UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Inference of Complex Motifs using Biclustering Techniques

Nuno Miguel Dias Mendes

(Licenciado)

Dissertação para Obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Orientadora Científica:	Doutora Ana Teresa Correia de Freitas
Co-Orientador Científico:	Doutor Arlindo Manuel Limede de Oliveira
Presidente do Júri:	Doutora Isabel Maria de Sá-Correia Leite de Almeida
Vogais:	Doutor Arlindo Manuel Limede de Oliveira
	Doutora Marie-France Sagot
	Doutora Ana Teresa Correia de Freitas

Novembro de 2005

UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO

Inference of Complex Motifs using Biclustering Techniques

Nuno Miguel Dias Mendes

(Licenciado)

Dissertação para Obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Orientadora Científica:	Doutora Ana Teresa Correia de Freitas
Co-Orientador Científico:	Doutor Arlindo Manuel Limede de Oliveira
Presidente do Júri:	Doutora Isabel Maria de Sá-Correia Leite de Almeida
Vogais:	Doutor Arlindo Manuel Limede de Oliveira
	Doutora Marie-France Sagot
	Doutora Ana Teresa Correia de Freitas

Novembro de 2005

O trabalho subjacente à presente dissertação foi realizado sob a orientação da

Professora Ana Teresa Correia de Freitas

Professora Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores do

Instituto Superior Técnico, da Universidade Técnica de Lisboa

e sob co-orientação do

Professor Arlindo Manuel Limede de Oliveira

Professor Catedrático do Departamento de Engenharia Informática e de Computadores do

Instituto Superior Técnico, da Universidade Técnica de Lisboa

Abstract

In this thesis we propose a method to estimate search parameters for modern combinatorial motif finders, with an emphasis on the identification of complex motifs. Currently available combinatorial algorithms have proved to be highly efficient in exhaustively enumerating motifs which fulfill certain extraction criteria. Addressing the problem of identifying complex motifs is extremely important, not only because these motifs can accurately model biological phenomena but because its extraction is highly dependent upon the appropriate selection of numerous search parameters.

Our method relies on a matrix of co-occurrences that, for each pair of small sequences of length λ, stores the number of input sequences in which the most common configuration of these small sequences occurs in. Using biclustering techniques it is possible to group elements of the matrix to form larger, possibly complex, motifs.

The proposed approach is not guaranteed to find all interesting correlations in the input sequences. However, it allows the efficient identification of unusual features referring to motifs that would otherwise require an exhaustive search in the parameter space to be extracted. This is particularly important when searching for complex motifs.

The experimental results show that this approach can effectively identify a set of important motif features that can guide the specification of search parameters for modern motif finders.

Keywords: Promoter prediction, Combinatorial algorithms, Motif extraction, Complex motifs, Biclustering techniques, Matrix of co-occurrences

Resumo

Nesta tese é proposto um método para estimar os parâmetros de pesquisa para os modernos algoritmos combinatórios de extracção de motivos, com ênfase na identificação de motivos complexos. Os algoritmos combinatórios disponíveis actualmente demonstraram ser muito eficientes na tarefa de identificar motivos que cumpram determinados critérios de extracção. A abordagem do problema de identificar motivos complexos é de extrema importância, não só porque estes motivos são capazes de modelar com exactidão os fenómenos biológicos, mas também pelo facto da sua extracção estar muito dependente da selecção adequada de numerosos parâmetros de pesquisa.

O método proposto utiliza uma matriz de co-ocorrências que, para cada par de pequenas sequências de tamanho λ, guarda o número de sequências de entrada em que a configuração mais comum destas pequenas sequências ocorre. Utilizando técnicas de *biclustering* é possível agrupar elementos desta matriz por forma a identificar motivos maiores, possivelmente complexos.

A abordagem proposta não garante a identificação de todas as correlações interessantes nas sequências de entrada. No entanto, possibilita a identificação eficiente de padrões pouco comuns que indicam a presença de motivos que de outro modo necessitariam de uma procura exaustiva no espaço de parâmetros para poderem ser extraídos. Isto é particularmente importante no contexto da procura de motivos complexos.

Os resultados experimentais mostram que esta abordagem permite identificar eficientemente um conjunto de características importantes de motivos que pode ser usado para guiar a especificação de parâmetros de pesquisa para algoritmos modernos de extracção de motivos.

Palavras chave: Predição de promotores, Algoritmos combinatórios, Extracção de motivos, Motivos complexos, Técnicas de *biclustering*, Matriz de co-ocorrências

Resumo Alargado

O genoma de um organismo pode ser visto como uma sequência de DNA definida sobre um alfabeto de quatro nucleótidos $\Sigma = \{A, T, G, C\}$. Algumas regiões desta sequência correspondem a genes e são, por isso, referidas como regiões codificantes. Cada gene codifica, em regra, uma proteína. As proteínas são polímeros de aminoácidos e estão envolvidas em praticamente todas as actividades celulares podendo ter uma função estrutural, constituíndo, por exemplo, a parede celular, ou uma função catalítica, assumindo o papel de enzimas no metabolismo da célula.

O dogma central da biologia estabelece um percurso para o fluxo de informação genética: DNA $\rightarrow$ RNA $\rightarrow$ proteína. De acordo com este princípio, o RNA é sintetizado a partir de um molde de DNA através de um processo designado de transcrição e as proteínas são sintetizadas a partir do RNA num processo designado de tradução. As moléculas de RNA são, assim, intermediárias na expressão da informação genética.

Os genes de um organismo não são todos expressos simultaneamente. A sua activação depende das necessidades metabólicas da célula e está sujeita a vários mecanismos de regulação. Um dos mais importantes mecanismos de regulação da expressão dos genes é a regulação ao nível da transcrição. Este mecanismo de regulação é mediado por proteínas designadas de factores de transcrição que reconhecem especificamente certas sequências de nucleótidos localizadas, geralmente, a montante dos genes na sequência de DNA em regiões denominadas de regiões promotoras. De entre sequências de nucleótidos reconhecidas pelos factores de transcrição podemos distinguir sequências contíguas designadas de motivos simples e sequências interrompidas por espaçamentos de nucleótidos pouco importantes para a ligação dos factores de transcrição designadas de motivos complexos. A identificação destes locais de ligação é uma tarefa fundamental na compreensão dos mecanismos de regulação da expressão génica.

Até muito recentemente, todos os algoritmos para identificar locais de ligação dos factores

de transcrição extraíam apenas motivos simples, traduzindo-se na pesquisa de sequências contíguas de nucleótidos (componentes) comuns a várias regiões promotoras, a menos de algumas substituições de nucleótidos.

Actualmente, a importância da identificação de motivos complexos é crescentemente reconhecida. Existem várias vantagens em privilegiar a pesquisa de motivos complexos. Por um lado, alguns factores de transcrição têm uma estrutura intrinsecamente complexa no sentido em que reconhecem sequências não-contíguas de nucleótidos e, nestes casos, os motivos complexos adaptam-se melhor à modelação dos locais de ligação. A ligação cooperativa de vários factores de transcrição à região promotora também parece envolver o reconhecimento de várias sequências contíguas de nucleótidos separadas por espaçamentos mais ou menos constantes. Por outro lado, a imposição de espaçamentos entre sequências facilita a tarefa de distinguir entre motivos biologicamente significativos para o processo de transcrição de motivos que estão presentes nas várias regiões promotoras mas que não são importantes neste contexto. Adicionalmente, os motivos complexos podem ser usados para modelar sequências contíguas de nucleótidos com regiões centrais pouco conservadas nas várias regiões promotoras.

Vários algoritmos actuais de pesquisa de motivos já suportam a extracção de motivos complexos muito embora com várias limitações. A maior parte considera motivos complexos constituídos por duas sequências contíguas de nucleótidos com um espaçamento fixo entre si o que limita severamente o tipo de motivos complexos passíveis de serem identificados. Propostas mais recentes eliminaram a necessidade de considerar espaçamentos fixos mas continuam a permitir apenas a extracção de motivos complexos com dois componentes. Adicionalmente, estes algoritmos tendem a ser pouco eficientes porque ou enumeram todos os motivos complexos possíveis [1] ou porque envolvem um pré-processamento das sequências de entrada [2–4]. Consequentemente, estes métodos estão limitados a considerar motivos relativamente curtos e uma pequena gama de valores possíveis para as distâncias entre cada componente.

Actualmente, tanto quanto é possível apurar, existe apenas um grupo de algoritmos que consegue eficientemente identificar motivos complexos com um número arbitrário de componentes separados por um espaçamento de tamanho variável [5, 6]. Adicionalmente, estes algoritmos incorporam a possibilidade de considerar substituições de nucleótidos nos vários componentes do motivo complexo.

Estes algoritmos têm, no entanto, uma desvantagem que diz respeito ao número de parâmet-

ros que é necessário especificar. Para efectuar uma pesquisa de motivos complexos é necessário indicar o número de componentes que se pretende extrair, o tamanho mínimo e máximo de cada componente, bem como o espaçamento mínimo e máximo entre cada componente. É necessário ainda indicar a percentagem de regiões promotoras em que se exige que o motivo ocorra para que seja reportado.

Este tese tem como principal objectivo o desenvolvimento de um método capaz de estimar os parâmetros de pesquisa para os algoritmos combinatórios deste tipo. O método proposto procura identificar correlações nas sequências de entrada que possam denunciar a presença de um motivo complexo comum a várias regiões promotoras.

O método que é apresentado faz uso de uma matriz de co-ocorrências. Cada elemento desta matriz indica o número de sequências de entrada em que a configuração mais comum entre dois pares de sequências de tamanho λ ocorre. Usando técnicas de *biclustering*, são agrupados elementos desta matriz de forma a construir motivos, eventualmente motivos complexos.

Os resultados experimentais mostram que esta abordagem permite identificar eficientemente um conjunto de características importantes de motivos que pode ser usado para guiar a especificação de parâmetros de pesquisa para algoritmos modernos de extracção de motivos.

Acknowledgments

I would like to thank my advisors, Ana Teresa Freitas and Arlindo Oliveira for supervising my work in the context of this thesis. Their scientific expertise was as instructive as it was an inspiration. Their continuous support and encouragement were essential to the fulfillment of this work.

I would also like to thank everyone at the ALGOS Group for the outstanding working environment and for all the fruitful discussions on all sorts of topics which made this last year truly enriching. I also thank the companionship of João Graça in the long nights of work at INESC. A special thanks to Pedro Monteiro, Miguel Bugalho, Luís Russo, Orlando Anunciação, Pedro Varela, Nuno Tenazinha, Sarah Silva, Liliana Salvador and Ricardo Ferreira for taking the time to proofreading the early draft of this thesis.

I would also like to thank my family and my friends for all the patience in these last few months.

Contents

List of Figures

List of Tables

List of Algorithms

Glossary

Aminoacid Molecule that contains an amino and a carboxylic acid functional group.

Chromatin Complex of DNA and proteins found in eukaryotic cells.

Cytosol Consists of the internal fluid of the cell where most of its metabolism occurs.

DNA Deoxyribonucleic acid molecule composed of two strands of nucleotides forming a double helix. It is the carrier of genetic information necessary to all cellular activities.

Enzyme Biopolymer that catalyzes chemical reactions. Most enzymes are proteins although some are made of RNA or DNA.

Eukaryotes Organisms in possession of a nuclear envelope.

Nucleus Organelle found in all eukaryotic cells which contains most of their genetic material.

Prokaryotes Organisms which lack a nuclear envelope.

Promoter see **Promoter Sequence**

Promoter sequence A sequence of nucleotides recognized by RNA polymerase required for the initiation of gene transcription. Some authors distinguish the core promoter which is in fact the one recognized by RNA polymerase and the proximal and distal promoters which are recognized by specific transcription factors. The nucleotide sequences of all these promoters are a subset of all the motifs which can be present in the regulatory region. The remaining being enhancers or silencers which are generally not considered to be promoter sequences since their influence is independent of position and orientation.

Protein Organic compound consisting of aminoacids joined by peptide bonds. It is essential to the structure and function of all living cells.

RNA Ribonucleic acid molecule. It has an important role in protein synthesis.

Index of Notation

$L(\mathcal{S})$	Set of λ-mers occurring in the set of input sequences $\mathcal{S}$
$B(I, \Lambda)$	Diagonally-punctured bicluster
$C_h(\mathcal{M})$	Cut of height h in a matrix of co-occurrences $\mathcal{M}$
$\Delta_{i,\mathcal{S}}(m_r, m_s)$	Set of all configurations of the pair of λ-mers (m_r, m_s) in the ith sequence of $\mathcal{S}$
ε	Tolerance given with respect to variation in length of the relative distance between pairs of λ-mers in a configuration
I	Set of indices of rows/columns belonging to a diagonally-punctured bicluster
Λ	Set of indices of elements in the main diagonal belonging to a diagonally-punctured bicluster
$\mathcal{M}_{\mathcal{S}}^{\varepsilon}$	Matrix of co-occurrences over the set of sequences $\mathcal{S}$ with ε tolerance

(m_r, m_s, d)	Configuration of a pair of λ-mers, (m_r, m_s), denoting the fact that they occur in the same input sequence at a relative distance of d nucleotides
$\mu_{i,\mathcal{S}}(m_r, m_s, d)$	Membership function of configuration (m_r, m_s, d) with respect to the set of all configurations of (m_r, m_s) in the ith sequence of $\mathcal{S}$
$\text{Occ}_{i,\mathcal{S}}(m)$	Set of coordinates of all occurrences of a λ-mer m in the ith sequence in $\mathcal{S}$
S_i	ith sequence in a set $\mathcal{S}$
Σ	Alphabet over which all sequences are defined, typically $\Sigma = \{A, T, G, C\}$
$\mathcal{S}$	Set of input sequences $\{S_1, \ldots, S_t\}$
$\sigma_{\mathcal{S}}(m_r, m_s, d)$	Score of a configuration (m_r, m_s, d) with respect to a set of input sequences, $\mathcal{S}$
$\sigma^{\varepsilon}_{\mathcal{S}}(m_r, m_s, d)$	ε-tolerant score of a configuration (m_r, m_s, d) with respect to a set of input sequences, $\mathcal{S}$

Chapter 1

Introduction

1.1 Context

The research effort underlying this thesis was carried out at the ALGorithms for Simulation and Optimization Group (ALGOS Group) of INESC-ID, Lisboa. This work benefits from the contributions of many of the ongoing projects in the ALGOS Group (in the area of data mining and bioinformatics) and it is part of a growing effort to embrace the field of computational biology.

This work was partially supported by the Project BIOGRID POSI/SRI/47778/2002.

1.2 Aims

In recent years, especially after the completion of genome sequencing projects for various organisms, there has been a growing interest in the study of regulation and gene expression mechanisms. The amount of data now available not only concerning genome sequences but also gene expression profiles makes it unfeasible to pursue a manual analysis, and calls for some sort of automatic processing. The study of biological systems requires computational approaches not only in the analysis of biological data but also in guiding laboratory research. In this context, bioinformatics tools have become more and more central to the activity of biologists.

Despite the remarkable success of these tools in some areas of application like gene finding, sequence alignment, etc, there are still problems for which no significant results have been

achieved. Notably, the identification of biologically meaningful nucleotide sequences in cis-regulatory regions remains an open problem.

The identification and characterization of regulatory regions is a fundamental task since the conditions that determine the activation and transcription of genes depend on nucleotide sequences found therein, referred to as motifs. Many approaches have been proposed and one can find a panoply of published papers describing novel algorithms to address the problem.

Currently available methods can roughly be classified in two main classes: probabilistic and combinatorial. Other approaches have also been tried including methods using neural networks, genetic programming, etc, but with unclear results.

Probabilistic methods have the advantage of requiring few search parameters but rely on probabilistic models of the regulatory regions which can be very sensitive with respect to small changes in the input data. Some of these methods also make simplifying assumptions about the nature and abundance of the motifs to be extracted.

Combinatorial methods tend to be exhaustive but are left with two main problems: identifying biologically relevant results in the output and determining the appropriate extraction parameters. For these methods, the problem of determining what portion of the output corresponds to a biologically significant result has been addressed mostly through the use of statistical techniques and biological reasoning and it is a challenge in its own right. The problem of determining the appropriate extraction parameters is one of the central goals of this thesis and can only be understood if we examine the way current algorithms operate.

A key feature of modern motif finders is the ability to extract complex motifs, i.e., non-contiguous nucleotide sequences. The advantages of considering complex motifs are twofold. On the one hand they are good representations of some instances of the underlying biological phenomena and on the other hand they are easier to extract since the distance between contiguous components can be a restriction that filters spurious output.

Currently there is, to the best of our knowledge, only one group of algorithms that allow the extraction of complex motifs with an arbitrary number of components (SMILE/RISO [5,6]). The SMILE/RISO algorithms are combinatorial approaches that prove to be effective and efficient when the appropriate extraction parameters are reasonably bound.

The main goal of this thesis is to devise an efficient method to adjust extraction parameters for modern motif finders, particularly SMILE/RISO, using biclustering techniques.

Furthermore, this method has been validated both with synthetic and real biological data.

