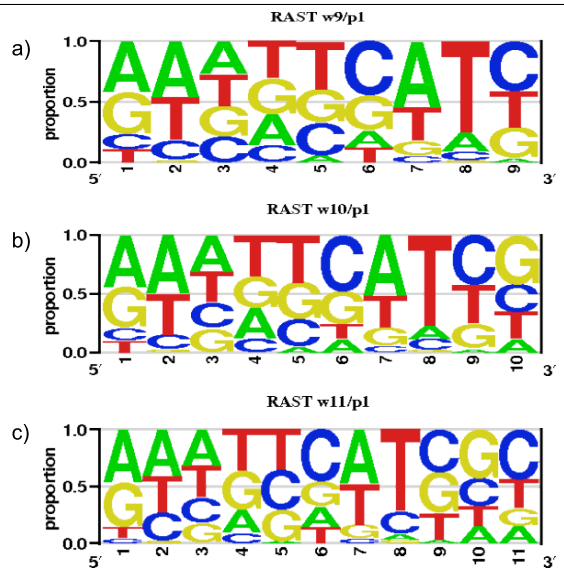

Figure 4.13: The results for the first pass for a) width-9, b) width-10 and c) width-11

```
(1)    A A A T T C A T C
(2)    A A A T T C A T C G
(3)    A A A T T C A T C G C
===
pos    1   3   5   7   9   11
```

Figure 4.14: The results for width-9, -10 and -11 tests are aligned.

On examining Figure 4.14, it can be seen that there is a highly conserved 9bp section (positions 1 to 9 above), which seems to suggest that this section is significant in some way. It is also possible that this section extends to the right if we consider positions 10 and 11; tests were conducted increasing the width further still but the results gained showed no consensus with the sequences above so we cannot be as sure about these positions.

The results of the second passes for widths between 6 and 10bp are shown in Figure 4.15; Figure 4.16 shows how these results can be aligned.

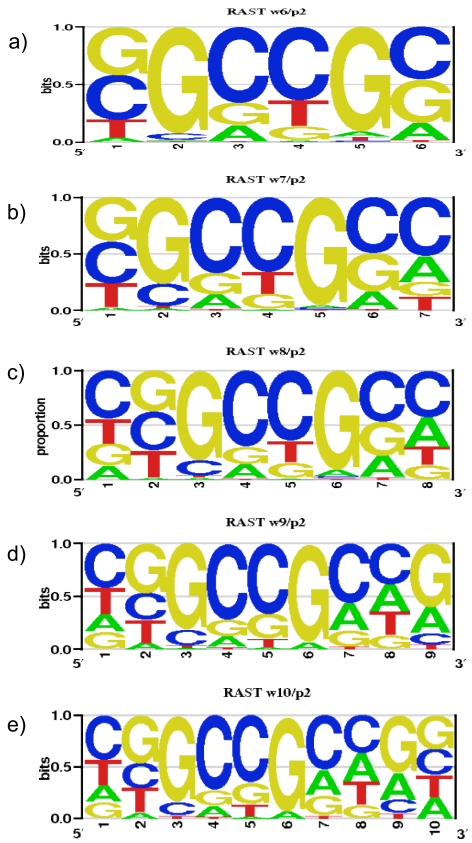

Figure 4.15: The results of the second pass for a) width-6, b) width-7, c) width-8, d) width-9 and e) width-10

```
(1)       G G C C G C
(2)       G G C C G C C
(3)     C G G C C G C C
(4)     C G G C C G C C G
(5)     C G G C C G C C G G
===
pos     1   3   5   7   9
```

Figure 4.16: The results for width-6, -7, -8, -9 and -10 tests are aligned.

On examining Figure 4.16, it is clear that again there is a highly conserved section of minimum 8bp (positions 1 to 8 above), which seems to suggest that this section may be significant. The above sequences were not optimal in log likelihood, but were discovered by JMemePlus as being optimal in information content. Again, further tests were conducted, increasing the tested width, but after width-10 the consensus with the above sequences decreased.

From the RAST dataset, we can conclude that the sequences 5'-AAATTCATC-3' and 5'-CGGCCGCC-3' are potential motif sequences. As noted earlier, an initial scan of

the genes contained in the dataset seemed to indicate that any motifs found would be motif sequences for regulator genes. A preliminary study of the sequence 5'-AAATTCATC-3' with RegulonDB[2] found it to be included in the Fur regulator of *Escherichia coli* so this sequence may provide a promising area for future research.

IRON

The results of the first pass on the IRON dataset for width-10, -11, -12 and -13 are shown in Figure 4.17; Figure 4.18 shows how these results can be aligned.

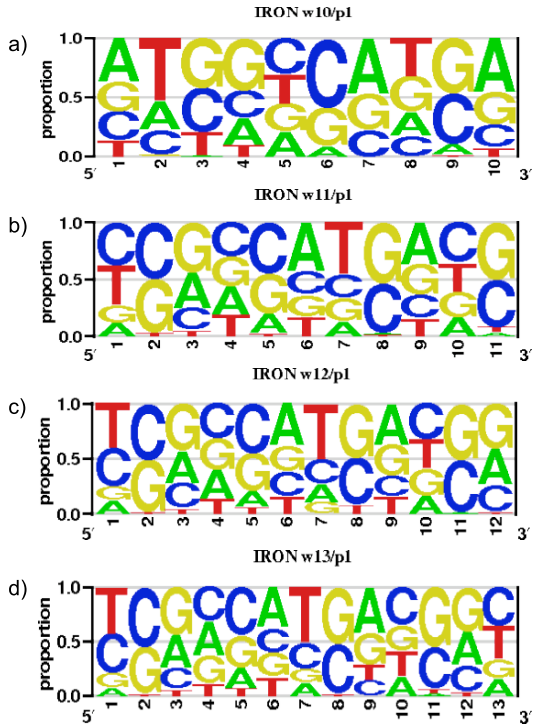

Figure 4.17: The results of the first pass for a) width-10, b) width-11, c) width-12 and d) width-13

```
(1)     A T G G C C A T G A
(2)       C C G C C A T G A G C
(3)       T C G C C A T G A C G G
(4)       T C G C C A T G A C G G C
===
pos     1   3   5   7   9   11  13
```

Figure 4.18: The results for width-10, -11, -12 and -13 tests are aligned.

On examining Figure 4.18, it can be seen that there is a conserved section of 9bp

[2]http://regulondb.cs.purdue.edu/ (Accessed 19th August 2009)

(positions 2 to 10 above), which seems to suggest that this sequence is significant in some way. Again, it is possible that this sequence could be extended as the two further positions to the right could be predicted to be 'either C or G'; future tests may reach a stronger consensus on these positions. Second passes over the IRON dataset were not clear enough to reach a conclusion about any further conserved motif sequences.