1.3 Claim of contributions

In this thesis we propose a method to adjust extraction parameters for modern motif finders, with an emphasis on the extraction of complex motifs. This method relies on a biclustering algorithm that operates on a matrix of co-occurrences of small sequences. The performance of this method is independent of the composite structure of the motifs being sought, making few assumptions about their characteristics.

1.4 Layout of the thesis

In the next chapter we introduce the essential concepts required to understand the underlying biological problem.

In **chapter 3** we consider the computational problem of extracting motifs and discuss the currently available methods.

In **chapter 4** we present our proposal to address the problem of parameter specification and introduce the concept of diagonally-punctured bicluster. An algorithm inspired by biclustering techniques is described alongside the presentation of an algorithm to generate a matrix of co-occurrences.

In **chapter 5** we present the experimental results of our method with both synthetic and real data.

In **chapter 6** we discuss the approach we have taken and the results that have been obtained while simultaneously presenting a roadmap for future research.

In order to facilitate the reading of this thesis we also present a glossary with key terms of molecular biology and an index of notation that we have introduced to formalize our approach.

1.5 Conventions

In this thesis we will use the common conventions adopted in the computer science community, with one notable exception. Many authors coming from the realm of computer science and mathematics and who have subsequently embraced the study of life sciences (and those who

did the opposite migration) giving birth to the multidisciplinary field of computational biology sometimes struggle with matters of terminology. In particular, the terms *sequence* and *sub-sequence* are many times used in bioinformatics to refer to the concepts of *string* and *sub-string* which are well known in the field of computer science. We will adopt the terminology used in the computational biology community and we shall always refer to sequences and sub-sequences when we mean string and sub-string. Furthermore, we will use Σ to denote the alphabet over which all the sequences are defined. Throughout this thesis we will always assume that $\Sigma = \{A, T, G, C\}$ but all assertions will be made to an unspecified Σ alphabet, unless otherwise indicated.

Chapter 2

Fundamentals of molecular biology

In this chapter we present the fundamental concepts required to understand the biological problem that motivates the computational methods discussed in this thesis.

2.1 Structure of nucleic acids

The field of molecular biology greatly benefited from the discovery of the three-dimensional structure of DNA by Watson and Crick in 1953 [7]. The DNA molecule, present in all living cells, is the carrier of genetic information which is necessary to control all cellular activities. This information is passed down to each new generation almost flawlessly. DNA is composed of two strands of nucleotides forming a double helix (Fig. 2.1). A nucleotide is a molecule formed by a pentose (deoxyribose in DNA), a phosphate group and a nitrogenous base. There are four such nucleotides found in DNA, differing only on their nitrogenous base: Adenine (A), Guanine (G), Cytosine (C) and Thymine (T).

The pentose sugar-phosphate links form the backbone of the DNA molecule and are located in the exterior of the double helix. The two strands of DNA are kept together by hydrogen bonds linking each pair of bases. In a complete helix, Adenine always pairs with Thymine and Cytosine always pairs with Guanine. Because of this, the two strands are said to be complementary (Fig. 2.2).

The information contained in DNA is represented by the specific sequence of nucleotides in either strand (the sequence of nucleotides in the complementary strand can be inferred considering the base-pairing scheme discussed earlier). It is, in fact, a digital repository of

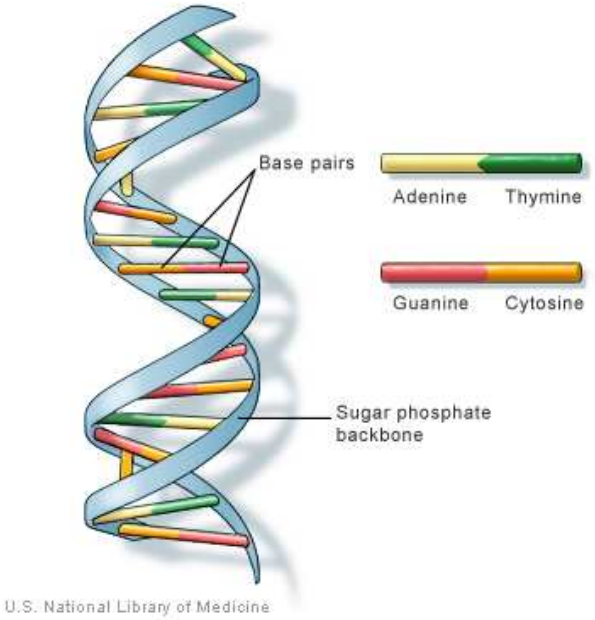

Figure 2.1: DNA double-helix structure.

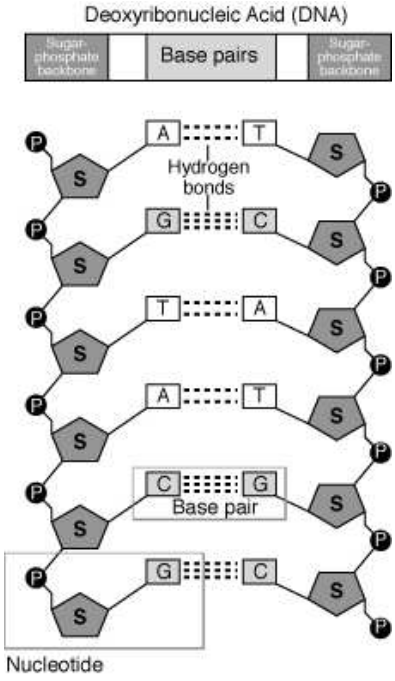

Figure 2.2: Nucleotides and the structure of DNA

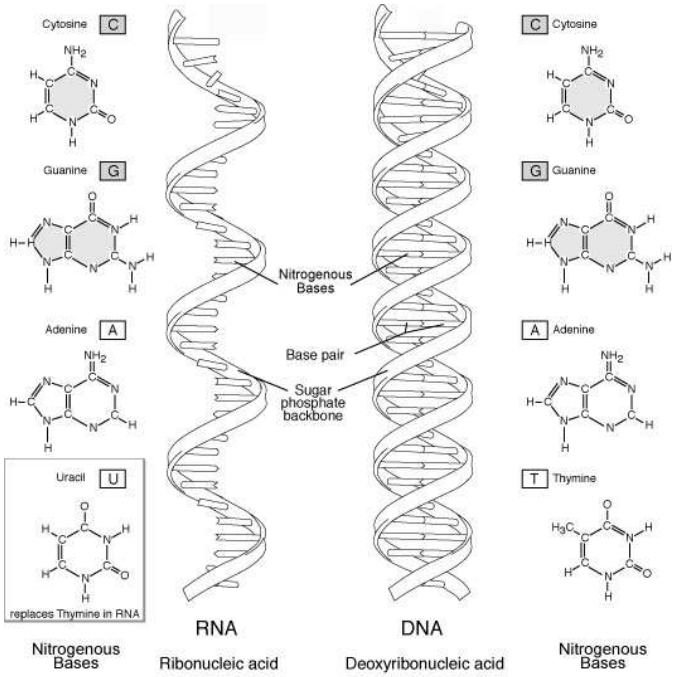

Figure 2.3: RNA versus DNA

information consisting of a text written with a four-letters alphabet.

Although DNA is structurally identical in all living cells, in prokaryotes consists of a single circular molecule whereas in eukaryotes is found associated with several proteins to form a complex named chromatin, which is located in the nucleus [8].

However, not all regions of the DNA molecule seem to carry information. Those regions which do carry information are named genes and are said to be coding regions. Genes contain the instructions necessary to direct biological activities in the cell and act by determining the structure of proteins. Genes are expressed as final products that generally consist of proteins which can serve different purposes: they can form part of the cell wall, act as catalytic components (enzymes) or influence the expression of genes and are, therefore, actors in virtually all cellular activities. The noncoding regions betweens genes are called spacer sequences. In eukaryotic cells it is common to find genes which contain large amounts of noncoding regions. In these genes, coding regions named exons are separated by noncoding regions named introns.

RNA is another nucleic acid related to DNA. There are some important differences between these two molecules. Firstly, unlike DNA, RNA is a single stranded molecule. The pentose found in RNA is ribose and not deoxyribose (thence the name of the molecules) and the

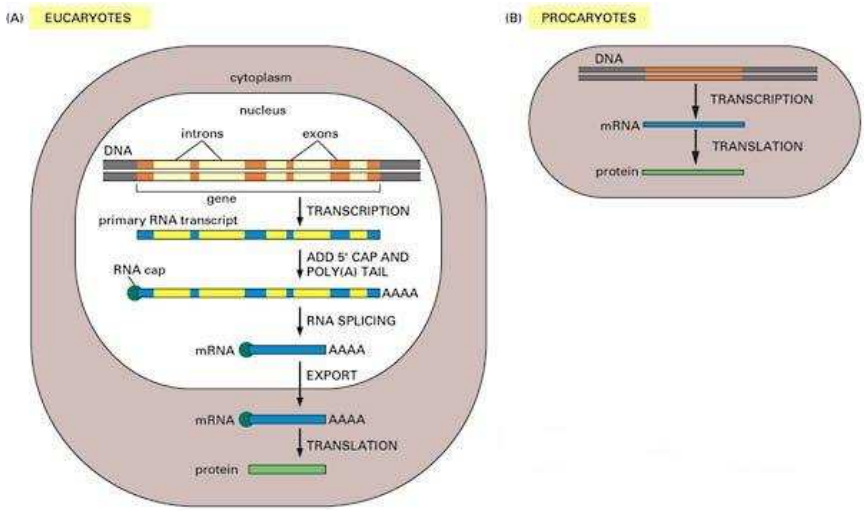

Figure 2.4: Schematic representation of the processes involved in gene expression in prokaryotes and eukaryotes

nucleotide Thymine is substituted by Uracil (U) (Fig. 2.3). Despite being a single stranded molecule, RNA sometimes presents loops where homologous portions of the molecule self-hybridize. Neither the different sugar nor the base substitution alter the base-pairing scheme found in DNA. Interestingly, in livings cells, one can find always larger quantities of RNA than of DNA. In fact, the amount of RNA varies with changing metabolic conditions whereas the amount of DNA is constant (in cells which are not in the process of cell division). This is consistent with the fact that RNA is a fundamental intermediary in the expression of genetic information as we will see below.

2.2 Gene expression

The central dogma of molecular biology [8] establishes a pathway for the flow of genetic information: DNA → RNA → protein, i.e., from the DNA repository to the final products of gene expression. The first process in which RNA molecules are synthesized from a DNA template is called transcription. The RNA molecule thus obtained is called Messenger RNA (mRNA). The subsequent process in which mRNA is used as a template for protein synthesis is called translation.

In prokaryotes, transcription and translation occur almost simultaneously whereas in eu-

		2nd base							
		U		C		A		G	
1st base	U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys
		UUC		UCC		UAC		UGC	
		UUA	Leu	UCA		UAA	(stop)	UGA	(stop)
		UUG		UCG		UAG		UGG	Trp
	C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg
		CUC		CCC		CAC		CGC	
		CUA		CCA		CAA	Gln	CGA	
		CUG		CCG		CAG		CGG	
	A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser
		AUC		ACC		AAC		AGC	
		AUA		ACA		AAA	Lys	AGA	Arg
		AUG	Met (start)	ACG		AAG		AGG	
	G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly
		GUC		GCC		GAC		GGC	
		GUA		GCA		GAA	Glu	GGA	
		GUG		GCG		GAG		GGG	

Figure 2.5: The universal genetic code

karyotes the two processes take place in different parts of the cell. In these organisms the transition from transcription to translation involves the migration of mRNA from the nucleus to the cytosol alongside with certain modifications to the mRNA molecule in a process called maturation. In Fig. 2.4 we can see a schematic representation of the different processes involved in gene expression for both prokaryotes and eukaryotes.

The typical products of gene expression, proteins, consist of sequences of aminoacids. Proteins, as we mentioned earlier, have a central role in all cellular activities and their function depends on their three-dimensional structure which, in turn, is derived from the specific linear ordering of their constituent aminoacids. The substitution of a single aminoacid in the chain can change both the structure and the function of the molecule. Since proteins are synthesized from an mRNA template it is not suprising to learn that the information about the sequence of aminoacids is represented in the sequence of nucleotides of the nucleic acid. In fact, each group of three nucleotides (codons or triplets) represents a particular aminoacid except for the so-called stop codons which signal the end of protein synthesis. There is yet another special triplet called start codon which, besides signalling the start of protein synthesis, also codes for an aminoacid, usually methionine.

The correspondence between codons and aminoacids, which is virtually identical in all living cells, is called the genetic code. Cells use 20 different aminoacids to build their proteins and there are $4^3 = 64$ different combinations of three nucleotides. In fact, several different

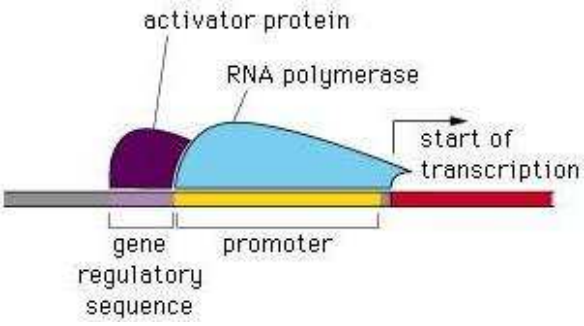

Figure 2.6: Schematic representation of regulation in prokaryotes

triplets are used to code for the same aminoacid, although no triplet is used to code for more than one aminoacid (Fig. 2.5). For this reason, the genetic code is said to be degenerate or redundant. The degeneracy of the genetic code is what accounts for the existence of silent mutations, i.e., DNA mutations that cause a codon to be changed into another which happens to code for the same aminoacid thus yielding an identical protein.

2.3 Regulation of gene expression

The genes of an organism are not all simultaneously expressed. Their activation depends on the current needs of the cell and is subjected to various regulatory mechanisms. One of the most important mechanisms is the transcriptional regulation. Some of the noncoding regions of DNA play a fundamental role in the regulation of transcription. These regions (regulatory regions) contain small sequences of nucleotides, known as motifs, which are recognized by proteins associated with the transcription machinery. The most common regulatory regions are located upstream of the start of transcription and are called promoter regions or, in a broader sense, cis-regulatory regions. The presence of these motifs is essential for the efficient binding of the cellular transcription machinery. Different motifs can play different roles in gene expression. While some are critical for eliciting the start of transcription others recruit proteins which act as activators or repressors.

RNA polymerase is responsible for the transcription process. This enzyme, when examined *in vitro*, transcribes DNA into RNA but initiates at nonspecific sites on the DNA.

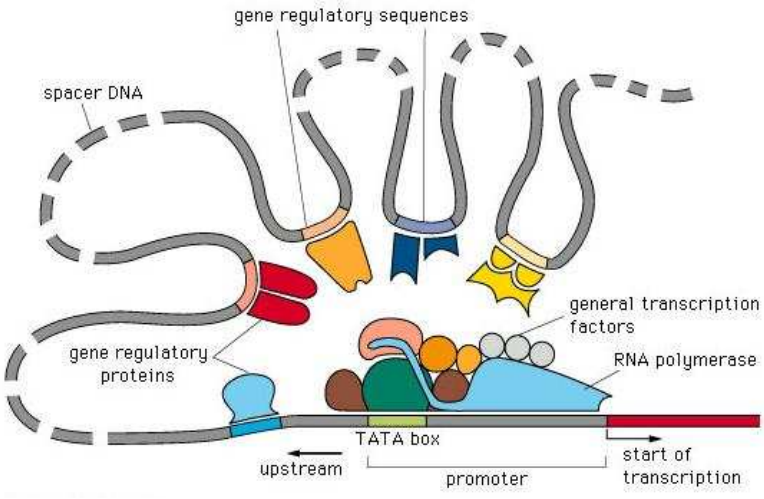

Figure 2.7: Schematic representation of regulation in eukaryotes

In bacteria[1], the RNA polymerase core is found associated to an essential subunit called Sigma (σ). This σ factor imposes a level of specificity restricting the initiation of transcription to promoter sequences. As many as seven different σ subunits have been identified [9], each of which directs the RNA polymerase to bind a unique set of promoters. The most common subunit is σ^{70} and is responsible for transcribing most genes. However, the association with a σ subunit will usually only yield a basal transcription level. The action of DNA-binding proteins called activators which also bind specific motifs allow for higher levels of expression by efficiently recruiting the RNA polymerase to specific genes (Fig. 2.6). On the other hand, the binding of another sort of proteins named repressors to specific sites can halt transcription altogether. Both activators and repressors can be referred to generically as transcription factors.

A notable exception is σ^{54} which binds to specific promoter sequences in a stable but inactive state. It requires the action of an activator to start any level of transcription. Other regulatory mechanisms in bacteria include the action of another kind of activator which induces a conformational change in the promoter to elicit the start of transcription [9].

Transcriptional regulation in eukaryotes [8] is considerably more complex although the same basic principles apply. In fact, eukaryotes use different types of RNA polymerase for different purposes. The most studied type which is also responsible for transcribing most

[1]It is worth noting that bacteria are prokaryotic organisms.

genes is RNA polymerase II. This type of RNA polymerase requires several general transcription factors to form a functional transcription initiation complex. As with bacteria, specific transcription factors modulate the activity of RNA polymerase.

The regulation of transcription in eukaryotes is primarily made at the level of initiation of transcription although in some cases it may be attenuated or stimulated at subsequent steps. Many genes in eukaryotic cells are controlled by regulatory sequences located far upstream from the transcription start site (sometimes over 10 000 nucleotides). These sequences, called enhancers, were found to stimulate transcription and are binding sites for transcription factors which are allowed to interact with the transcription machinery because the intervening DNA can form loops (Fig. 2.7). Interestingly, enhancers are active regardless of orientation with respect to the direction of transcription and can be located either upstream or downstream of the transcription start site. In addition to these regulatory mechanisms, eukaryotic cells can also regulate transcription by modifying the state of condensation of chromatin.

Genes which are co-regulated are bound to share at least a subset of motifs which correspond to binding sites of transcription factors. Similarly, genes from closely related species, performing the same biological function and purportedly having evolved from an ancestral gene (orthologous genes), are expected to have conserved regulatory sequences. Finding common sequences in the regulatory regions of these sets of genes is the basis for the operation of motif finders as we shall see in the next chapter.

Chapter 3

Related work

3.1 Motif finding

The identification of promoter sequences and binding sites for transcription factors is one of the most important tasks in the study of gene regulation. The search for the elements involved in gene expression regulation consists, essentially, in the identification of well conserved regions in noncoding DNA.

These well conserved regions are usually referred to as consensus sequences or motifs. Motif finding is the problem of discovering these motifs without any prior knowledge of their characteristics. As we said in the previous chapter these motifs can be sought by analyzing regulatory regions taken from genes of the same organism or from related genes of different organisms.

The first approach is based on the assumption that motifs common to a number of regulatory regions are likely to have a relevant role in gene expression regulation. In this approach we can largely benefit from knowledge derived from microarray experiments or from quantitative proteomics analysis which allows us to group genes that are coordinately expressed in certain experimental conditions. It is, then, reasonable to assume that some of these genes will be co-regulated, in the sense that they will share active regulatory elements.

The second approach, known as phylogenetic footprinting [10], requires careful selection of what sequences to include. These sequences must correspond to regulatory regions of genes which are evolutionarily related and that are involved in the same biological activities in different species. This approach is based on the assumption that functional regions of DNA

suffer fewer mutations than non-functional regions due to the selective pressure to preserve their biological role. Well conserved regions across these sequences are therefore expected to have a regulatory function.

Given a set of genes chosen following one of the previously described approaches, the task of identifying their regulatory regions is not always straightforward. In eukaryotes, the regulatory elements can be located quite far upstream from the start of transcription but cover only a small portion of the intergenic regions [8, 11]. As a rule of thumb, one can choose to consider stretches of up to a few thousand nucleotides upstream from the transcription start site. However, in the case of enhancers, active binding sites can be located downstream of the gene or even in introns. In prokaryotes, intergenic regions are usually much smaller and regulatory elements are located fairly near the start of transcription. However, prokaryotic genes also tend to cluster in structures called operons which share a regulatory region governing the expression of all the genes in the group [8,9]. Moreover, there are cases in which genes of an operon contain secondary promoters in addition to the common regulatory region so that even if information were available about which genes form operons (which, generally, is not) the corresponding intergenic regions could not be discarded without careful analysis.

It is also known that the transcription machinery will recognize binding sites even if the motifs do not occur exactly [12], i.e., if there are some nucleotide substitutions or even insertions and deletions. Since we cannot always confidently establish a set of co-regulated genes a computational approach to motif finding should also permit motifs not to occur in all input sequences.

An algorithm to address motif finding (i.e., a motif finder) should, therefore, tackle the problem of extracting motifs under these difficult conditions and with relatively few information. This problem is sometimes referred to as *ab initio* motif extraction. A related problem is motif localization which consists in the identification of the occurrences of a motif in a sequence given a motif description. In this thesis we are mainly concerned with motif finding. However, both problems bear the question of motif representation.

Motifs have been represented as a nucleotide sequence (consensus sequence), a profile matrix, a weight matrix, an automaton or a sequence over a degenerate alphabet [12], but most modern motif finders report extracted motifs as plain nucleotide sequences or as a weight matrix. These weight matrices, called PWM (Position Weight Matrices) or PSSM (Position

Specific Score Matrices) generally represent contiguous nucleotide sequences of a certain length l. These $|\Sigma| \times l$ matrices keep, for each position in the motif, a score for each character in the nucleotide alphabet. Recall that a PSSM is describing a set of motif occurrences so these scores should allow us to distinguish a true occurrence from a non-occurrence.

The first attempt to compute the scores for these weight matrices used a perceptron [13] and was aimed at detecting translation initiation regions in mRNA. The weights of the matrix were the same computed for the neural network, given the appropriate encoding for each motif occurrence to be described.

Later attempts computed the scores as the negative logarithms of the frequencies of each nucleotide at each position [14–16]. The sum of the scores for any particular sequence yields the negative logarithm of the probability of observing that particular sequence in the collection of described motif occurrences, assuming that the positions are independent.

In [17], a study was made concerning the information content held by several known binding sites at each position. This culminated in another way to compute the scores of the PSSM [18] where each element in the weight matrix is calculated as:

$$H(b,i) = -\ln \frac{f_{b,i}}{p_b}$$

where $b \in \Sigma$ is one of the four nucleotides, i is a position in the motif being described, $f_{b,i}$ is the frequency of the nucleotide b in position i across the set of occurrences and p_b is the frequency of nucleotide b across the entire genome of the organism being considered. It is not clear, however, what p_b should be when we try to describe occurrences of motifs taken from regions of different organisms. In [19] the authors noted the lack of a good estimate of the statistical significance of observing a specific information content and proposed a method for calculating the p-value of an information content score.

It is easy to see that in the methods discussed so far the score of a particular sequence is simply the result of the additive contribution of the scores of each nucleotide in each position.

More recent definitions compute the score of each element of the matrix as the relative frequency of each nucleotide introducing pseudo-counts to compensate for small learning sets [12,20]. In this approach, each element is computed as:

$$W(k,j) = \frac{m_j(k) + b_k}{m + b}$$

where $m_j(k)$ denotes the number of times the nucleotide $k \in \Sigma$ occurs in position j in the set of

known binding sites, b_k is the pseudo-count introduced for each nucleotide k, m is the number of known binding sites and $b = \sum_{k \in \Sigma} b_k$. In this case, the score of a particular sequence is the product of the corresponding elements.

Using weight matrices to represent a motif has the clear advantage of capturing much more information about the putative binding site than other representations. For instance, many transcription factors will recognize, at some positions, a purine (adenine or guanine), a pyrimidine (cytosine or thymine), a weak bond (thymine or adenine) or a strong bond (cytosine or guanine) regardless of the specific nucleotide present therein. This information is not so clearly represented by an equivalent collection of plain nucleotide sequences (patterns). On the other hand, plain sequences are more appropriate for motifs with few degenerate positions and facilitate the problem of determining what is a motif occurrence.

There are two major classes of motif finders: probabilistic and combinatorial. Although not all algorithms fit adequately into this classification, the most popular motif finders currently available do.

Probabilistic methods include approaches based on EM (Expectation-Maximization) [21] like PROJECTION [22] and MEME [23,24] or its stochastic analog, Gibbs sampling [25–27] used by GibbsDNA [25]. These methods use a two-phase iterative procedure where in the first step the likeliest occurrences of the motif are identified, based on a model computed in the previous iteration. The second step adjusts the model for the motif (usually a weight matrix) based on the occurrences determined in the previous step. In the first iteration the parameters of the initial model are usually set randomly.

Some probabilistic approaches assume that the motif will occur in all input sequences or require the specification of a fixed length. The most flexible algorithms in this class require only a length range to be specified. The major drawback with these algorithms is their sensitivity to noise in the data and the fact that they are not guaranteed to converge to a global maximum. Moreover, most of them assume that there will only be one motif occurring in the input sequences and at most once in each sequence. Some algorithms like MEME have removed these assumptions but are less efficient [23,24].

CONSENSUS [19] is a greedy algorithm that outputs PSSMs, saving instances with the best information content score in each step. It is, once again, not guaranteed to find optimal solutions but it can cope with zero or multiple occurrences of the motif in each input sequence.

Combinatorial methods, which typically extract motifs consisting of plain nucleotide sequences or sequences over a degenerate alphabet, usually involve enumerating all possible patterns either explicitly or implicitly. The simplicity of this approach allows us to define a clear computational problem. Consider a set of sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$. We are asked to find motifs within a range of lengths $l_{\min}, \ldots, l_{\max}$, which occur on $q \leq t$ of the presented sequences with at most e mismatches, i.e., at most e nucleotide substitutions (also referred to as having a Hamming distance up to e). It follows from this definition that a motif may or may not occur exactly on the given set of sequences, due to the allowed degeneration. For instance, in the example illustrated by Fig. 3.1, only the motif CATAT is extracted, whereas in the case of Fig. 3.2 motifs CATAA, CATAC, CATAG and CATAT satisfy the extraction parameters. The reason for requiring the motif to occur in less than t sequences is related to the fact that some input sequences may be corrupted in the sense that they may not actually contain the motif being sought. Algorithms that take this approach either enumerate all possible patterns of a fixed length l, which we will henceforth refer to as an l-mer, and verify its occurrence in the input sequences with at most e mismatches (pattern-driven approach) or take each l-mer occurring in the input sequences and generate its e-mismatch neighbourhood, i.e., all the patterns up to e mismatches away from the pattern being considered, and keep a table with a hit count, reporting all patterns above the q threshold (sample-driven approach). Several branch-and-bound algorithms have been proposed in the last few years that try to reduce the exponential search space taking advantage of sophisticated data structures. The MULTIPROFILER algorithm [28] follows a sophisticated sample-driven approach whereby it manages to avoid generating the e-mismatch neighbourhood for all sampled sequences. PatternBranching [29], on the other hand, manages to avoid analyzing all the patterns in the e-mismatch neighbourhood of a sample sequence. The WINNOWER algorithm [2] is based on graph theory. It represents each l-mer as a vertex in a graph and each pair of vertices is connected by an edge if the two l-mers have no more than $2e$ mismatches. Motifs are found by identifying cliques in the graph. MITRA [3] relies on a mismatch tree that partitions the search space. Each branch of the tree is labeled with a letter representing one of the four nucleotides and each node is associated with the l-mers in the input sequences whose prefix matches the path-label of the node with at most e mismatches. The algorithm will stop branching as soon as it determines that the subspace associated with a node is unable to hold

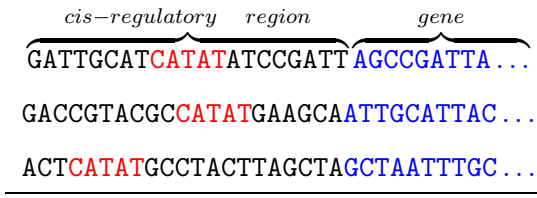

Figure 3.1: Motif extraction for $l = 5$, $e = 0$ and $q = t = 3$

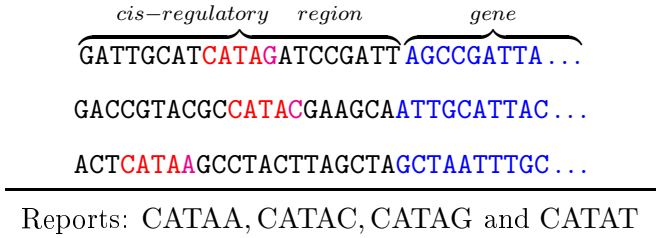

Figure 3.2: Motif extraction for $l = 5$, $e = 1$ and $q = t = 3$

a motif occurring in a least $q \leq t$ input sequences. SMILE [5] and RISO [6] use a generalized suffix-tree [30, 31] to represent the set of input sequences. They then perform an exhaustive lexicographic search to identify motifs which occur in $q \leq t$ input sequences with at most e mismatches. While traversing the suffix-tree the algorithm avoids visiting all nodes by halting the search whenever it determines that the restrictions imposed by the extraction parameters can no longer be met.

Despite the fact that these algorithms take exponential time or space in terms of l, they represent a straightforward approach to motif finding and, unlike the probabilistic methods, their output is easily interpreted.

The major problems with combinatorial motif finders are their inability to discriminate the relevant extracted motifs from the potentially numerous false positives and the large number of parameters that need to be specified (especially when searching for complex motifs as we will discuss in the next section). The large number of false positives is mostly dealt with using statistical tests to assess how unexpected is an extracted motif for a specific set of parameters, given the statistical characteristics of the input sequences [32–34]. Addressing the problem of the large parameter space is the central aim of this thesis and will be discussed in the next chapter.

3.1.1 Extraction of complex motifs

So far we have mainly discussed the extraction of motifs consisting of contiguous sequences of nucleotides, also known as simple motifs, monads or ungapped patterns. In effect, early algorithms had little or no support for the extraction of motifs with gaps. These motifs with gaps or spacers, which we will refer to as complex motifs, otherwise known as composite motifs, structured motifs or multi-ads (dyads, triads, etc.) consist of several ordered simple motifs at a certain distance of one another.

The advantages of considering complex motifs are manyfold. On the one hand, complex motifs can be better models of promoter regions. Some transcription factor DNA-binding domains have a composite structure, forming dimers (helix-loop-helix or leucine-zipper domains) and the cooperative binding of several transcription factors and RNA polymerase to the DNA molecule also seems to be bound by distance restrictions. On the other hand, many authors now agree that component motifs may be too weak to be extracted in isolation, i.e., they may be poorly distinguishable from the surrounding noise in the sequences, but by imposing a certain distance between component motifs an unusual (and thus statistically significant) pattern may be identified. This is a critical issue for algorithms that extract too many motifs and are left with the problem of deciding which of them are to be considered relevant. In addition to these advantages, complex motifs can be used to model simple motifs with highly degenerate central regions. In this case, the gap between component motifs effectively corresponds to a set of wildcards.

As we said, most motif finders have limited ability to incorporate gaps. However, in recent years several proposals have been published. Some combinatorial as well as probabilistic algorithms can now extract complex motifs although usually with no more than two components and often searching for a gap of a fixed length. These approaches are in general not very efficient since they enumerate all possible motifs with two components either explicitly [1] or by preprocessing the input sequences, as is the case with WINNOWER [2] and MITRA [3,4]. This preprocessing involves generating virtual $(l_1 + l_2)$-mers resulting from the concatenation of every l_1-mer at a certain range of distances from every other l_2-mer in the input sequences thereby reducing the problem to finding simple motifs. If the range of acceptable distances between each component is wide this method becomes very inefficient in practice, especially for large or numerous input sequences. These methods are, therefore, restricted to considering

	y_1	y_2	y_3	...	y_m
x_1	a_{11}	a_{12}	a_{13}	...	a_{1m}
x_2	a_{21}	a_{22}	a_{23}	...	a_{2m}
x_3	a_{31}	a_{32}	a_{33}	...	a_{3m}
...	...	...	...	...	...
x_n	a_{n1}	a_{n2}	a_{n3}	...	a_{nm}

Figure 3.3: Representation of a data matrix

relatively short motifs and a limited range of distances between each component.

To the best of our knowledge, there is only one family of algorithms which can efficiently search for complex motifs with an arbitrary number of components separated by a variable distance and, in addition, is able to incorporate mutations in any of the components: SMILE/RISO [5,6]. These combinatorial algorithms take advantage of a suffix-tree to deliver their unmatched flexibility. However, there is a price to pay for this flexibility which has to do with the size of the parameter space and the need to adjust the search parameters to obtain a tractable output.

3.2 Biclustering

In the next chapter we will introduce a method for parameter estimation using biclustering techniques. It is, then, important to offer some background on the problem of identifying biclusters and to discuss currently available algorithms.

Biclustering algorithms have already been extensively used to address problems in the field of computational biology, in particular, in the analysis of gene expression data [35]. Usually, gene expression data is arranged in a data matrix, where each row corresponds to a gene and each column corresponds to an instant of time or an experimental condition. Fig. 3.3 illustrates a data matrix where each row x_i can represent a different gene and each column y_i a specific condition.

In order to identify an activation pattern common to a group of genes under a subset of all the experimental conditions we have to search for a proper sub-matrix, i.e., a subset of rows and a subset of columns. Traditional clustering algorithms are not able to achieve

this, since they would only identify either a subset of genes presenting a similar behaviour across all experimental conditions or a subset of conditions where every gene behaves similarly. A new approach which came to be known as biclustering allows us to group rows and columns simultaneously reporting a subset of genes exhibiting a similar behaviour on a subset of conditions.

Given a data matrix, A, with n rows and m columns, a_{ij} is the matrix element on row i and column j. Matrix A, can be seen as a set of rows $X = \{x_1, \ldots, x_n\}$ and columns $Y = \{y_1, \ldots, y_m\}$, denoted by (X, Y). A bicluster, B, being a subset of rows $I \subseteq X$ and columns $J \subseteq Y$, can be denoted by (I, J).