From the IRON dataset we can conclude that the sequence 5'-TCGCCATGA-3' is a potential motif sequence; it is possible that further positions could be added after further research.

FD

The results of the first pass on the FD dataset for width-10, -11 and -12 are shown in Figure 4.19; Figure 4.20 shows how these results can be aligned.

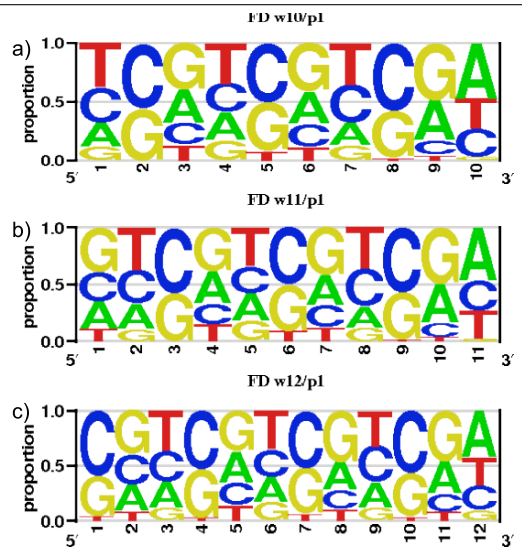

Figure 4.19: The results of the first pass for a) width-10, b) width-11 and c) width-12

```
(1)         T C G T C G T C G A
(2)       G T C G T C G T C G A
(3)     C G T C G T C G T C G A
===
pos     1   3   5   7   9   11
```

Figure 4.20: The results for width-10, -11 and -12 tests are aligned.

On examining Figure 4.20, it is clear that there is a conserved sequence of 10bp from positions 3 to 12. However, looking at Figure 4.19 seems to suggest that there is a repeating pattern of CGT and that the most significant positions are the C positions

(1, 4, 7 and 10 above); indeed, this is confirmed by further tests with longer widths. This seems to suggest that the sequence discovered is not a motif in the same sense as those discovered previously and may be a feature of genes connected in some way with ferredoxin; this is perhaps a question for biologists. Once again, the results of further passes over the FD dataset were not deemed to be significant.

From the FD dataset, we can conclude that 5'-TCGTCGTCGA-3' is a possible motif sequence, with the note of caution above.

4.2 Evaluation of Extensions

The JMemePlus application implements two extensions designed to improve the performance of JMeme, as described in Section 3.9. The effects of these extensions in the JMeme application are presented here.

4.2.1 Choosing Optimal EM Models

As explained in Section 3.9.1, altering the choice of model to be based on information content rather than log likelihood was designed in order to avoid situations where potentially more interesting models were passed over and motifs where every probability is 0.25 returned. The method of calculating the information content, taking into account the background data, is a good choice for deciding between several different motif models, as we expect that '0.25 motifs' will have very little information due to their similarity to the background data. It follows that models where more definite decisions regarding the letter at each position are taken will have a much higher information content.

As a test of the effectiveness of the implemented extension, a typical multiple motif run from the previously carried out test was chosen and retested with JMemePlus. The dataset for width-8 with 3 motifs was chosen as in the initial test, forcing JMeme to print the result from every run (rather than simply every *pass*) showed that potential motif models were being passed over for models that were less interesting, but with a higher log likelihood. The same test was run in JMeme to confirm the previous result; a percentage accuracy of 58.3% was calculated, as before. In addition, the information content per column was calculated, averaging over every column in the

three runs; this figure was calculated to be 0.8911bits/col. The test was then repeated using JMemePlus; this time, a percentage accuracy of 75.0% was calculated, due to the motif model for the third run changing from a '0.25 motif' to a more interesting motif. The results of the first two passes of JMemePlus remained the same as those for JMeme. It must be noted that the percentage accuracy for this motif alone was only 50%; that is, however, an improvement from 0%. The information content per column was again calculated, increasing to 1.0997bits/col.

It is clear that this extension allows JMemePlus to return information-optimal motifs where the log likelihood of these motifs is not optimal. Crucially, in tests, choosing motif models using the information content appears not to affect results which were optimal in both information content and log likelihood. From this, we can conclude that this extension makes a small but significant difference to the implemented algorithm, especially when multiple motifs are being searched for.

4.2.2 Incorporating Prior Beliefs

As explained in Section 3.9.2, the MEME algorithm as described by Bailey and Elkan (1994b, 1995b) contains no way of utilising prior knowledge that we may have about a motif. The concept of the energy of a motif (based on work by Werhli and Husmeier, 2007) is introduced and implemented as part of JMemePlus. It is easy to see how this can be used if we consider the synthetic data tests evaluated above. For many of the tests, the main factor in decreasing the score is the effect of shifting; for instance, if we know we are searching for a width-8 motif and JMeme only returns the last 6 positions (as the first 6 positions of the returned motif, followed by 2 noise positions). Using this belief as an input to JMemePlus allows us to give more weighting to motifs with these 6 positions as the last 6 positions, declaring the first two as 'unknown'; this should increase the chance that we should discover the whole motif with JMemePlus. The same method can be applied to real-world data, although we are unlikely to have as strong a signal or prior knowledge of the exact length of the motif.

As a test of the effectiveness of the implemented extension, a dataset containing the width-8 motif 'CTAAATGC' is used as an input to JMeme. The result produced is shown in Figure 4.21a); it is clear that we have a shifted version of the motif we are looking for. We can use the result we have as an input to JMemePlus: we input the 'belief' parameter as "xTAAATGC" (recall x represents an unknown letter). The

output of JMemePlus is shown in Figure 4.21b); we have found the first letter by incorporating our belief.