The problem addressed by biclustering algorithms is the identification of a set of biclusters $B_k = (I_k, J_k)$ given a data matrix A, so that each element on a bicluster B_k satisfies some specific characteristic of homogeneity. In this thesis we are only interested in identifying a variation of constant biclusters, i.e., a bicluster, (I, J), where each of its elements has the same value, α, i.e., $a_{ij} = \alpha$ for all $i \in I, j \in J$. A bicluster that obeys the previous condition is said to be a perfect constant bicluster, but in many situations one is content with a near-constant or low-variance bicluster.

A data matrix can be seen as a representation of a weighted bipartite graph. A graph $G = (V, E)$, where V is the set of vertices and E the set of edges, is said to be bipartite if its vertices can be partitioned into two sets L and R ($V = L \cup R$), such that every edge, $(u, v) \in E$, is such that $u \in L$ and $v \in R$. A data matrix $A = (X, Y)$ can be seen as a weighted bipartite graph where each node $n_i \in L$, corresponds to a row and $n_j \in R$ corresponds to a column. The edge $(n_i, n_j) \in E$ has weight a_{ij} denoting the matrix element on row i and column j.

The problem of finding biclusters can be equated with the problem of finding a biclique in a bipartite graph. A biclique in a bipartite graph, $G = (L \cup R, E)$, is a sub-graph $G' = (L' \cup R', E')$ such that $L' \subseteq L$, $R' \subseteq R$, $E' = \{(u, v) \in E : u \in L', v \in R'\}$ in which $(u, v) \in E'$ for all $u \in L'$,$v \in R'$. Fig. 3.4 shows a bipartite graph, with each l_i and r_i vertex pertaining to the L and R partitions, respectively. The maximum size biclique, in this example, is the one formed by vertices l_1, l_2, r_1, r_2 and the edges between them.

Considering the simplest case, when our data matrix A is a binary matrix, i.e., a matrix whose elements are either 0 or 1, the corresponding bipartite graph will contain the edge

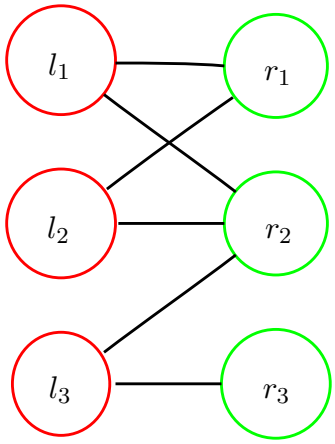

Figure 3.4: Example of a bipartite graph

(n_i, n_j) iff $a_{ij} = 1$. In this case, a constant bicluster in A with each $a_{ij} = 1$ corresponds to a biclique in the bipartite graph. Thus, identifying a maximum size bicluster in A is equivalent to finding a maximum edge biclique in a bipartite graph, which is known to be an NP-complete problem [36]. The search for more sophisticated types of biclusters that has to perform computations on the actual value of each matrix element is necessarily not less complex than this case. It is not surprising, then, that most algorithms that address this problem use heuristic approaches.

In many situations, however, we have to consider the fact that the value of an element a_{ij} in the data matrix must be seen as the result of the contribution of all the biclusters that share row i and column j. To account for this situation, some authors have introduced a plaid model [37] in which each element of the data matrix is viewed as a sum of layers. The plaid model can be defined as follows:

$$a_{ij} = \sum_{k=0}^{K} \theta_{ijk} \rho_{ik} \kappa_{jk}$$

where K is the number of layers (biclusters) sharing row i and column j of the data matrix, θ_{ijk} denotes the contribution of bicluster B_k for the value of the specified matrix element, and where ρ_{ik} and κ_{jk} are binary values representing the membership of row i and column j with respect to bicluster B_k. The value θ_{ij0} is used to model a possible bicluster including the entire data matrix contributing with a background value common to all matrix elements. Therefore, we define $\rho_{i0} = \kappa_{j0} = 1$. This has been designated as the *general additive model* [35]. If we

are considering only constant biclusters then $\theta_{ijk} = \alpha_k$, where α_k is the constant value of all elements of B_k.

Similarly, one can define a *general multiplicative model* [35], as such:

$$a_{ij} = \prod_{k=0}^{K} \theta_{ijk}\rho_{ik}\kappa_{jk}$$

Current algorithms will either try to identify a single bicluster which maximizes a given merit function (e.g. minimum variance, in the case of constant biclusters) or a given number of biclusters. To this effect, the different methods can take one of many approaches. In one approach, the algorithms will discover one bicluster at a time [37,38]. In this case, previously identified biclusters need to be masked (usually with random values) so that the algorithm does not repeatedly extract the same bicluster or they can extract each bicluster iteratively relying on a plaid model. In another approach, the methods may try to discover a set of biclusters at a time [39–41]. These methods usually rely on hierarchical clustering algorithms that iteratively generate clusters of rows and columns which are subsequently combined to extract biclusters. Finally, the algorithms can try to discover all biclusters simultaneously [42–44]. In this case, they usually rely on a set of initial biclusters called seeds which are obtained by randomly assigning rows and columns to each. The algorithms will then start an iterative process in which they try to improve the quality of the biclusters with respect to a merit function by adding or removing rows and columns. Another alternative is to try to exhaustively enumerate all biclusters [45–51], an approach we will discuss below.

Currently available biclustering algorithms have been divided into five classes:

1. Iterative row and column clustering combination
2. Divide and conquer
3. Greedy iterative search
4. Exhaustive bicluster enumeration
5. Distribution parameter identification

The first class is the most straightforward approach to biclustering and consists of algorithms which try to iteratively combine clusters of rows and clusters of columns obtained

separately [40, 41, 48]. The class named *divide and conquer* refers to algorithms which break the problem into several similar subproblems of smaller size [39, 52]. These smaller problems are then recursively solved and subsequently combined to obtain a solution to the original problem. These methods generally work by splitting the data matrix into sub-matrices according to some heuristic function. Divide-and-conquer algorithms are potentially very fast but they may miss an undefined number of biclusters whose elements occur across various sub-matrices and which are consequently split before they can be identified. Methods which perform a *greedy iterative search* proceed by always making the locally optimal choice hoping that it will lead to a good global solution [38, 42–44, 50, 51, 53]. These methods are based in the greedy addition or removal of rows/columns in order to maximize a merit function. The class of algorithms performing *exhaustive bicluster enumeration* [45–47] is based on the observation that the best biclusters with respect to a merit function can only confidently be identified using a thorough analysis of all possible biclusters. There is, however, an exponential number of possible biclusters in terms of the size of the data matrix. These methods, therefore, are forced to assume restrictions on the size of the biclusters and to use efficient techniques if they are to be of practical use. Finally, algorithms based on *distribution parameter identification* [37, 54, 55] assume that biclusters are generated according to a statistical model. The rationale is, therefore, to estimate the distribution parameters that better fit the data. This is done by relying on an iterative procedure which tries to minimize a given criterion.

In the next chapter we will discuss a method that uses biclustering techniques relying on an algorithm which can be placed within the class of algorithms performing greedy iterative search. Despite using a non-conventional definition of the problem it will still apply the basic principles discussed in this section.

Chapter 4

Inference of complex motifs

Current methods for the extraction of complex motifs have a major drawback. Their output is, in practice, extremely sensitive with respect to input parameters. If we are too permissive by allowing a high degree of degeneration or by considering a large range of allowable lengths or yet, if we require the motifs to be present only in a small fraction of the input sequences, we may get an incommensurate number of motifs as output and we are left with the problem of identifying the biologically relevant ones. On the other hand, if we specify rigid parameters, like a specific length, low degree of degeneration and require the motif to be present in all sequences we may get no output at all. In fact, without any prior information, any rigid parameter specification is purely speculative. These problems are even more pressing for complex motifs where one needs to specify the number of components, the allowed length and the number of mismatches for each, and also the distance between components. An exhaustive search in the parameter space in this case is absolutely unfeasible. In this thesis we present a method which can address these issues by avoiding some degree of parameter sensitivity, especially in what concerns complex motifs.

In [5], Sagot pointed out the need to distinguish between a motif and its occurrences in the input sequences. In fact, Sagot avoided the use of the term motif altogether introducing the notion of model. A model corresponds to a description of what constitutes a model occurrence. This distinction is particularly useful for complex motifs. A model (simple motif) is defined as a sequence over Σ^+. A model m is said to have an e-occurrence, or simply an occurrence, in the input sequences, if there is a word u in the input sequences at a Hamming distance of m not greater than e. A model is said to be valid if it has occurrences in at least $q \leq t$ input

sequences. A structured model (complex motif) is defined as a pair (m, d) where:

- $m = (m_i)_{1 \leq i \leq p}$ is a p-tuple of models ($m_i \in \Sigma^+$), denoting p components
- $d = (d_{\min_i}, d_{\max_i}, \delta_i)_{1 \leq i \leq p-1}$ is a $(p-1)$-tuple of triplets, denoting the $p-1$ gaps between components

Furthermore, considering a set of input sequences $\mathcal{S} = \{S_1, \ldots, S_t\}$, a structured model is said to be valid if, for all $1 \leq i \leq p-1$ and for all occurrences u_i of m_i, there are occurrences $u_1, \ldots, u_p$ of simple motifs $m_1, \ldots, m_p$ such that:

- $u_1, \ldots, u_p$ belong to the same input sequence
- there exists d_i, with $d_{\min_i} + \delta_i \leq d_i \leq d_{\max_i} - \delta_i$, such that the distance between the end position of u_i and the start position of u_{i+1} in the sequence is in $[d_i - \delta_i, d_i + \delta_i]$
- d_i is the same for the p-tuple of occurrences present in at least $q \leq t$ distinct input sequences

These definitions serve the purpose of a motif finder which needs to restrict the search space and to decide what is a sufficiently common pattern so that it can be reported. They offer a clear definition of what should be extracted and reported as a valid motif under specific search parameters, including the number of components, and the distance parameters.

Our purpose is to identify features in the input sequences that indicate the presence of interesting patterns (not unlike motif finders in this regard), by taking a broader view of the search space. We make no assumptions about the number of components of the complex motifs we are looking for, nor about the distances between each component. We sacrifice, however, the predictability of the results insofar as the method we propose is not guaranteed to identify the presence of all interesting complex motifs and will also be vulnerable to the possibility of reporting false positives. Furthermore, as an initial approach, we do not consider the search for motifs with degeneration.

In this context, we define a complex motif loosely as being composed of an undefined number of component simple motifs each separated by a distance that is allowed to vary within an interval of width 2ε, this means that a complex motif can be seen as a pair (m, d) where $m = (m_i)_{1 \leq i \leq p}$, with $m_i \in \Sigma^+$ and $d = (d_i)_{1 \leq i \leq p-1}$. An occurrence of a complex motif

S_1 : TAACCTGGTACA

S_2 : CGAATCTTGGTC

S_3 : GGAACTGCGGTG

S_4 : CTAATCCTAGGC

S_5 : GTAACTTCCGGT

S_6 : TCAAGCCTAGGC

Figure 4.1: A set of input sequences

is, similarly, a set of exact occurrences of each component simple motif, $u_1, \ldots, u_p$ in the same input sequence, where each occurrence is separated by a gap whose length is in $[d_i - \varepsilon, d_i + \varepsilon]$.

The only parameter we are expected to specify is ε. All the other characteristics of the complex motif are to be identified by this new exploratory method.

4.1 Matrix of co-occurrences

As we have said, the motivation for the method we propose is the need to avoid arbitrarily defined extraction parameters. Thus, instead of seeking motifs which conform to certain predefined criteria, we try to characterize certain features of the input sequences. To that effect, we begin by building a matrix of co-occurrences, $\mathcal{M}$, as we will explain below.

To build this matrix we will first need to identify all occurrences of sequences of very small length, λ, i.e., all λ-mers in the input sequences, $\mathcal{S} = \{S_1, \ldots, S_t\}$.

Let $L(\mathcal{S}) = \{m_1, \ldots, m_z\}$ be the list of all such λ-mers[1] , noting that $z \leq |\Sigma|^\lambda$. Fig. 4.1 shows a set of input sequences that we will use to illustrate the definitions given below. In this example we will use $\lambda = 2$ to make it easier to follow.

Definition 4.1 (List of occurrences of a λ-mer) *Let $\mathcal{S}$ be a set of input sequences and let $m \in L(\mathcal{S})$. $Occ_{i,\mathcal{S}}(m)$ denotes the set of coordinates of all occurrences of m in $S_i \in \mathcal{S}$.*

This set, $\mathrm{Occ}_{i,\mathcal{S}}(m)$, is therefore a list of integers denoting the positions at which we can find m on a sequence $S_i \in \mathcal{S}$. Whenever the set of input sequences, $\mathcal{S}$, is clear from the

[1]Good mathematical practice would advise us to denote the list of all λ-mers on a set of sequences $\mathcal{S}$ as $L_\lambda(\mathcal{S})$. We will, however, omit the λ lest our notation becomes too dense. We will assume that the value of λ is fixed and known across all definitions.

context we will simply write $\mathrm{Occ}_i(m)$. Concerning the example in Fig. 4.1, it is easy to see that $\mathrm{Occ}_3(\mathtt{GG}) = \{1, 9\}$, $\mathrm{Occ}_1(\mathtt{AA}) = \{2\}$ or that $\mathrm{Occ}_6(\mathtt{TT}) = \emptyset$.

Definition 4.2 (Configuration of a pair of λ-mers) *Let $\mathcal{S}$ be a set of input sequences. A configuration of a pair of λ-mers is a triple (m_r, m_s, d), with $m_r, m_s \in L(\mathcal{S})$ and $d \in \mathbb{Z} \setminus \{0\}$.*

In this definition we simply introduce a mathematical object which we call a configuration. This object is associated to a set of input sequences and will be used to denote the co-occurrence of a pair of λ-mers in a specific relative position.

Definition 4.3 (Configurations of a pair of λ-mers over a sequence $S_i \in \mathcal{S}$) *Let $\mathcal{S}$ be a set of input sequences and let m_r, $m_s \in L(\mathcal{S})$. $\Delta_{i,\mathcal{S}}(m_r, m_s)$ denotes the set of all configurations of m_r and m_s over $S_i \in \mathcal{S}$.*

$$\Delta_{i,\mathcal{S}}(m_r, m_s) = \{(m_r, m_s, d) : d = c_s - c_r, c_r \in Occ_{i,\mathcal{S}}(m_r), c_s \in Occ_{i,\mathcal{S}}(m_s), c_r \neq c_s\}$$

Note that if we consider the configuration (m_r, m_s, d) and if $d < \lambda$ then the configuration actually represents the occurrence of a $(\lambda + d)$-mer. It is also interesting to note that $\Delta_{i,\mathcal{S}}(m_s, m_r) = \{(m_s, m_r, d) : (m_r, m_s, -d) \in \Delta_{i,\mathcal{S}}(m_r, m_s)\}$. Once again, we will use $\Delta_i(m_r, m_s)$ every time the set of input sequences is clear from the context. Considering the previous example, we can observe that $\Delta_3(\mathtt{AA}, \mathtt{GG}) = \{(\mathtt{AA}, \mathtt{GG}, -2), (\mathtt{AA}, \mathtt{GG}, 6)\}$ or that $\Delta_6(\mathtt{AA}, \mathtt{TT}) = \emptyset$.

Definition 4.4 (Score of a configuration of a pair of λ-mers) *Let $\mathcal{S}$ be a set of input sequences and let m_r, $m_s \in L(\mathcal{S})$ and $d \in \mathbb{Z}$.*

$\mu_{i,\mathcal{S}} : \Sigma^\lambda \times \Sigma^\lambda \times \mathbb{Z} \mapsto \{0, 1\}$ is the membership function of a configuration with respect to the set of all configurations of the two λ-mers on an input sequence $S_i \in \mathcal{S}$, defined as:

$$\mu_{i,\mathcal{S}}(m_r, m_s, d) = \begin{cases} 1 & \text{if} \quad (m_r, m_s, d) \in \Delta_{i,\mathcal{S}}(m_r, m_s) \\ 0 & \text{otherwise} \end{cases}$$

$\sigma_{\mathcal{S}} : \Sigma^\lambda \times \Sigma^\lambda \times \mathbb{Z} \mapsto \{0, \ldots, |\mathcal{S}|\}$ is the score function for a configuration and is defined as:

$$\sigma_{\mathcal{S}}(m_r, m_s, d) = \sum_{i=1}^{|\mathcal{S}|} \mu_{i,\mathcal{S}}(m_r, m_s, d)$$

The score of a configuration of a pair of λ-mers is nothing more than the number of sequences where that particular configuration can be observed. Like in previous definitions, we will use $\sigma(m_r, m_s, d)$ without mentioning the set of input sequences whenever it is clear which set we are considering. In the example of Fig. 4.1 we have $\sigma(\texttt{AA}, \texttt{GG}, 7) = 3$, since GG occurs 7 positions after AA in sequences S_4, S_5 and S_6, yielding $\mu_{4,\mathcal{S}}(\texttt{AA}, \texttt{GG}, 7) = \mu_{5,\mathcal{S}}(\texttt{AA}, \texttt{GG}, 7) = \mu_{6,\mathcal{S}}(\texttt{AA}, \texttt{GG}, 7) = 1$.

Definition 4.5 (ε-tolerant score of a configuration of a pair of λ-mers) *Let $\mathcal{S}$ be a set of input sequences and let $m_r, m_s \in L(\mathcal{S})$ and $\varepsilon \in \mathbb{N}_0$. The ε-tolerant score of a configuration $\sigma^{\varepsilon}_{\mathcal{S}} : \Sigma^{\lambda} \times \Sigma^{\lambda} \times \mathbb{Z} \mapsto \mathbb{N}_0$ is defined as:*

$$\sigma^{\varepsilon}_{\mathcal{S}}(m_r, m_s, d) = \sum_{i=1}^{|\mathcal{S}|} \max_{k=-\varepsilon,\ldots,\varepsilon} \mu_{i,\mathcal{S}}(m_r, m_s, d+k) \qquad (d \neq 0)$$

Furthermore, $\sigma^{\varepsilon}_{\mathcal{S}}(m_r, m_s, 0) = 0$.

The concept of ε-tolerant score of a configuration of a pair of λ-mers addresses the need to allow for a configuration to have slight variations. This removes the strictness of requiring a pair of λ-mers to co-occur at fixed relative positions in order to have a high score. This can be illustrated by the example shown in Fig. 4.1 where $\sigma(\texttt{AA}, \texttt{GG}, 7) = 3$, $\sigma(\texttt{AA}, \texttt{GG}, 6) = 2$ and $\sigma(\texttt{AA}, \texttt{GG}, 5) = 1$ but $\sigma^1(\texttt{AA}, \texttt{GG}, 7) = 5$, $\sigma^1(\texttt{AA}, \texttt{GG}, 6) = 6$ and $\sigma^1(\texttt{AA}, \texttt{GG}, 5) = 3$. A 1-tolerant score is able to grasp the fact that the 2-mer AA co-occurs with GG in all input sequences at a distance of 6 ± 1 positions. Incidentally, $\sigma^1(\texttt{AA}, \texttt{GG}, 4) = 1$, despite the fact that $\sigma(\texttt{AA}, \texttt{GG}, 4) = 0$. This can be useful to describe patterns of co-occurrence that have a high ε-tolerant score even though they never actually occur in the input sequences.

Definition 4.6 (Most common configuration of a pair of λ-mers) *Let $\mathcal{S}$ be a set of input sequences and let $m_r, m_s \in L(\mathcal{S})$. A configuration (m_r, m_s, d^*) is said to a most common configuration of the two λ-mers if, for every configuration (m_r, m_s, d), $\sigma_{\mathcal{S}}(m_r, m_s, d^*) \geq \sigma_{\mathcal{S}}(m_r, m_s, d)$.*

Furthermore, we say it is a ε-tolerant most common configuration if the same assertion holds for the ε-tolerant score.

The notion of most common configuration will be used to find the configuration or, indeed, the configurations with the highest score for a pair of λ-mers. From the example shown in

	AA	AC	AG	AT	CA	CC	CG	CT	GA	GC	GG	GT	TA	TC	TG	TT
AA	0	3	2	2	1	3	2	6	2	2	6	4	3	3	3	2
AC	3	1	0	0	1	1	2	3	1	1	3	3	2	1	2	1
AG	2	0	1	1	1	2	0	2	0	2	2	0	2	2	0	0
AT	2	0	1	0	0	1	1	2	1	1	2	1	1	2	1	1
CA	1	1	1	0	0	1	0	1	0	1	1	1	1	1	1	0
CC	3	1	2	1	1	0	1	3	0	2	4	2	2	2	1	1
CG	2	2	0	1	0	1	0	2	1	1	2	2	1	1	1	1
CT	6	3	2	2	1	3	2	1	2	3	5	3	3	2	3	2
GA	2	1	0	1	0	0	1	2	0	1	2	2	0	1	2	1
GC	2	1	2	1	1	2	1	3	1	1	2	1	2	2	1	0
GG	6	3	2	2	1	4	2	5	2	2	1	4	3	3	3	2
GT	4	3	0	1	1	2	2	3	2	1	4	1	2	2	3	2
TA	3	2	2	1	1	2	1	3	0	2	3	2	2	2	1	1
TC	3	1	2	2	1	2	1	2	1	2	3	2	2	1	1	1
TG	3	2	0	1	1	1	1	3	2	1	3	3	1	1	1	1
TT	2	1	0	1	0	1	1	2	1	0	2	2	1	1	1	0

Figure 4.2: Matrix of co-occurrences, $\mathcal{M}^1$, for the input sequences of Fig. 4.1

Fig. 4.1 it is easy to see that the most common configuration for the pair (AA, GG) is (AA, GG, 7). The 1-tolerant most common configuration is, however, (AA, GG, 6).

We can now define a matrix of co-occurrences that gathers the information about the ε-tolerant score of the most common configuration of every pair of λ-mers.

Definition 4.7 (Matrix of co-occurrences) *Let $\mathcal{S}$ be a set of input sequences. A matrix of co-occurrences over $\mathcal{S}$ with ε tolerance, $\mathcal{M}^{\varepsilon}_{\mathcal{S}}$, is the matrix where each of its elements a_{ij} is defined as:*

$$a_{ij} = \sigma^{\varepsilon}_{\mathcal{S}}(m_i, m_j, d^*)$$

where (m_i, m_j, d^*) is a ε-tolerant most common configuration of $m_i, m_j \in L(\mathcal{S})$ and $i, j = 1, \ldots, |L(\mathcal{S})|$.

The matrix of co-occurrences , $\mathcal{M}^1$, derived from the input sequences of Fig. 4.1 is shown in Fig. 4.2. We can see, by inspecting the matrix and the input sequences, that there are two configurations with the maximum 1-tolerant score: (AA, CT, 3) and (AA, GG, 6). In this case, it is easy to see that AA co-occurs with CT in all sequences at a relative distance of 3 ± 1 and with GG, also in all sequences at a relative distance of 6 ± 1.

The matrix of Fig. 4.2 is symmetric. The next lemma will prove that it is always the case.

Lemma 4.1 *A matrix of co-occurrences, $\mathcal{M}^{\varepsilon}_{\mathcal{S}}$ is symmetric, i.e., $a_{ij} = a_{ji}$ for every $i, j = 1, \ldots, |L(\mathcal{S})|$.*

Proof

Let us assume there are $p, q \in \{1, \ldots, |L(\mathcal{S})|\}$ such that $a_{pq} \neq a_{qp}$. This entails the assertion that $\sigma^{\varepsilon}_{\mathcal{S}}(m_p, m_q, d^*_{pq}) \neq \sigma^{\varepsilon}_{\mathcal{S}}(m_q, m_p, d^*_{qp})$, where (m_p, m_q, d^*_{pq}) and (m_q, m_p, d^*_{qp}) are ε-tolerant most common configurations of (m_p, m_q) and (m_q, m_p), respectively.

Let us consider, without loss of generality, that $\sigma^{\varepsilon}_{\mathcal{S}}(m_p, m_q, d^*_{pq}) > \sigma^{\varepsilon}_{\mathcal{S}}(m_q, m_p, d^*_{qp})$

It follows that

$$\sum_{i=0}^{|\mathcal{S}|} \max_{k=-\varepsilon,\ldots,\varepsilon} \mu_{i,\mathcal{S}}(m_p, m_q, d^*_{pq} + k) > \sum_{i=0}^{|\mathcal{S}|} \max_{k=-\varepsilon,\ldots,\varepsilon} \mu_{i,\mathcal{S}}(m_q, m_p, d^*_{qp} + k)$$

It is easy to see that $\Delta_{i,\mathcal{S}}(m_p, m_q) = \{(m_p, m_q, d^*_{pq}) : (m_q, m_p, -d^*_{pq}) \in \Delta_{i,\mathcal{S}}(m_q, m_p)\}$. Therefore, for every sequence S_i we have that $\mu_{i,\mathcal{S}}(m_p, m_q, d^*_{pq}) = \mu_{i,\mathcal{S}}(m_q, m_p, -d^*_{pq})$. And consequently,

$$\sum_{i=0}^{|\mathcal{S}|} \max_{k=-\varepsilon,\ldots,\varepsilon} \mu_{i,\mathcal{S}}(m_p, m_q, d^*_{pq} + k) = \sum_{i=0}^{|\mathcal{S}|} \max_{k=-\varepsilon,\ldots,\varepsilon} \mu_{i,\mathcal{S}}(m_q, m_p, -d^*_{pq} + k)$$

But this means that $\sigma^{\varepsilon}_{\mathcal{S}}(m_q, m_p, -d^*_{pq}) > \sigma^{\varepsilon}_{\mathcal{S}}(m_q, m_p, d^*_{qp})$ which contradicts the fact that (m_q, m_p, d^*_{qp}) is a ε-tolerant most common configuration of (m_q, m_p).

□

Algorithm 1 computes the matrix of co-occurrences with ε-tolerance. We will now show that its time complexity is $O(N^2)$ where $N = \sum_{i=1}^{|\mathcal{S}|} |S_i|$. It is easy to see that the lists of occurrences for every λ-mer in every sequence can be obtained in a pre-processing stage in $O(N)$ time. The cycle from line 5 through line 18 considers all occurrences of all λ-mers in each sequence. There are exactly $\sum_{i=1}^{|\mathcal{S}|} |S_i| - \lambda + 1 < N$ such occurrences and therefore the number of possible pairs of occurrences is $O(N^2)$, which corresponds to the number of times the cycle will be invoked. Each operation in the cycle can be performed in $O(1)$ considering that ε is fixed and that the sets Conf_i can be implemented using two arrays of size $|S_i|$ (one for negative and another for positive values of d). The cycle from line 19 through line 21 is invoked at most $\sum_{i=1}^{|\mathcal{S}|}(|S_i| - \lambda + 1)^2 < N^2$ times since there can be no more than $(|S_i| - \lambda + 1)^2$

Algorithm 1 Computes the ε-tolerant matrix of co-occurrences

1: **for all** $m_r, m_s \in L(\mathcal{S})$ **do**
2: MaxScore $\leftarrow 0$
3: **for all** $S_i \in \mathcal{S}$ **do**
4: $\text{Conf}_i \leftarrow \emptyset$
5: **for all** $c_r \in \text{Occ}_i(m_r), c_s \in \text{Occ}_i(m_s)$ **do**
6: $d \leftarrow c_r - c_s$
7: **if** $d \neq 0$ **then**
8: $\text{Conf}_i \leftarrow \text{Conf}_i \cup \{(m_r, m_s, d)\}$
9: **for** $k = 1$ to ε **do**
10: **if** $d + k \neq 0$ **then**
11: $\text{Conf}_i \leftarrow \text{Conf}_i \cup (m_r, m_s, d + k)$
12: **end if**
13: **if** $d - k \neq 0$ **then**
14: $\text{Conf}_i \leftarrow \text{Conf}_i \cup (m_r, m_s, d - k)$
15: **end if**
16: **end for**
17: **end if**
18: **end for**
19: **for all** $(m_r, m_s, d) \in \text{Conf}_i$ **do**
20: $\text{Score}[(m_r, m_s, d)] \leftarrow \text{Score}[(m_r, m_s, d)] + 1$
21: **end for**
22: **end for**
23: **for all** $(m_r, m_s, d) \in \bigcup_i \text{Conf}_i$ **do**
24: **if** $\text{Score}[(m_r, m_s, d)] > \text{MaxScore}$ **then**
25: $\text{MaxScore} \leftarrow \text{Score}[(m_r, m_s, d)]$
26: **end if**
27: **end for**
28: $M[r, s] \leftarrow \text{MaxScore}$
29: **end for**

Complex motif:	TTGCAn_5TATTA	
Configurations of 4-mers:	(TTGC,TGCA,1)	(TGCA,TTGC,-1)
	(TTGC,TATT,6)	(TATT,TTGC,-6)
	(TTGC,ATTA,7)	(ATTA,TTGC,-7)
	(TGCA,TATT,5)	(TATT,TGCA,-5)
	(TGCA,ATTA,6)	(ATTA,TGCA,-6)
	(TATT,ATTA,1)	(ATTA,TATT,-1)

Figure 4.3: Configurations of 4-mers induced by the presence of a complex motif

configurations of pairs of λ-mers in a sequence. The same can be said for the cycle from line 23 through line 27. This yields a time complexity of $O(N + N^2 + N^2 + N^2) = O(N^2)$. In terms of space, the lists of occurrences of λ-mers combined will take $O(N)$ space. The same can be said of the arrays implementing the Conf_i sets. Similarly, the Score attribute of each configuration can be implemented with a pair of arrays, taking $O(N)$ space for each pair of λ-mers. Since these arrays can be re-used for each pair, they will take no more than $O(N)$ space. The matrix itself requires $O(|L(\mathcal{S})|^2 \leq |\Sigma|^{2\lambda})$ space. The total space requirements are therefore in $O(N + |\Sigma|^{2\lambda})$.

4.2 Biclustering approach

As we have said, the matrix of co-occurrences gives us a view, for each pair of λ-mers, of the abundance of its most common configuration (or configurations). The next step is to try to combine these configurations to form larger patterns, possibly complex motifs. In doing so, we are guided by the score values computed during the construction of the matrix.

Consider the example in Fig. 4.3. The presence of a complex motif with two components of length 5 each separated by a distance of 5 nucleotides induces 12 configurations of pairs of 4 different 4-mers. Let us suppose that this complex motif is present in exactly 8 different input sequences. The score of each of these configurations is therefore no lower than 8. Admitting that none of these configurations occur in other input sequences, Fig. 4.4 represents a sub-matrix of the matrix of co-occurrences that would be generated.

This example illustrates the basic principle of our approach to the inference of complex

	ATTA	TATT	TGCA	TTGC
ATTA	0	8	8	8
TATT	8	0	8	8
TGCA	8	8	0	8
TTGC	8	8	8	0

Figure 4.4: Sub-matrix induced by TTGCAn_5TATTA, assuming that it occurs in 8 input sequences

motifs using the matrix of co-occurrences. However, what we set out to do is the reverse of the reasoning shown in this example, i.e., we want to identify certain patterns in the matrix of co-occurrences that could indicate the presence of a complex motif.

We begin by characterizing the patterns we are looking for.

Definition 4.8 (Diagonally-punctured bicluster in a matrix of co-occurrences) *A diagonally-punctured bicluster, $B(I,\Lambda)$, in a matrix of co-occurrences $\mathcal{M}$ is a sub-set of the elements a_{ij} of $\mathcal{M}$ described by a pair (I,Λ), with $\Lambda \subseteq I$, and defined as:*

$$B(I,\Lambda) = \{a_{ij} : i \neq j, i \in I, j \in I\} \cup \{a_{ii} : i \in \Lambda\}$$

with $i,j = 1,\ldots,|L(\mathcal{S})|$.

A diagonally-punctured bicluster in a matrix of co-occurrences is, therefore, an object that roughly corresponds to a square sub-matrix of $\mathcal{M}$ except that the elements in the diagonal have an optional membership.

This is an unconventional type of bicluster for two reasons. Firstly, the columns that belong to the bicluster are entirely defined by the indices of the rows (and vice-versa) and, secondly, the diagonal elements are not necessarily included in the set of elements of the bicluster. This is, arguably, not a bicluster at all but since we lack a more appropriate term we will still call it a bicluster bearing in mind its special characteristics.

Definition 4.9 (h-valid diagonally-punctured bicluster in a matrix of co-occurrences) *An h-valid diagonally-punctured bicluster, $B(I,\Lambda)$, in a matrix of co-occurrences $\mathcal{M}$ is a diagonally-punctured bicluster such that $a_{ij} \geq h$ for every $a_{ij} \in B(I,\Lambda)$.*

The sub-matrix of Fig. 4.4 illustrates this concept. We can think of it as a diagonally-punctured bicluster where I corresponds to the set of indices of the 4-mers `ATTA`, `TATT`, `TGCA` and `TTGC`, and where $\Lambda = \emptyset$. If this is the case, we are in the presence of an 8-valid diagonally-punctured bicluster.

Definition 4.10 (Cut of height h in a matrix of co-occurrences) *A cut of height h in a matrix of co-occurrences, $\mathcal{M}$, $C_h(\mathcal{M})$ is a set of its elements defined as:*

$$C_h(\mathcal{M}) = \{a_{ij} : a_{ij} = h\}$$

with $i, j = 1, \ldots, |L(\mathcal{S})|$.

The notion of cut in a matrix of co-occurrences will be useful later. At this point it is only worth noting that all h-valid diagonally-punctured biclusters have their elements in $\bigcup_{l=h}^{|\mathcal{S}|} C_l(\mathcal{M})$.