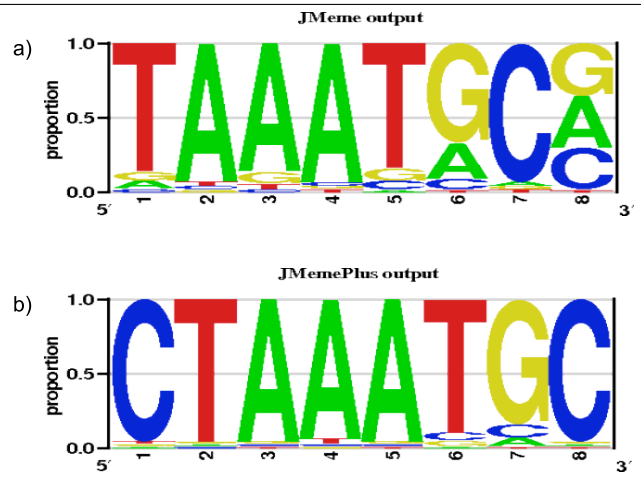

Figure 4.21: We can incorporate prior beliefs to improve the accuracy of the algorithm

In practice, things are not quite so simple and involve a slight amount of adjustment to the JMemePlus parameters. The initial output from JMemePlus is a '0.25 motif'; if we consider how our beliefs are incorporated, it is easy to see why. We are incorporating our beliefs by altering the count variables c_{jk}, which are then used to reestimate θ. This reestimation procedure requires our pseudocount parameter β. When we incorporate our beliefs, c_{jk} is multiplied by the probability $p(X|B)$, which reduces it greatly; this in turn makes β much larger than the count variables. By lowering β, the count variables are not dwarfed by β and we receive the motif we are looking for. We can find out by how much we should lower β by considering the normalising constant $C(\gamma)$, which is by far the largest factor in calculating $p(X|B)$. $C(\gamma)$ can be calculated to be approximately 4^W, therefore we should also reduce β by this amount. In this case, $W = 8$, therefore our value for β should be $0.25 \times \frac{1}{4^8} = 3.81\text{E-}6$. Running JMemePlus with this value for β does indeed produce the motif we are looking for as above.

In conclusion, we can say that, although some additional tweaking of the parameters is required, using the energy of a motif to incorporate prior beliefs can improve on the results of the JMeme algorithm. Ideally, we would like to make the incorporation of prior beliefs as simple as possible; some thoughts on how this could be done are presented in Chapter 5.

4.3 Discussion

The results presented in this section have shown the JMeme and JMemePlus applications in action. The results for the tests carried out with synthetic data indicate that JMeme performs reasonably well compared to the MEME algorithm described by Bailey and Elkan (1994b, 1995b). Although there does not seem to be any hard data presented, results of tests performed in this study seem to confirm that the basic MEME algorithm is reasonably robust to noise sequences, although obviously the fewer noise sequences in the dataset, the better the performance of JMeme will be. JMeme has also been shown to handle gapped motifs reasonably well, an important factor for searching for regulatory sequences, which can often contain gap positions.

In general, tests with data for the CtrA metabolism seem to confirm the results gained by Lan (2008). A gapped motif which is a variation on the gapped motif proposed by Lan (2008) has been proposed. The shorter ungapped motif proposed by Lan (2008) was not completely confirmed; however, 5 out of the 6 positions proposed were confirmed. Tests with the online MEME suite confirm the result proposed by Lan (2008) so it is perhaps strange that JMeme could not completely confirm this result. One potential stumbling point appears to be the length of the motif; synthetic data tests show that width-6 motifs are at the bottom limit of what can be discovered by JMeme and the result gained in the LAN test is actually better than could be expected.

Tests with previously characterised data for the iron metabolism confirmed the results proposed by Lan (2008) and Rodionov, *et al.* (2006). A 17bp motif sequence was proposed, with slight differences from the two previously proposed motif sequences; these differences are backed up by conserved sequences over a number of runs with different motif widths.

Three uncharacterised datasets were tested, with the aim of discovering potential regulatory sequences. A number of highly conserved sequences were found, which indicates that they may be significant. However, the only way to confirm which, if any, of these sequences are involved in regulatory mechanisms is to conduct wet-laboratory tests on the bacteria. Although wet-lab tests are extremely labour-intensive and despite a growing number of researchers using computational methods for motif discovery, they remain the best way to confirm a particular sequence and the majority of confirmed regulatory sequences have been found using these methods. Briefly, wet-lab tests involve creating conditions where a regulator is predicted to be active and then

'knocking out' this regulator to see if any changes take place. If, for example, the appropriate gene switches off, we know that we have found the correct regulator.

We have seen that extensions to the basic MEME algorithm can improve the results that are returned; there are, however, a number of issues with these extensions, particularly with incorporating prior beliefs. At present it is not possible to incorporate a number of different prior beliefs, so use of JMemePlus is limited to one pass of the dataset. One way around this problem is to run JMemePlus for a number of passes with no prior belief and then run JMemePlus for one pass multiple times, changing the prior belief each time. We have also seen that introducing prior beliefs requires extra parameters to be modified, which is not ideal; with further research it may be possible to solve this problem, however.

One major question which appears to remain unresolved is the lack of any promoter sequences in the results returned by JMeme. We would expect to find a number of these as promoter sequences facilitate the transcription of a gene and are generally found between 10 and 35 bases upstream of the start codon of the gene. However, no promoter sequences were uncovered during the conducted tests. There are a number of possible reasons for this; perhaps most likely is that the promoter sequences are too short and not well enough conserved in the upstream sequences to appear as results from JMeme. Generally, promoter sequences consist of two parts, each 6 bases long, with a large gap between them. As we have seen previously, this length lies at the lower limit of what can be discovered using JMeme so even with complete conservation these sequences may be harder to find. In addition, it appears that these promoter sequences are not necessarily well-conserved, which may hinder their discovery further. Another reason why they may remain undiscovered is the large gap between the two sequences; it may be that this gap is too large for JMeme to make sense of. The tests conducted with synthetic gapped sequences kept the gap fairly short so further testing of gapped motifs with JMeme may confirm if this is the reason why it is so hard to find these promoter sequences. One final possibility is that there are simply too many of these sequences and different assumptions in our model are required in order to find these sequences.

Chapter 5

Conclusion

This chapter presents a number of conclusions and observations drawn from the research project. A number of unsolved problems and suggestions for further work are also presented.

5.1 Project Summary

This project set out to investigate a number of different algorithms, implementing the most promising in Java. This was carried out, resulting in the JMeme application. While this implementation does not yet perform quite to the standards of the most current version of MEME, this version is designed to be more easily extensible. Using the created implementation, a number of previously proposed regulatory sequences were confirmed and potentially improved. A number of new potential regulatory sequences were also proposed through analysis of three new datasets.

The project also set out to test the hypothesis that extensions to the implemented motif discovery algorithm could improve motif discovery. Two extensions to the basic MEME algorithm were designed and implemented as JMemePlus. Evaluation of these extensions shows that the tested hypothesis is correct; that is, extensions to the basic MEME algorithm have been shown to improve its performance in discovering regulatory sequences in bacteria with respect to the basic MEME algorithm.

5.2 Suggestions for Further Work

There are clearly many avenues still to be explored in this area. A number of these possibilities are described here, although many more possibilities remain.