Let us recall that we are looking for patterns in the matrix of co-occurrences that can indicate the presence of a complex motif. We are interested in identifying diagonally-punctured biclusters that include as many elements of the matrix as possible and are h-valid for the highest value of h attainable. Such a bicluster would hopefully signal the presence of a complex motif in as many as h different input sequences. As we have remarked earlier, a simple motif is just a particular case of a complex motif and a diagonally-punctured bicluster could, in fact, indicate the presence of a simple motif of length greater than λ. For instance, the motif `AAATT` induces the following configurations of 4-mers[2] : $(\texttt{AAAT}, \texttt{AATT}, 1)$ and $(\texttt{AATT}, \texttt{AAAT}, -1)$, which would correspond to a diagonally-punctured bicluster in the matrix of co-occurrences (provided the motif was frequent enough across the input sequences).

This approach thinks of complex motifs (and simple motifs) as compositions of configurations of λ-mers that will be shown in the matrix of co-occurrences in the form of diagonally-punctured biclusters. However, since we consider only the most common configurations of pairs of λ-mers some information can be lost. In addition, the input sequences can contain many complex motifs that will in turn induce many configurations of λ-mers, possibly interfering with the configurations induced by other complex motifs. It is important, then, to

[2]It also induces 6 configurations of 3-mers, 12 configurations of 2-mers, etc. The impact of the choice of the value of λ will be discussed later.

systematize what we can and what we cannot hope to find in searching for biclusters in the matrix of co-occurrences.

Firstly, we should note that this method is unable to identify simple motifs of length λ or shorter. This is due to the fact that the matrix of co-occurrences only considers configurations of pairs of λ-mers, (m_r, m_s, d). All simple motifs that can be identified are $(\lambda + d)$-mers. Likewise, complex motifs whose components are shorter than λ will not be identified.

Secondly, motifs composed of λ-mers which repeat more than twice will induce multiple configurations of the same pair of λ-mers with identical scores. Consider, for example, the motif $\texttt{AAATTT}n_3\texttt{AATTAAT}$ and suppose we choose $\lambda = 3$. This motif will induce many configurations of 3-mers, including $(\texttt{AAT}, \texttt{AAT}, 8)$, $(\texttt{AAT}, \texttt{AAT}, 12)$ and $(\texttt{AAT}, \texttt{AAT}, 4)$. These configurations will have the same score and will be represented in the matrix of co-occurrences by a single element in the main diagonal. Furthermore, the configurations $(\texttt{AAT}, \texttt{ATT}, 1)$ and $(\texttt{AAT}, \texttt{ATT}, 9)$ will also be induced and will be represented by the same pair of elements in the matrix of co-occurrences. This is not a problem, in principle, for it is still possible to infer the structure of the complex motif from these configurations. But the smaller the value we choose for λ the more likely it is that configurations of λ-mers unrelated with the complex motif be included in the bicluster. For instance, if the motif $\texttt{AATAAT}$ is at least as common as the complex motif we are considering, then, the configurations it induces will pollute those induced by the complex motif and will effectively undermine our ability to infer the structure of the complex motif.

Finally, an interesting motif could fall short from being identified. This can happen if another motif composed by the same set of λ-mers (or a superset) is more frequent. Consider the complex motif we discussed above and let $\lambda = 3$. If the motif $\texttt{ATTT}n_3\texttt{TAAT}$ is a different binding site but present in less sequences than $\texttt{AAATTT}n_3\texttt{AATTAAT}$ then it will be undetectable for the configurations it induces will not be most common configurations of the 3-mers that compose it. It is worth noting that if it occurred in more sequences than the previous motif it would have compromised our ability to reconstruct it from the configurations it induces since some would have been superseded by those with a higher score. If it occurred in exactly the same number of sequences it would result in a merger of the sets of most common configurations induced by each motif which would likewise make the task of inferring the motifs much harder.

This illustrates the fact that the choice of λ is critical. If it is too large it may miss smaller motifs and if it is too small it will render our method vulnerable to spurious configurations

interfering with interesting motifs or similar motifs interfering with each other. These shortcomings of our approach, albeit numerous, concern situations which are very unlikely for an appropriate choice of λ and are here presented for the sake of a thorough discussion.

It may happen, however, that by chance several unrelated configurations of pairs of λ-mers have identical scores and end up being grouped to form a bicluster. These false positives may be detected by considering each contributing configuration because it is unlikely that the distance values are compatible in the sense that they cannot be combined to form a motif. An easier way to detect these spurious biclusters would be to keep, for each configuration, information about which sequences it occurred in. This would allow us to determine that these configurations were occurring in different sets of sequences and were therefore unrelated.

If two motifs share co-occurring pairs of λ-mers regardless of whether they concern the same configurations or not, then the score of the matrix element representing the shared pair will take the value corresponding to the most frequent motif across input sequences, i.e., the score of the most common configuration. This gives us our particular plaid model:

$$a_{ij} = \max_{k=1,\ldots,K} \theta_k \rho_{ijk}$$

where θ_k represents the contribution of the kth bicluster to the value of a_{ij} and ρ_{ijk}, is a binary value representing the membership of the element a_{ij} to the kth bicluster. If the interference refers to different configurations then we may not be able to reconstruct the less frequent motif. However, if it refers to the same configuration we can identify both.

This gives us the right cue for the algorithm we propose. Algorithm 2 begins by considering the matrix of co-occurrences, $\mathcal{M}^{\epsilon}_{\mathcal{S}}$, starting with the elements in $C_h(\mathcal{M}^{\epsilon}_{\mathcal{S}})$ with the highest score, h. Since this matrix is symmetrical, our starting point is either an element on the main diagonal or a pair of elements from the upper and lower triangle respectively. In either case, it is a diagonally-punctured bicluster. For each of these biclusters we will then greedily add rows/columns as long as the corresponding elements have a score not lower then the score of our initial elements. The same is performed for the diagonal elements. Elements which have already been included in a bicluster are not used as a starting point for ulterior biclusters. This way we are effectively seeking biclusters with high scores which include as many matrix elements as possible. By allowing initial biclusters to include elements with a higher score than the score of the original elements we are addressing the observation we made while discussing

our plaid model. This can be translated in the fact that a configuration which occurs in h input sequences will *a fortiori* also occur in $h' < h$ input sequences. The algorithm will continue considering elements with decreasing scores until a minimum score is reached, below which any bicluster is deemed insufficiently common to be of interest.

The algorithm will consider at most $|L(\mathcal{S})|^2 \leq |\Sigma|^{2\lambda}$ matrix elements as the starting point and to each of these initial biclusters will add at most $|L(\mathcal{S})| \leq |\Sigma|^{\lambda}$ rows/columns. At each tentative addition of rows/columns it will have to check whether the resulting bicluster is h-valid resulting in at most $|L(\mathcal{S})|^2 \leq |\Sigma|^{2\lambda}$ comparisons. This yields a time complexity of $O(|\Sigma|^{4\lambda})$. However, the larger the biclusters the less matrix elements will be used as a starting point, so this bound is not tight. Determining a tighter bound is quite difficult since the relation between the average size of the biclusters and the number of initial elements considered is not easily established due to the fact that different biclusters can effectively share many matrix elements.

Algorithm 2 is a heuristic approach to our problem in the sense that it misses an undefined part of the solution. In effect, it can determine at most $|L(\mathcal{S})|^2$ different biclusters. There are, however, as many as $3^{\frac{|L(\mathcal{S})|}{3}}$ possible biclusters, as we shall see.

Consider a matrix of co-occurrences and another matrix with the same size. Each element of this new matrix holds the value 1 if the corresponding score in the matrix of co-occurrences is not below h and 0 otherwise. This binary matrix can be seen as a graph $G = (V, E)$. Each row/column is a vertex and each pair of vertices is connected by an edge if the corresponding element in the binary matrix has the value 1. We can effectively ignore the values held by the main diagonal for this discussion. Searching for all largest diagonally-punctured biclusters in this binary matrix is the same as searching for all maximal cliques in the corresponding graph. This problem is known to be NP-hard and there can be as many as $3^{\frac{|V|}{3}}$ maximal cliques in a graph [56].

Our algorithm is, therefore, trying to solve the equivalent to the problem of enumerating all maximal cliques for each score h it considers. In fact, we do not actually require the biclusters to be maximal and since the binary matrix for each score h tends to be sparse the number of maximum size biclusters is likely to be much smaller than the theoretical maximum.

Once we have identified the diagonally-punctured biclusters of interest we can then try to reconstruct the motif that induced the configurations we have grouped. To that effect,

Algorithm 2 Extracts biclusters in a matrix of co-occurrences

1: **for** $h = t$ to minscore **do**
2: $\text{biclusters}_h \leftarrow \emptyset$
3: **for all** $a_{ij} \in C_h(\mathcal{M})$ **do**
4: **if** $a_{ij} \notin \bigcup_{B_k \in \text{biclusters}_h} B_k$ **then**
5: **if** $i = j$ **then**
6: $I \leftarrow \{i\}$
7: $\Lambda \leftarrow \{i\}$
8: **else**
9: $I \leftarrow \{i, j\}$
10: $\Lambda \leftarrow \emptyset$
11: **end if**
12: **for** $k = 1$ to $|L(\mathcal{S})|$ **do**
13: **if** $B(I \cup \{k\}, \Lambda)$ is h-valid **then**
14: $I \leftarrow I \cup \{k\}$
15: **if** $B(I, \Lambda \cup \{k\})$ is h-valid **then**
16: $\Lambda \leftarrow \Lambda \cup \{k\}$
17: **end if**
18: **end if**
19: **end for**
20: $\text{biclusters}_h \leftarrow \text{biclusters}_h \cup \{B(I, \Lambda)\}$
21: **end if**
22: **end for**
23: **end for**

we need to have previously taken note of the possibly multiple most common configurations associated with each element in the matrix of co-occurrences. These configurations can easily be assembled to reconstruct the original motif unless they have been polluted by configurations induced by other motifs as we have already discussed. For small biclusters this assembly can be done by inspection. A general method of assembly is left for future work. However, as a first approach we can think of a weighted multi-graph whose vertices correspond to the λ-mers participating in the bicluster and whose edges are labeled with the relative distances between each pair of λ-mers as indicated by the most common configurations in each matrix element grouped in the bicluster under consideration. A traversal of this graph will purportedly be able to perform the assembly of the original motif.

In this chapter we described a new method that can effectively guide the parameter specification for modern motif finders by estimating the number of components and the component length for complex motifs (and simple motifs, which are a particular case). It cannot, however, give a reliable indication of the number of sequences in which the reconstructed motifs occur in. An h-valid bicluster is simply a bicluster whose elements, i.e., whose configurations of λ-mers occur in no less than h input sequences. We keep no information about which sequences they actually occur in so we cannot confidently say that they all refer to the same set of sequences. For this reason, and for the fact that it is, in effect, simply an heuristic approach it cannot compete with motif finders. It can, instead, be used as a tool to capture the characteristics of the input sequences and collect evidence of the presence of interesting motifs which could otherwise go unnoticed.

In the next chapter we present the results of the application of this method to both synthetic and biological data sets.

Chapter 5

Results

In this chapter we present and discuss the result of applying the method proposed in this thesis to several data sets. The method was applied to both artificially generated (synthetic) data sets and to a real data set. The advantage of using synthetic data sets is the ability to specify exactly which motifs we wish to plant in the data against a random background. This way we can safely test our method since we control every aspect of the signal hidden in the random sequences. However, synthetic data sets are still very different from real sequences in the sense that these cannot be accurately modeled as motifs with a role in transcription regulation surrounded by meaningless nucleotides. Regulatory regions are the result of the interference of various signals which are important for different processes. The distribution of nucleotides in these regions is not random since it is influenced by many factors like the evolutionary history of the species. Other restrictions come from the very nature of these regions which need to be easily accessed by the transcription initiation complex and are therefore usually richer in A-T content (A-T bonds are weaker than C-G bonds). It is, therefore, important to test our method with real data. To this effect we chose to apply it to a well characterized data set [57] for which a binding site has been determined with high confidence.

5.1 Synthetic Data

The synthetic data sets that we will describe in this section were produced using a simple random generator based on `ran2` [58]. Each data set, unless otherwise indicated, consists of 100 sequences of length 600. The length was chosen to be 600 to conform to the average length

of the sequences that will be used in the analysis of real data.

There are two important parameters in our method: λ which defines the length of our λ-mers and ε which defines the tolerance with which we score configurations of λ-mers. Recall that the ε-tolerant score of a configuration, (m_r, m_s, d), considers the contribution of all configurations (m_r, m_s, d') such that $d' \in [d - \varepsilon, d + \varepsilon]$.

We did not consider the cases where $\lambda \leq 2$ because, as we mentioned in the previous chapter, this will increase the number of most common configurations in each element of the matrix of co-occurrences making the task of identifying motifs harder. The cases in which $\lambda > 4$ have two inconvenients. Not only the generated matrix is exceedingly large but we will also be unable to identify complex motifs with components shorter than 5 nucleotides or simple motifs less than 6 nucleotides long. Our results will, therefore, only show the cases where $\lambda = 3$ and $\lambda = 4$. We will also only consider the cases where $\varepsilon = 0$ and $\varepsilon = 1$ since larger values for our tolerance will inflate the score of most configurations. In any case, we do not discard the interest of performing a broader study.

Recall that Algorithm 2, which identifies diagonally-punctured biclusters also uses the parameter `minscore` referring to the minimum score required for each element of a bicluster to allow it to be identified. In every analysis performed in this chapter we have considered `minscore` $= 10$.

In the following discussion we will refer to score levels or simply levels to talk about features which become apparent when looking at elements in the matrix of co-occurrences with a given score. Therefore, when we refer to all elements at level h we are referring to all elements in the matrix of co-occurrences, $\mathcal{M}$, with score h or, equivalently, to all elements in $C_h(\mathcal{M})$. Similarly, when we refer to biclusters identified at level h we mean all biclusters whose elements with least score are at level h (h-valid biclusters).

5.1.1 No planted motifs

The first step in this analysis is to characterize the noise. That is, we want to characterize the output of our method when applied to random sequences with no planted motifs. This gives us a baseline with which to compare results.

In Fig. 5.1 we can see the superposition of the distribution of the scores of configurations (matrix elements) from three different random data sets with no planted motifs. We can

distinguish three local maxima in the graph: one around level 90, another close to level 45 and yet another slightly above level 25. We argue that the maximum at the highest level corresponds to the number of expected occurrences of motifs of length 4 in a data set with these characteristics. Each motif of length 4 has a probability of $(\frac{1}{4})^4 = \frac{1}{256}$. In a sequence of length 600 it has many opportunities to occur. A naive approach would indicate that given a probability of $\frac{1}{256}$ and $600 - 4 + 1 = 597$ opportunities of occurrence one would expect 2.33 occurrences of motifs of length 4 in every sequence, yielding a score of 100 for each corresponding configuration. However, this approach ignores the fact that not all motifs can be overlapped (thus not having those many opportunities of occurrence). The true expected number of occurrences is surely below 2.33. So we would expect our local maximum to be somewhere close to but below 100. The same reasoning can be applied to motifs of length 5 yielding an expected number of occurrences close to 0.58 and could, therefore, explain the second local maximum. This local maximum is higher simply because more elements of the matrix are required to compose a motif of length 5 than a motif of length 4. The remaining local maximum is simply the combination of the expected number of occurrences for less likely patterns. It is also worth noting that the height of the maxima at lower levels depends on the height of the maxima at higher levels. This is due to the fact that the score of a configuration refers only to a most common configuration. Therefore, if a matrix element is commited to represent a motif it cannot be used to represent a less likely motif.

Using this model we can predict the shape of a similar plot for data sets with varying sequence length. For a data set with longer sequences we expect the maximum referring to motifs of length 4 to be higher and closer to level 100 reflecting both the increased number of matrix elements commited to represent these motifs and the greater likelihood of their occurrence in input sequences. In a data set with shorter sequences we expect all maxima to be at lower levels and to be closer to one another. Fig. 5.2 shows the number of matrix elements per score level of a data set with no planted motifs and with sequences of length 2000. Fig. 5.3 shows the same but for a data set with sequences of length 100.

Another interesting plot is the number of identified biclusters per score level. Fig. 5.4 shows this information for the data sets analyzed in Fig. 5.1. Both plots have a similar shape showing that the number of biclusters for a given score level is, in this case, highly correlated with the number of matrix elements with the same score. This is not suprising, especially if

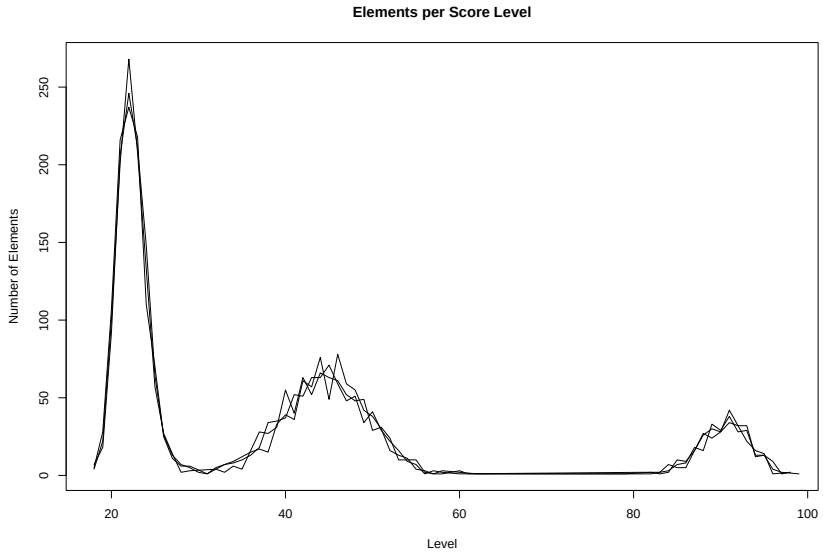

Figure 5.1: Number of Elements per Score Level in 3 synthetic data sets without planted motifs ($\lambda = 3, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

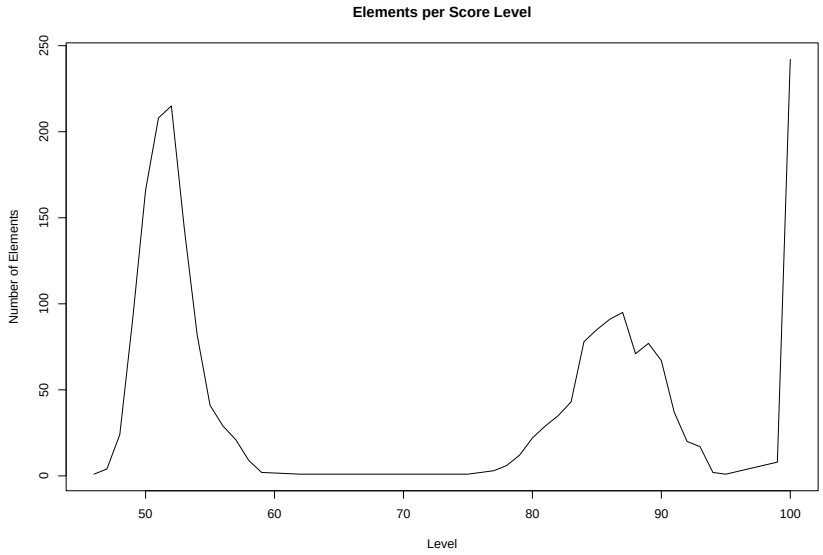

Figure 5.2: Number of Elements per Score Level in a synthetic data set without planted motifs ($\lambda = 3, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 2000$)

the identified biclusters have a low number of elements. Let us define the volume of a bicluster, $B(I, \Lambda)$, as the number of matrix elements covered by $B(I, \Lambda)$. Fig. 5.5 shows the average volume of biclusters per score level for the same data sets. We can see that the average volume

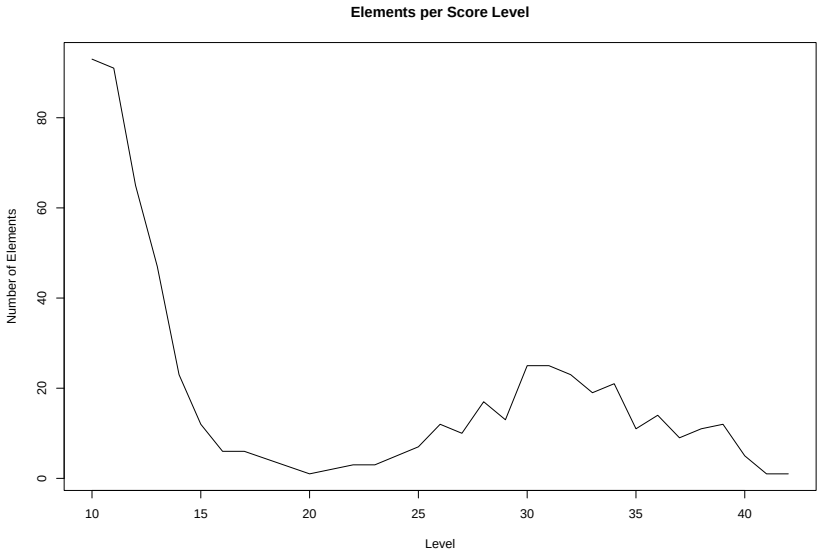

Figure 5.3: Number of Elements per Score Level in a synthetic data set without planted motifs ($\lambda = 3, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 100$)

of biclusters is very low (below 8, but mostly around 1-2) down until around score level 60. Below this point many spurious biclusters are identified and below level 20 they cover almost the entirety of the co-occurrence matrix.

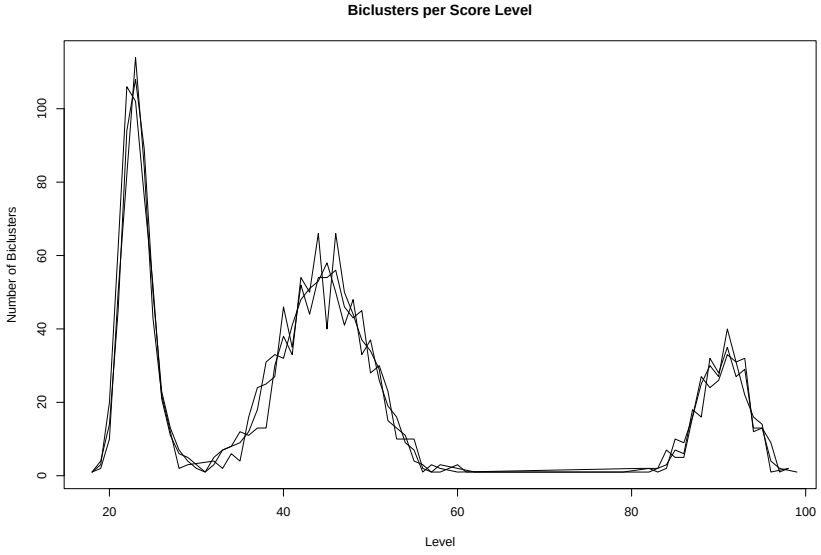

Figure 5.4: Number of Biclusters per Score Level in 3 synthetic data sets without planted motifs ($\lambda = 3, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

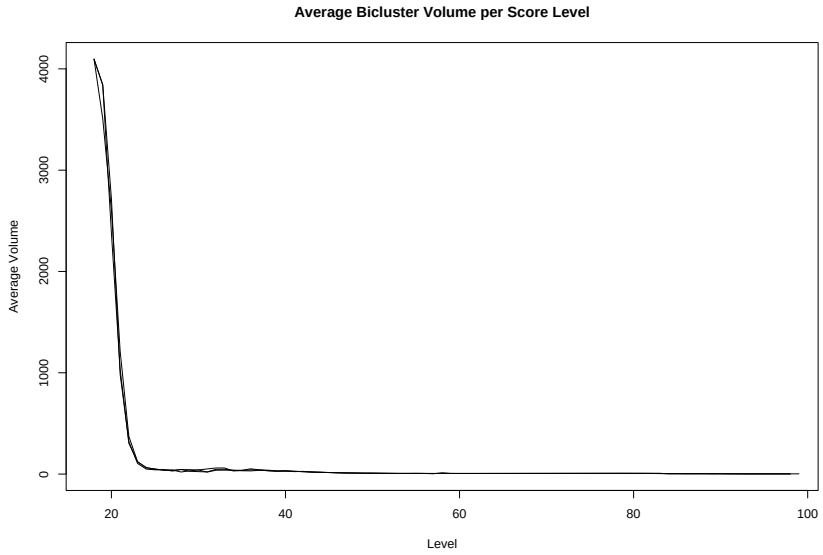

Figure 5.5: Average Bicluster Volume per Score Level in 3 synthetic data sets without planted motifs ($\lambda = 3, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

These results show that if we planted a motif in these data sets and made it occur in less than 20 input sequences it would be indistinguishable from the noise. The same, however, does not happen if we analyze the same data sets but having $\lambda = 4$. In Fig. 5.6 we present the results for this analysis.

By using $\lambda = 4$ our method becomes oblivious to motifs of length 4 which explains the disappearance of the corresponding local maximum in the number of elements per score level plot. The overall magnitude of the noise is also greatly decreased (note that the co-occurrence matrix in this case has 65536 elements, compared to the 4096 elements in the case where $\lambda = 3$). The highest scoring most common configuration is now down to a much lower level (level 62) and the average bicluster volume increases with decreasing score levels with a much gentler slope.

It is clear that a higher tolerance level will increase the noise in any data set. The only reason why our method considers using tolerance at all is the fact that in some cases it may increase the signal more efficiently than the noise. The only question at this point is how much increase in noise is one to expect. Fig. 5.7 gives us an idea of the impact of considering $\varepsilon = 1$ for the same three data sets considered so far, maintaining $\lambda = 4$.

We can see a significant increase in noise magnitude, especially for lower score levels. We

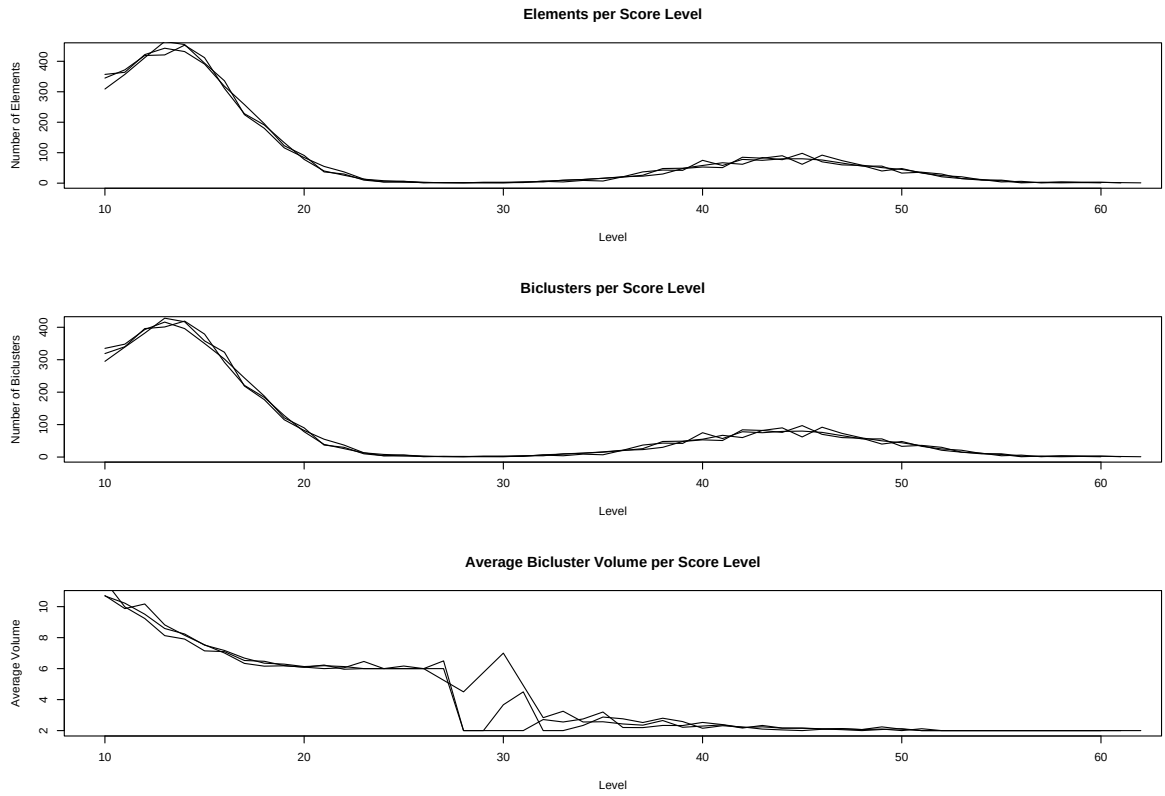

Figure 5.6: Number of Elements, Number of Biclusters and Average Bicluster Volume per Score Level in 3 synthetic data sets without planted motifs ($\lambda = 4, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

should, therefore, be conservative in using tolerant scores.

5.1.2 Planted Motifs

We will now address the case where we plant motifs in the input sequences. As we mentioned earlier, all data sets have 100 sequences with 600 nucleotides. Tab. 5.1 summarizes the cases we shall consider, indicating which motifs were planted and the percentage of input sequences containing the planted motifs. The motifs were planted only once in each of the randomly selected input sequences.

For each case we indicate what is to be expected and we present a general description of the output of our method. In all cases only parameters $\lambda = 4$ and $\varepsilon = 0$ are used.

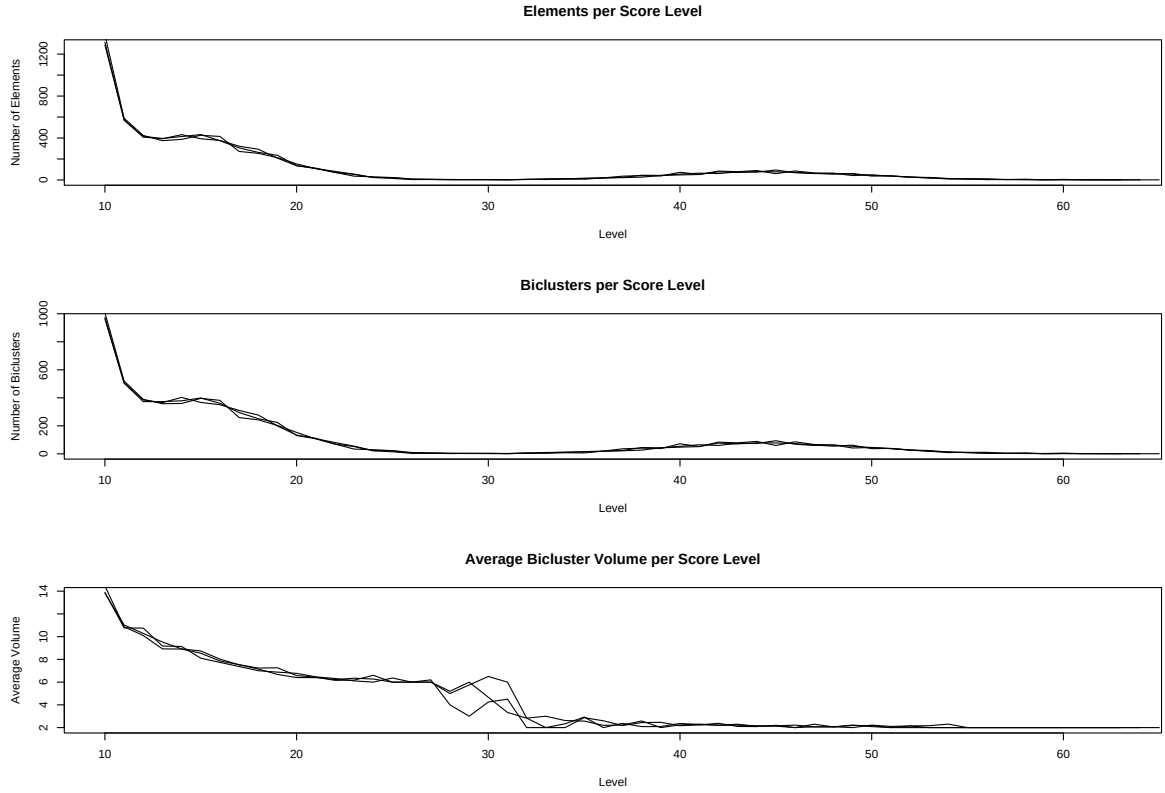

Figure 5.7: Number of Elements, Number of Biclusters and Average Bicluster Volume per Score Level in 3 synthetic data sets without planted motifs ($\lambda = 4, \varepsilon = 1, |\mathcal{S}| = 100, |S_i| = 600$)

Case	**Motif(s)**	**% of Input Sequences**
A	AAAAA	80%
B	AAAAn_5TTTT	80%
C	AAAATn_{20}TTTTA	80%
D	AAAATn_5TTTTAn_5CCCCT	80%
E	AAAAA	40%
	AAAATn_{20}TTTTA	30%
	AAAATn_5TTTTAn_5CCCCT	30%

Table 5.1: Motifs planted in synthetic data sets and the percentage of sequences containing them for cases A, B, C, D and E

Case A

In this case we have planted the motif AAAAA which generates the following configurations: (AAAA, AAAA, 1) and (AAAA, AAAA, −1). These configurations are associated with a single diagonal element in the matrix of co-occurrences (once they prove to be most common configurations). We expect to extract the bicluster $B(\{i\}, \{i\})$ where i is the index of the 4-mer AAAA.

Fig. 5.8 shows the relevant results of the application of our method to case A. The plots are almost identical to the ones obtained for the random data sets with no planted motifs, shown in Fig. 5.6. The difference lies in the fact that biclusters were found above score level 62 and the existence of some small perturbations around level 60. The introduction of our motif in the otherwise random sequences changes the proportion of nucleotides and, therefore, the likelihood of occurrence of some motifs. Tab. 5.2 shows the top 5 scoring motifs identified with our method. The assembly of the motifs from the identified biclusters was performed by inspection.