5.2.1 Other Approaches to Dataset Construction

There are other possible approaches to dataset construction which might provide interesting results. One possibility is using the Gene Ontology (GO; a bioinformatics initiative to unify the representation of gene and gene products across all species) to search for genes in the AMB-1 genome which are thought to be linked in some way to magnetosome production. One drawback is that currently the GO does not contain AMB-1 annotations, so a search must be performed for a specific term (e.g. 'iron ion binding'[1]), then using the results of that search to search through a separate annotated AMB-1 genome[2] to find potential genes. It is also possible that there may be very specific subterms of searched terms indicating magnetosome production; without some knowledge of the structure of the GO, it is not easy to uncover these terms. It is clear that the GO has some potential for the creation of test datasets, however, due to the time constraints of the current study, this avenue could not be explored

5.2.2 Further Investigation of the Regulatory Network

It is clear from even the basic description of regulatory networks provided in Section 1.3 that the true mechanism of gene regulation in *Magnetospirillum magneticum* is much more complicated than we have assumed. Most important amongst the features described is the problem of operons, where a number of genes may be co-regulated by a single regulatory sequence (see Figure 1.6). If the genes we have used in this study are controlled by operons, it is clear that we may not have extracted the gene sequences that contain regulatory sequences and instead, introduced noise sequences into our dataset. Operon prediction is a very difficult task but could be carried out

[1] http://amigo.geneontology.org/cgi-bin/amigo/term-assoc.cgi?term=GO:0005506 (Accessed 21st July 2009)

[2] ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/proteomes/22510.M_magneticum.goa (Accessed 21st July 2009)

crudely based on the distances between genes[3]; this could be carried out to check if any of the genes we are interested in are controlled in this way.

5.2.3 Further Investigation of Genes

As we have seen, at least one potential sequence has been proposed as being important in iron regulation in *M. magneticum* (Rodionov, *et al.*, 2006). This sequence and perhaps the sequences in this project (and variations of these sequences) could be used to search the AMB-1 genome for other genes which are important in iron regulation. This could be performed relatively easily using the online BLAST suite. Similarly, there are another four strains of magnetotactic bacteria (*Magnetococcus* MC-1, *Magnetospirillum gryphiswaldense* MSR-1, *Magnetospirillum magnetotacticum* MS-1 and *Marine vibrio* MV-1) which could be studied as there is likely to be a high degree of similarity between these strains and AMB-1. The genomes of these species have been fully sequenced; study of these species would likely take less time now that an application for doing so has been implemented.

While we have been concerned with genes important in iron regulation, further studies could be carried out investigating regulatory sequences directly important in production of magnetosome particles. A list of these genes is currently under construction[4] and may provide some interesting results when analysed in a similar fashion to the datasets examined in this study.

5.2.4 Improvements to the JMemePlus Application

While we have implemented two extensions to the basic JMeme algorithm implemented in this study, there are a number of possible improvements which could be made. A number are outlined here, although there are many more possibilities.

One improvement which would make the application more useful from a biological perspective would be to implement some way of aligning the result motif with the input sequences. This would allow a researcher to easily see where in the input dataset the motif (or variations of the motif) can be found. This feature could perhaps be

[3] Coulson, A., in correspondence (2009)
[4] Ward, B., in conversation (2009)

implemented using the BLAST API. Similarly, a BLAST search could be performed to search for homologous genes given a result motif.

We have seen that using a combination of the basic MEME algorithm and our method for incorporating prior beliefs that we can minimise the effects of shifting for single motifs. However, as noted previously, the shifting phenomenon becomes very important when multiple passes of MEME are carried out, as we could end up erasing shifted versions of motifs, which complicates second and subsequent passes of MEME. Clearly, we would like some way of minimising shifting automatically. Lawrence, *et al.* (1993) propose a method by which shifted versions of the current motif model are tested every n iterations, with the motif model being changed if the likelihood of a shifted version is higher than that of the current motif model. It is possible that a similar procedure could be incorporated into the JMeme application fairly easily.

As noted, there are a number of improvements which could be made when using prior beliefs. At present, there is no simple way to use multiple prior beliefs with JMemePlus. In theory, however, it should be possible to state a set of multiple prior beliefs and use all of these over a single pass of MEME, choosing the motif/belief which match with the highest information, before probabilistically erasing that motif and removing the appropriate belief from the belief set. The other present issue with using beliefs is manually having to choose a value for β; as we have seen this must be altered depending on the width parameter. Future versions could incorporate a heuristic where a value for β is chosen depending on the value of the width parameter; again, this could be acheived fairly easily. Future versions of the application could also incorporate a different heuristic for calculating our degree of confidence in our belief (the γ parameter). At present, this is estimated by maximum likelihood from the dataset but there may be better ways of calculating this value. As we have seen, we do not use our belief directly in calculating the log likelihood of a particular motif model due to difficulties decomposing the equation for log likelihood. In future studies, it may be possible to include our belief as a parameter to the model; this would have the advantage that the β parameter could remain relatively constant as we are not altering our count variables.

In retrospect, it may seem strange that we use the EM algorithm to iteratively maximise the log likelihood of the motif model, yet get the best results when we disregard the log likelihood measure when choosing the best starting parameters (recall these are chosen using the information content in JMemePlus). Perhaps one area to explore would be a reformulation of the EM algorithm, maximising the current information

content, rather than the current expected log likelihood; the form this reformulation would take is currently unknown.

5.3 Concluding Remarks

This project has successfully completed all of the aims that we set out to achieve and has confirmed the tested hypothesis. Although the scope of this project has been limited by the time constraints, it is hoped that a useful contribution to the area has been made.

Appendix A

Dataset Contents

BLAST

amb0024 TPR repeat protein
amb0070 Predicted Fe-S oxidoreductase
amb0077 Nucleoside-diphosphate-sugar epimerase
amb0080 Sialic acid synthase
amb0092 Sialic acid synthase
amb0099 Predicted nucleoside-diphosphate sugar epimerase
amb0153 Molybdenum cofactor biosynthesis enzyme
amb0189 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb0220 Methyl-accepting chemotaxis protein
amb0263 Methyl-accepting chemotaxis protein
amb0274 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb0283 Membrane protein related to metalloendopeptidase
amb0304 Membrane-bound metallopeptidase
amb0324 CheY-like receiver
amb0441 Signal transduction histidine kinase
amb0549 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb0554 Methyl-accepting chemotaxis protein
amb0586 Sensory rhodopsin II transducer
amb0614 Flagellar basal body protein
amb0629 Response regulator
amb0650 Predicted periplasmic ligand-binding sensor domain
amb0684 Flagellin and related hook-associated protein
amb0688 TPR repeat
amb0708 Predicted O-linked N-acetylglucosamine transferase
amb0713 Sialic acid synthase
amb0716 Predicted nucleoside-diphosphate sugar epimerase
amb0759 GGDEF domain
amb0848 Response regulator
amb0854 Methyl-accepting chemotaxis protein
amb0902 Membrane protein

amb0994 Methyl-accepting chemotaxis protein
amb0998 Uncharacterized low-complexity protein
amb1015 Cell division GTPase
amb1110 Signal transduction histidine kinase
amb1259 Cytochrome C2
amb1305 ATPase involved in chromosome partitioning
amb1335 Response regulator
amb1345 Response regulator
amb1370 Response regulator
amb1432 Signal transduction histidine kinase
amb1868 Methyl-accepting chemotaxis protein
amb1913 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb1956 Signal transduction histidine kinase
amb1959 Methyl-accepting chemotaxis protein
amb1984 Methyl-accepting chemotaxis protein
amb2067 SOS-response transcriptional repressors
amb2104 Methyl-accepting chemotaxis protein
amb2136 TPR repeat
amb2156 hypothetical protein
amb2196 Methyl-accepting chemotaxis protein
amb2293 Signal transduction histidine kinase
amb2357 SPY protein
amb2498 Ribosomal protein S2
amb2517 Methyl-accepting chemotaxis protein
amb2518 Membrane protein related to metalloendopeptidase
amb2564 Uncharacterized low-complexity protein
amb2639 Signal transduction histidine kinase
amb2660 Methyl-accepting chemotaxis protein
amb2788 Methyl-accepting chemotaxis protein
amb2792 Protease subunit of ATP-dependent Clp protease
amb2795 Methyl-accepting chemotaxis protein
amb2826 Methyl-accepting chemotaxis protein
amb2841 Predicted O-linked N-acetylglucosamine transferase
amb2866 LexA repressor
amb2895 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb2930 Sensor protein barA
amb3005 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb3006 Response regulator containing a CheY-like receiver domain and a GGDEF domain
amb3041 Uncharacterized protein conserved in bacteria
amb3054 Methyl-accepting chemotaxis protein
amb3060 FOG: TPR repeat
amb3102 Methyl-accepting chemotaxis protein
amb3135 Ribosomal protein S12
amb3195 Signal transduction histidine kinase
amb3243 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb3261 FOG: GGDEF domain
amb3267 Methyl-accepting chemotaxis protein
amb3314 Methyl-accepting chemotaxis protein
amb3344 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb3384 hypothetical protein

amb3401 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb3426 Cytochrome C2, iso-2
amb3438 Sensor protein gacS
amb3451 Response regulator containing a CheY-like receiver domain and a GGDEF domain
amb3467 Methyl-accepting chemotaxis protein
amb3474 Methyl-accepting chemotaxis protein
amb3495 Flagellar basal body rod protein
amb3496 Flagellar basal body rod protein
amb3517 Bacterial cell division membrane protein
amb3539 Methyl-accepting chemotaxis protein
amb3570 FOG: CheY-like receiver
amb3602 FOG: GGDEF domain
amb3690 Methyl-accepting chemotaxis protein
amb3697 FOG: CheY-like receiver
amb3699 FOG: GGDEF domain
amb3735 Signal transduction histidine kinase
amb3793 Methyl-accepting chemotaxis protein
amb3804 Methyl-accepting chemotaxis protein
amb3819 ATPase involved in chromosome partitioning
amb3829 FlbT protein
amb3847 Bacterial cell division membrane protein
amb3854 Cell division GTPase
amb3867 Membrane protein related to metalloendopeptidase
amb3871 FOG: TPR repeat
amb3878 Sensor protein gacS
amb3897 Methyl-accepting chemotaxis protein
amb3923 Phosphate starvation-inducible protein PhoH, predicted ATPase
amb3937 DNA-directed RNA polymerase specialized sigma subunit
amb3981 Chromosome segregation ATPase
amb3988 Modification methylase CcrmI
amb4000 Response regulator containing a CheY-like receiver domain and a GGDEF domain
amb4015 Signal transduction histidine kinase
amb4344 Response regulator containing a CheY-like receiver domain and a GGDEF domain
amb4347 Peptidyl-tRNA hydrolase
amb4371 Panthothenate synthetase
amb4415 FOG: GGDEF domain
amb4454 L-lactate dehydrogenase and related alpha-hydroxy acid dehydrogenase
amb4464 Phosphate starvation-inducible protein PhoH, predicted ATPase
amb4469 S-adenosylmethionine synthetase

LAN

amb0153 Molybdenum cofactor biosynthesis enzyme
amb0614 Flagellar basal body protein
amb0629 Response regulator
amb0684 Flagellin and related hook-associated protein
amb0759 GGDEF domain
amb1015 Cell division GTPase

amb1956 Signal transduction histidine kinase
amb2564 Uncharacterized low-complexity protein
amb3005 Response regulator consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain
amb3041 Uncharacterized protein conserved in bacteria
amb3060 FOG: TPR repeat
amb3467 Methyl-accepting chemotaxis protein
amb3496 Flagellar basal body rod protein
amb3738 Flp pilus assembly protein, pilin Flp
amb3829 FlbT protein
amb3854 Cell division GTPase
amb3867 Membrane protein related to metalloendopeptidase
amb3923 Phosphate starvation-inducible protein PhoH, predicted ATPase
amb4161 MoxR-like ATPase
amb4454 L-lactate dehydrogenase and related alpha-hydroxy acid dehydrogenase

LAN7

amb0614 Flagellar basal body protein
amb0629 Response regulator
amb0759 GGDEF domain
amb1956 Signal transduction histidine kinase
amb3854 Cell division GTPase
amb3923 Phosphate starvation-inducible protein PhoH, predicted ATPase
amb4454 L-lactate dehydrogenase and related alpha-hydroxy acid dehydrogenase

ROD

amb0918 CheY-like receiver
amb0936 hypothetical protein
amb0940 Uncharacterized protein
amb0954 bacterial magnetic particle specificiron-binding protein
amb0955 hypothetical protein
amb1008 hypothetical protein
amb1022 hypothetical protein
amb1424 hypothetical protein
amb1662 Fe2+/Zn2+ uptake regulation protein
amb1681 High-affinity Fe2+/Pb2+ permease
amb1811 hypothetical protein
amb2732 Ferrous iron transport protein A
amb2978 Cu/Zn superoxide dismutase
amb3013 hypothetical protein
amb3546 hypothetical protein
amb3711 hypothetical protein
amb4411 Uncharacterized protein probably involved in high-affinity Fe2+ transport

RAST

```
amb0707 Signal transduction histidine kinase
amb0823 Signal transduction histidine kinase
amb1109 Sensor protein fixL
amb1252 Signal transduction histidine kinase
amb1264 Signal transduction histidine kinase
amb1336 Signal transduction histidine kinase
amb1411 Probable sensor kinase silS
amb1662 Fe2+/Zn2+ uptake regulation protein
amb2005 Signal transduction histidine kinase
amb2213 Signal transduction histidine kinase
amb2291 Signal transduction histidine kinase
amb2736 Signal transduction histidine kinase
amb2801 Signal transduction histidine kinase
amb2876 Signal transduction histidine kinase
amb3423 Signal transduction histidine kinase involved in nitrogen
        fixation and metabolism regulation
amb3469 Signal transduction histidine kinase regulating C4-dicarboxylate
        transport system
amb3564 Signal transduction histidine kinase
amb3626 Sensor protein fixL
amb3689 Signal transduction histidine kinase
amb3881 Signal transduction histidine kinase
amb4013 Signal transduction histidine kinase
amb4306 Fe2+/Zn2+ uptake regulation protein
amb4439 Signal transduction histidine kinase
```

IRON

```
amb0705 Predicted iron-dependent peroxidase
amb0783 Putative heme iron utilization protein
amb0889 Uncharacterized iron-regulated protein
amb0937 High-affinity Fe2+/Pb2+ permease
amb0954 bacterial magnetic particle specificiron-binding protein
amb0956 bacterial magnetic particle specific iron-binding protein
amb1009 Fe2+/Zn2+ uptake regulation protein
amb1023 Fe2+ transport system protein A
amb1024 Fe2+ transport system protein B
amb1568 Nitrogenase iron-molybdenum cofactor biosynthesis protein nifN
amb1569 Nitrogenase molybdenum-iron protein alpha and beta chains
amb1572 Nitrogenase molybdenum-iron protein alpha and beta chains
amb1573 Nitrogenase molybdenum-iron protein alpha chain
amb1662 Fe2+/Zn2+ uptake regulation protein
amb1681 High-affinity Fe2+/Pb2+ permease
amb2309 Fe2+/Zn2+ uptake regulation protein
amb2731 Fe2+ transport system protein B
amb2732 Ferrous iron transport protein A
amb3037 Protein implicated in iron transport
amb3335 Predicted ferric reductase
amb3990 ferrous transporter
```

amb4088 Ubiquinol-cytochrome C reductase iron-sulfur subunit
amb4306 Fe2+/Zn2+ uptake regulation protein
amb4411 Uncharacterized protein probably involved in high-affinity Fe2+ transport
amb4460 Fe2+/Zn2+ uptake regulation protein

FD

amb0176 Ferredoxin-NADP reductase
amb0179 Indolepyruvate ferredoxin oxidoreductase
amb0180 ferredoxin oxidoreductase and related 2-oxoacid ferredoxin oxidoreductase
amb0532 Ferredoxin subunits of nitrite reductase and ring-hydroxylating dioxygenase
amb0938 Polyferredoxin
amb1057 Ferredoxin
amb1059 Polyferredoxin
amb1060 Ferredoxin
amb1488 Tungsten-containing aldehyde ferredoxin oxidoreductase
amb1565 Ferredoxin
amb1579 Ferredoxin V
amb1677 Pyruvate with ferredoxin oxidoreductase and related 2-oxoacid with ferredoxin oxidoreductase
amb2137 Ferredoxin
amb2145 627aa long 2-oxoacid-ferredoxin oxidoreductasealpha subunit
amb2146 Pyruvate:ferredoxin oxidoreductase and related 2-oxoacid:ferredoxin oxidoreductase, beta subunit
amb2157 Indolepyruvate ferredoxin oxidoreductase with alpha and beta subunits
amb2479 Ferredoxin, 2Fe-2S (AaFd4)
amb2665 Ferredoxin
amb2688 Polyferredoxin
amb2689 Ferredoxin
amb2692 Ferredoxin
amb2922 Tungsten-containing aldehyde ferredoxin oxidoreductase
amb2982 Ferredoxin
amb3023 Ferredoxin
amb3040 Ferredoxin
amb3079 Polyferredoxin
amb3080 Ferredoxin
amb3234 Pyruvate:ferredoxin oxidoreductase and related 2-oxoacid:ferredoxin oxidoreductase, gamma subunit
amb4130 Uncharacterized conserved protein containing a ferredoxin-like domain
amb4176 Ferredoxin subunits of nitrite reductase and ring-hydroxylating dioxygenase
amb4319 Ferredoxin II
amb4366 Polyferredoxin
amb4413 Polyferredoxin
amb4559 Ferredoxin

Bibliography

Aitken, M. and Rubin, D.B. (1985). 'Estimation and Hypothesis Testing in Finite Mixture Models', in *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol.47, No.1, pp.67-75

Arakaki, A., Nakawaza, H., Nemoto, M., Mori, T. and Matsunaga, T (2008). 'Formation of magnetite by bacteria and its application', in *Journal of the Royal Society Interface*, Vol.5, pp.977-99

Aziz, R.K., Bartels, D., Best, A.A., DeJongh, M., Disz, T., Edwards, R.A., Formsma, K., Gerdes, S., Glass, E.M., Kubal, M., Meyer, F., Olsen, G.J., Olson, R., Osterman, A.L., Overbeek, R.A., McNeil, L.K., Paarmann, D., Paczian, T., Parrello, B., Pusch, G.D., Reich, C., Stevens, R., Vassieva, O., Vonstein, V., Wilke, A. and Zagnitko, O. (2008). 'The RAST Server: Rapid Annotations using Subsystems Technology', in *BMC Genomics* 9:75

Baker, M.E., Grundy, W.N. and Elkan, C.P. (1999). 'A common ancestor for a subunit in the mitochondrial proton-translocating NADH:ubiquinone oxidoreductase (complex I) and short-chain dehydrogenases/reductases', in *Cellular and Molecular Life Sciences*, Vol.55, No.3, pp.450-5

Bailey, T.L. and Elkan, C. (1994a). 'Fitting a Mixture Model by Expectation Maximization to Discover Motifs in Biopolymers', in *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pp.28-36, AAAI Press, Menlo Park, California

Bailey, T.L. and Elkan, C. (1994b). 'Fitting a Mixture Model by Expectaion Maximization to Discover Motifs in Biopolymers'. Technical Report CS94-351, Department of Computer Science, University of California, San Diego

Bailey, T.L. and Elkan, C. (1995a). 'The Value of Prior Knowledge in Discovering Motifs with MEME', in *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, pp.21-9, AAAI press, Menlo Park, California

Bailey, T.L. and Elkan, C. (1995b). 'The Value of Prior Knowledge in Discovering Motifs with MEME++'. Technical Report CS95-413, Department of Computer Science, University of California, San Diego

Bailey, T.L. and Elkan, C. (1995c). 'Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization', in *Machine Learning*, Vol.21, pp.51-80

Bailey, T.L., Williams, N., Misleh, C. and Li, W.W. (2006). 'MEME: Discovering and Analysing DNA and Protein Sequence Motifs', in *Nucleic Acids Research*, Vol. 34, pp.W369-W373

Barnes, D.J. and Kölling, M. (2006). *Objects First With Java (Third Edition)* (Harlow: Pearson Education Ltd.)

Berezikov, E., Guryev, V., Plasterk, R.H.A. and Cuppen, E. (2004). 'CONREAL: Conserved Regulatory Elements Anchored Alignment Algorithm for Identification of Transcription Factor Binding Sites by Phylogenetic Footprinting', in *Genome Research*, Vol.14, pp.170-8

Bishop, C.M. (2006). *Pattern Recognition and Machine Learning* (New York: Springer)

Bussemaker, H.J., Li, H. and Siggia, E.D. (2001). 'Regulatory Element Detection using Correlation with Expression', in *Nature Genetics*, Vol.27, pp.167-71

Campione, M., Walrath, K. and Huml, A. (2001). *The Java Tutorial, Third Edition* (Boston: Addison-Wesley)

Collado-Vides, J., Salgado, H., Morett, E., Gama-Castro, S., Jiménez-Jacinto, V., Martínez-Flores, I., Medina-Rivera, A., Muñiz-Rascado, L., Peralta-Gil, M. and Santos-Zavaleta, A. (2009). 'Bioinformatics Resources for the Study of Gene Regulation in Bacteria', in *Journal of Bacteriology*, Vol.191, No.1, pp. 23-31

Das, M.K. and Dai, H.-K. (2007). 'A Survey of DNA Motif Finding Algorithms', in *BMC Bioinformatics*, Vol. 8, Suppl. 7, Pages S21

Dekhtyar, M., Morin, A. and Sakanyan, V. (2008). 'Triad Pattern Algorithm for Predicting Strong Promoter Candidates in Bacterial Genomes', in *BMC Bioinformatics*, Vol. 9, Article Number 233

Delcher, A.L., Kasif, S., Fleischmann, R.D., Peterson, J., White, O. and Salzberg, S.L. (1999). 'Alignment of Whole Genomes', in *Nucleic Acids Research*, Vol. 27, No. 11, pp.2369-76

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977). 'Maximum Likelihood from Incomplete Data via the EM Algorithm', in *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol.39, No.1, pp.1-38

D'haeseleer, P. (2006a). 'What are DNA Sequence Motifs?', in *Nature Biotechnology*, Vol. 24, No. 4, pp.423-5

D'haeseleer, P. (2006b). 'How Does DNA Sequence Motif Discovery Work?', in *Nature Biotechnology*, Vol. 24, No. 8, pp.959-61

Escolar, L., Pérez-Martín, J. and de Lorenzo, V. (1999). 'Opening the Iron Box: Transcriptional Metalloregulation by the Fur Protein', in *Journal of Bacteriology*, Vol.181, No.11, pp.3402-8

Gelfand, M.S., Koonin, E.V. and Mironov, A.A. (1999). 'Prediction of Transcription Regulatory Sites in Archaea by a Comparitive Genomic Approach', in *Nucleic Acids Research*, Vol. 28, pp.695-705

Grainger, D.C., Aiba, H., Hurd, D., Browning, D.F. and Busby, S.J.W. (2007). 'Transcription factor distribution in *Escherichia coli*: studies with FNR protein', in *Nucleic Acids Research*, Vol.35, No.1, pp.269-78

Haugen, S.P., Ross, W. and Gourse, R.L. (2008). 'Advances in Bacterial Promoter Recognition and its Control by Factors That Do Not Bind DNA', in *Microbiology*, Vol.6, pp.507-19

Heikkinen, L., Asikainen, S. and Wong, G. (2008). 'Identification of phylogenetically conserved sequence motifs in microRNA 5' flanking sites from *C. elegans* and *C. briggsae*', in *BMC Molecular Biology*, 9:105

van Helden, J., Andre, B. and Collado-Vides, J. (1998). 'Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies', in *Journal of Molecular Biology*, 281:827-42

Hertz, G.Z. and Stormo, G.D. (1999). 'Identifying DNA and protein patterns with statistically significant alignments of multiple sequences', in *Bioinformatics*, Vol.15, pp.563-77

Hu, J., Li, B. and Kihara, D. (2005). 'Limitations and Potentials of Current Motif Discovery Algorithms' in *Nucleic Acids Research*, 33:4899-913

Hughes, J.D., Estep, P.W., Tavazoie, S. and Church, G.M. (2000). 'Computational Identification of Cis-regulatory Elements Associated with Groups of Functionally Related Genes in *Saccharomyces cerevisiae*', in *Journal of Molecular Biology*, 296: 1205-14

Lan, T. (2008). 'Predict Binding Sites for Transcriptional Regulators in Magnetospirillum sp. strain ABM-1', University of Edinburgh MSc Bioinformatics Dissertation

Larrson, E., Lindahl, P. and Mostad, P. (2007). 'HeliCis: a DNA Motif Discovery Tool for Colocalized Motif Pairs with Periodic Spacing', in *BMC Bioinformatics*, Vol. 8, Article Number 418

Laub, M.T., Chen, S.L., Shapiro, L. and McAdams, H.H. (2002). 'Genes directly controlled by CtrA, a master regulator of the *Caulobacter* cell cycle', in *PNAS*, Vol.99, No.7 pp.4632-7

Lawrence, C.E. and Reilly, A.A. (1990). 'An Expectation Maximization (EM) Algorithm for the Identification and Characterization of Common Sites in Unaligned Biopolymer Sequences', in *PROTEINS: Structure, Function and Genetics*, Vol.7, pp.41-51

Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F. and Wootton, J.C. (1993). 'Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment', in *Science*, Vol.262, pp.208-14

Leahy, S. (2006). 'Power Up With Magnetic Bacteria', [online]. http://www.wired.com/science/discoveries/news/2006/05/70882 (Accessed 15th July 2009)

Leung, H.C.M. and Chin, F.Y.L. (2006). 'Finding motifs from all sequences with and without binding sites', in *Bioinformatics*, Vol.22, No.18, pp.2217-23

Liang, S. (2003). 'cWINNOWER algorithm for finding fuzzy DNA motifs', in *IEEE Computer Society Bioinformatics Conference*, pp.260-5

Liu, D., Xiong, X., DasGupta, B. and Zhang, H. (2006). 'Motif discoveries in unaligned molecular sequences using self-organizing neural network', in *IEEE Transactions on Neural Networks*, 2006, 17:919-28

Liu, F.F.M., Tsai, J.J.P., Chen, R.M., Chen, S.N. and Shih, S.H. (2004). 'FMGA: Finding Motifs by Genetic Algorithm', in *Fourth IEEE Symposium on Bioinformatics and Bioengineering*, 2004:459

Liu, J., Xu, X. and Stormo, G.D. (2008). 'The *cis*-regulatory map of *Shewanella* genomes', in *Nucleic Acids Research*, Vol.36, pp.5376–5390

Liu, J.S., Neuwald, A.F. and Lawrence, C.E. (1995). 'Bayesian Models for Multiple Local Sequence Alignment and Gibbs Sampling Strategies', in *Journal of the American Statistical Association*, 90:1156-70

Liu, J.S., Gupta, M., Liu, X. and Lawrence, C.E. (2004). 'Statistical Models for Motif Discovery', in Gatsonis, C., Kass, R.E., Carriquiry, A., Gelman, A., Higdon, D., Pauler, D.K. and Verdinelli, I. (eds.) *Case Studies in Bayesian Statistics Volume VI*, pp.3-23, Springer-Verlag, New York

Matsunaga, T., Okamura, Y., Fukuda, Y., Wahyudi, A.T., Murase, Y. and Takeyama, H. (2005). 'Complete Genome Sequences of the Facultative Anaerobic Magnetotactic Bacterium Magnetospirillum sp. strain AMB-1', in *DNA Research*, Vol.12, pp.157-66

Matsunaga, T. and Okamura, Y. (2003). 'Genes and proteins involved in magnetic particle formation', in *Trends Microbiology*, Vol.11, pp.536-41

Mironov, A.A., Koonin, E.V., Roytberg, M.A. and Gelfand, M.S. (1999). 'Computer Analysis of Transcription Regulatory Patterns in Completely Sequenced Bacteria Genomes', in *Nucleic Acids Research*, Vol. 27, No. 14, pp.2981-9

Neuwald, A.F., Liu, J.S. and Lawrence, C.E. (1995). 'Gibbs motif sampling: Detection of bacterial outer membrane protein repeats', in *Protein Science*, 4:1618-32

van Nimwegen, E. (2007). 'Finding Regulatory Elements and Regulatory Motifs: A General Probabilistic Framework', in *BMC Bioinformatics*, 8(Suppl 6):S4

Noriyuki, N. (2001). 'Applications of Magnetic Bacteria in Biotechnology' (Abstract), in *Papers of Technical Meeting on Magnetics, IEE Japan*, Vol.MAG-01, No.242-61, pp.41-44

Pavesi, G., Mereghetti, P., Mauri, G. and Pesole, G. (2004). 'Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes', in *Nucleic Acids Research* 32,W199-203

Pevzner, P.A. and Sze, S.-H. (2000). 'Combinatorial Approaches to Finding Subtle Signals in DNA Sequences', in *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology* (ed. Altman, R. *et al.*). pp.269-78

Rodionov, D.A., Gelfand, M.S., Todd, J.D., Curson, A.R.J. and Johnston, A.W.B. (2006). 'Computational Reconstruction of Iron- and Manganese- Responsive Transcriptional Networks in α-Proteobacteria' in *PLoS Computational Biology*, Vol. 2, No. 12, e163

Roth, F.P., Hughes, J.D., Estap, P.W. and Church, G.M. (1998). 'Finding DNA Regulatory Motifs Within Unaligned Noncoding Sequences Clustered by Whole-Genome mRNA Quantitation', in *Nature Biotechnology*, 16:939-45

Saiyed, Z.M., Telang, S.D. and Ramchand, C.N. (2003). 'Application of Magnetic Techniques in the Field of Drug Discovery and Biomedicine', in *BioMagnetic Research and Technology*, 1:2

Sharan, R., Ovcharenko, I., Ben-Hur, A. and Karp, R.M. (2003). 'CREME: a framework for identifying cis-regulatory modules in human-mouse conserved segments', in *Bioinformatics*, Vol.19, Suppl.1, pp.i283-91

Simon, J., van Spanning, R.J.M. and Richardson, D.J. (2008). 'The Organisation of Proto Motive and Non-Proton Motif Redox Loops in Prokaryotic Respiratory Systems', in *Biochimica et Biophysica Acta*, 1777, pp.1480-1490

Thijs, G., Marchal, K. and Moreau, Y. (2001). 'A Gibbs sampling method to detect over-represented motifs in upstream regions of co-expressed genes', in *RECOMB*, Vol.5, pp.305-12

Tompa, M., Li, N., Bailey, T., Church, G.M., De Moor, B., Eskin, E., Favorov, A., Frith, M.C., Fu, Y., Kent, W.J., Makeev, V.J., Mironov, A.A., Nobel, W.S., Pavesi, G., Pesole, G., Regnier, M., Simonis, N., Sinha, S., Thijs, G., van Helden, J., Vandenbogaert, M., Weng, Z., Workman, C., Ye, C. and Zhu, Z. (2005). 'Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites', in *National Biotechnology*, 23:137-44

Wang, T. and Stormo, G.D. (2005). 'Identifying the Conserved Network of cis-regulatory Sites of a Eukaryotic Genome', in *Proceedings of the National Academy of Sciences 2005*, 102:17400-5

Werhli, A.V. and Husmeier, D. (2007). 'Reconstructing Gene Regulatory Networks with Bayesian Networks by Combining Expression Data with Multiple Sources of Prior Knowledge', in *Statistical Applications in Genetics and Molecular Biology*, Vol.6, Issue 1, Article 15

Wingender, E., Dietze, P., Karas, H. and Knüppel, R. (1996). 'TRANSFAC: A database on transcription factors and their DNA binding sites', in *Nucleic Acids Research*, Vol.24, pp.238-41

Workman, C.T., Yin, Y., Corcoran, D.L., Ideker, T., Stormo, G.D. and Benos, P.V. (2005). 'enoLOGOS: A versatile web tool for energy normalized sequence logos', in *Nucleic Acids Research*, Vol.33 (Web server issue):W389-92

Zhou, D. and Yang, R. (2006). 'Global Analysis of Gene Transcriptional Regulation in Prokaryotes' in *Cellular and Molecular Life Sciences*, 63, pp.2260-90

Zhou, S. (2008). 'Software for the Identification of Regulatory DNA Sequences in Magnetic Bacteria', University of Edinburgh MSc Bioinformatics Dissertation