#	**Motif**	**Score**
1	AAAAA	85
2	TGAAA	64
3	AAAAG	63
4	GAAAA	62
5	AAAAC	61
⋮	⋮	⋮
4271		

Table 5.2: Top scoring motifs inferred from the top scoring biclusters for case A

As expected, the top scoring motif is the one which was planted. The following motifs have clearly benefited from the increased proportion of A's in the data set but still score significantly less than the planted motif and not much higher than the top scoring motif found in data sets with no planted motifs. It is also interesting to note that the planted motif scores higher than what is warranted by the number of sequences in which it was planted. This is also not surprising since the motif in itself is very likely to occur in a random sequence of

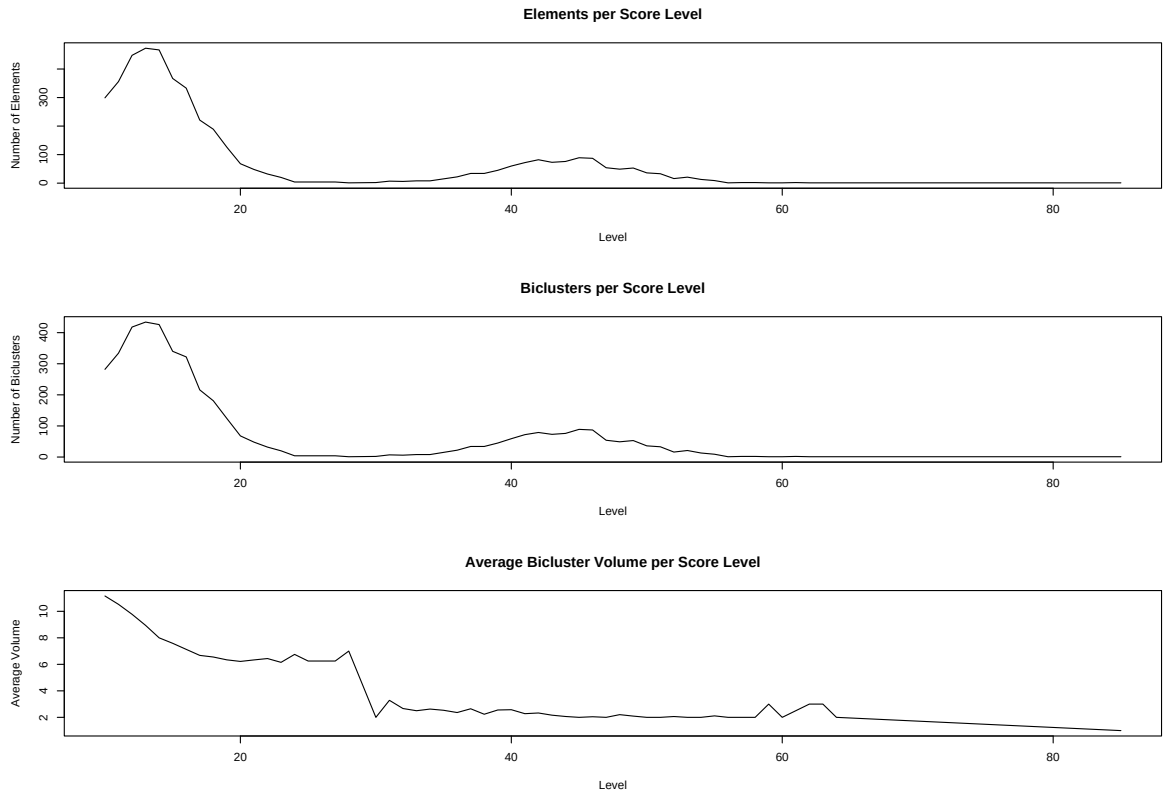

Figure 5.8: Number of Elements, Number of Biclusters and Average Bicluster Volume per Score Level for case A ($\lambda = 4, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

the specified length. The planted occurrences and the spurious occurrences have combined to yield a score of 85.

Case B

In this case we have planted the motif $\texttt{AAAA}n_5\texttt{TTTT}$. It generates the following configurations: $(\texttt{AAAA}, \texttt{TTTT}, 8)$ and $(\texttt{TTTT}, \texttt{AAAA}, -8)$. Fig. 5.9 summarizes the results obtained for this case.

The plots shown for this case are quite similar to those obtained in the previous case, as expected. Likewise, the top scoring motif is expected to be the planted motif and the following are likely to be simple motifs derived from the two components of the complex motif that was placed in 80 of the input sequences. Tab. 5.3 shows the top scoring motifs, confirming our predictions. Unlike the previous case, the score of the planted motif was not inflated by

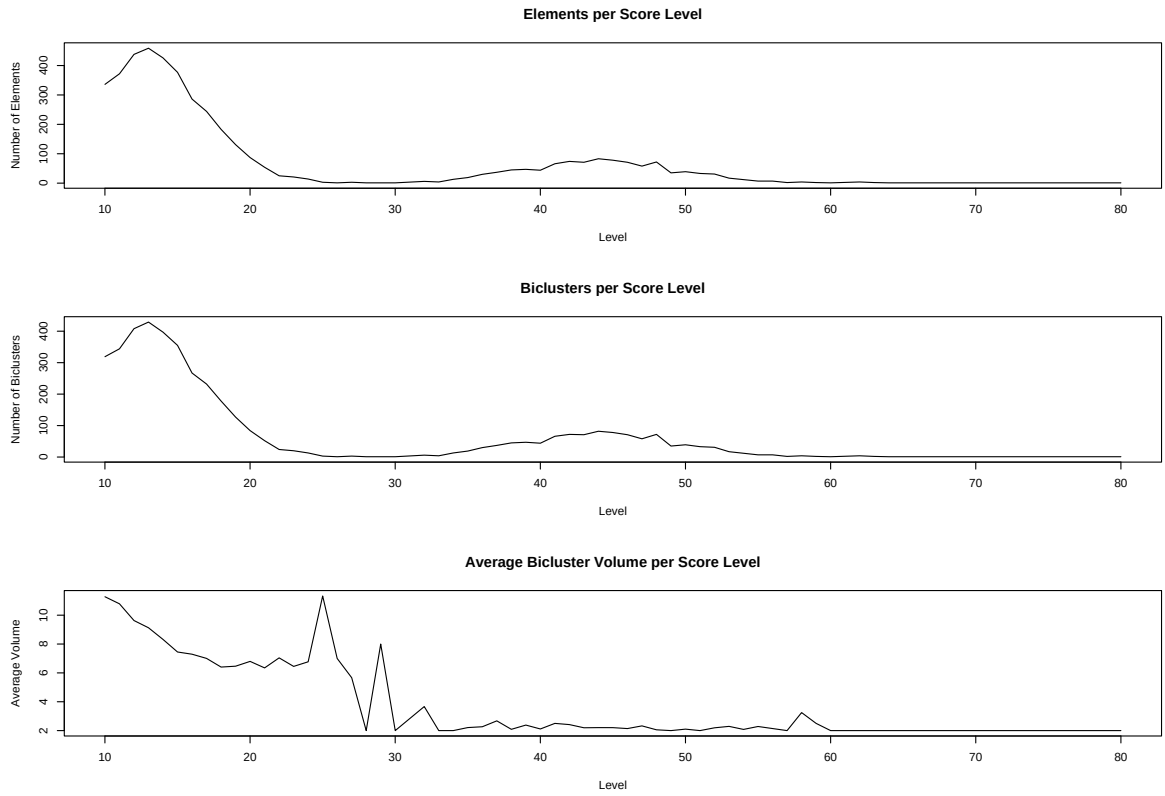

Figure 5.9: Number of Elements, Number of Biclusters and Average Bicluster Volume per Score Level for case B ($\lambda = 4, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

pre-existing occurrences in the random sequences. This is due to the fact that a motif with non-contiguous components is very unlikely to occur by chance, as we have mentioned in previous chapters.

Case C

The planted motif for case C is $\texttt{AAAAT}n_{21}\texttt{TTTTA}$, which generates the following set of configurations: $(\texttt{AAAA}, \texttt{AAAT}, 1)$, $(\texttt{AAAT}, \texttt{AAAA}, -1)$, $(\texttt{TTTT}, \texttt{TTTA}, 1)$, $(\texttt{TTTA}, \texttt{TTTT}, -1)$, $(\texttt{AAAA}, \texttt{TTTT}, 24)$, $(\texttt{TTTT}, \texttt{AAAT}, -24)$, $(\texttt{AAAA}, \texttt{TTTA}, 25)$, $(\texttt{TTTA}, \texttt{AAAA}, -25)$, $(\texttt{AAAT}, \texttt{TTTT}, 23)$, $(\texttt{TTTT}, \texttt{AAAT}, -23)$, $(\texttt{AAAT}, \texttt{TTTA}, 24)$ and $(\texttt{TTTA}, \texttt{AAAT}, -24)$. Fig. 5.10 shows the plots obtained for this case. It is interesting to note the peak on the plot of the average bicluster volume per score level that appears at level 80. There is only one bicluster at this level and it refers to the planted motif

#	Motif	Score
1	AAAAn_5TTTT	80
2	TTTTC	64
3	AAAAC	63
4	CAAAA	63
5	AAAAT	62
⋮	⋮	⋮
4272		

Table 5.3: Top scoring motifs inferred from the top scoring biclusters for case B

#	Motif	Score
1	AAAAT	90
2	TTTTA	90
3	AAAATn_{20}TTTTA	80
4	GAAAA	65
5	AAATG	63
⋮	⋮	⋮
4291		

Table 5.4: Top scoring motifs inferred from the top scoring biclusters for case C

which induces a bicluster of volume 12 (corresponding to the number of configurations listed above).

Contrarily to previous cases, the top scoring motif is not the planted motif, as shown in Tab. 5.4. Each component of the planted complex motif has combined with random occurrences of identical 5-mers to obtain a score higher than that of the planted motif.

Case D

In case D, a complex motif with three components was planted in the input sequences: AAAATn_5TTTTAn_5CCCCT. This motif generates as many as 30 configurations of pairs of 4-mers. Fig. 5.11 shows the relevant information for this case. Just like in case C, the planted

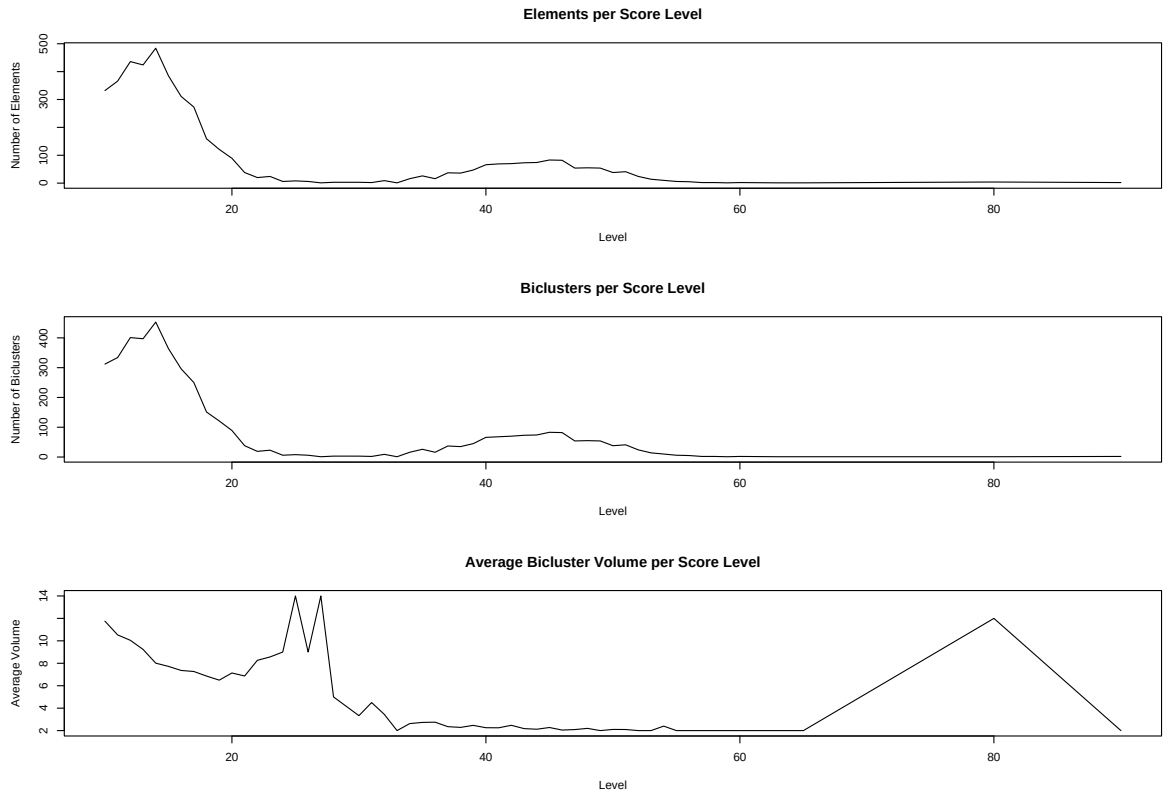

Figure 5.10: Number of Elements, Number of Biclusters and Average Bicluster Volume per Score Level for case C ($\lambda = 4, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

motif is listed among the top scoring motifs, as can be seen in Tab. 5.5 and each component scores higher than the planted motif due to the contribution of spurious occurrences.

Case E

This case is more interesting because the motifs planted in cases B, C and D have all been planted in this data set. Furthermore, each of these motifs was planted in much less input sequences. Fig. 5.12 summarizes the results for case E. These plots are unsurprisingly much different from the ones shown in the previous cases, showing a peak around score level 30 for the average bicluster volume plot.

Tab. 5.6 shows a portion of the list of motifs inferred from the identified biclusters. The top scoring motifs are the result of the contribution of components of the planted complex motifs

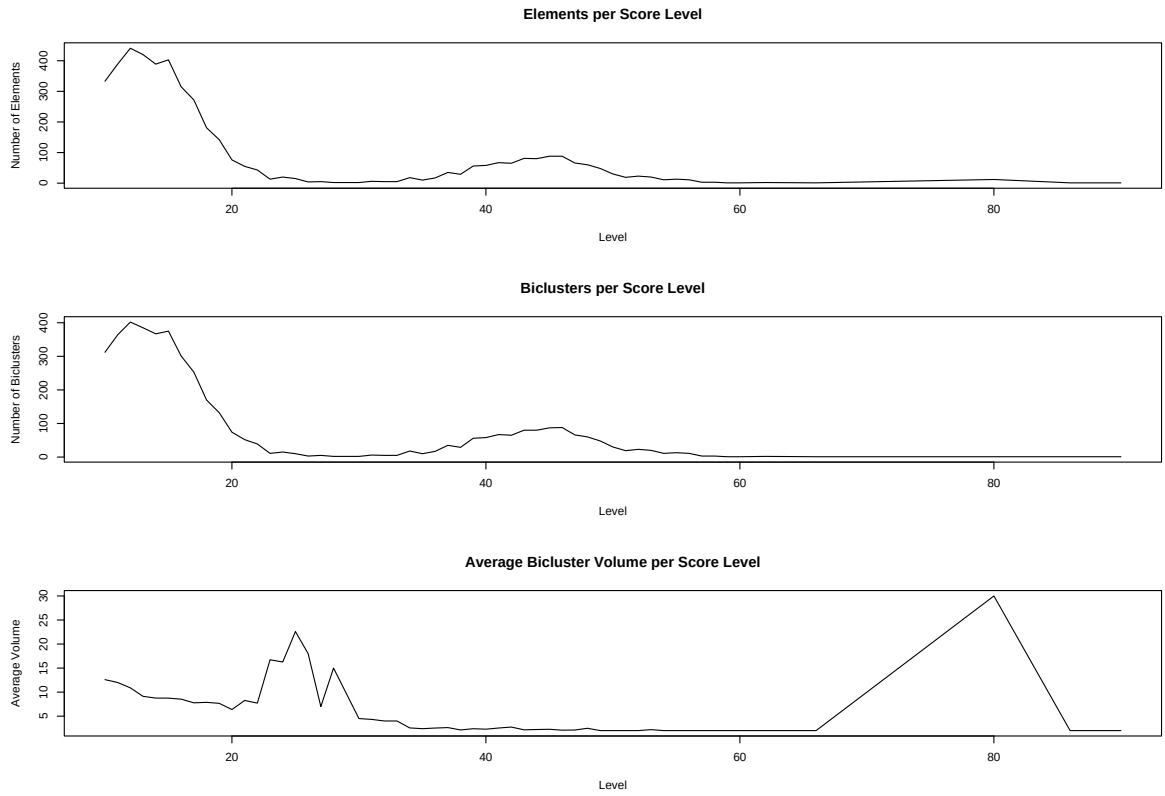

Figure 5.11: Number of Elements, Number of Biclusters and Average Bicluster Volume per Score Level for case D ($\lambda = 4, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

#	**Motif**	**Score**
1	AAAAT	90
2	CCCCT	89
3	TTTTA	86
4	AAAATn_5TTTTAn_5CCCCT	80
5	ACCCC	66
⋮	⋮	⋮
4296		

Table 5.5: Top scoring motifs inferred from the top scoring biclusters for case D

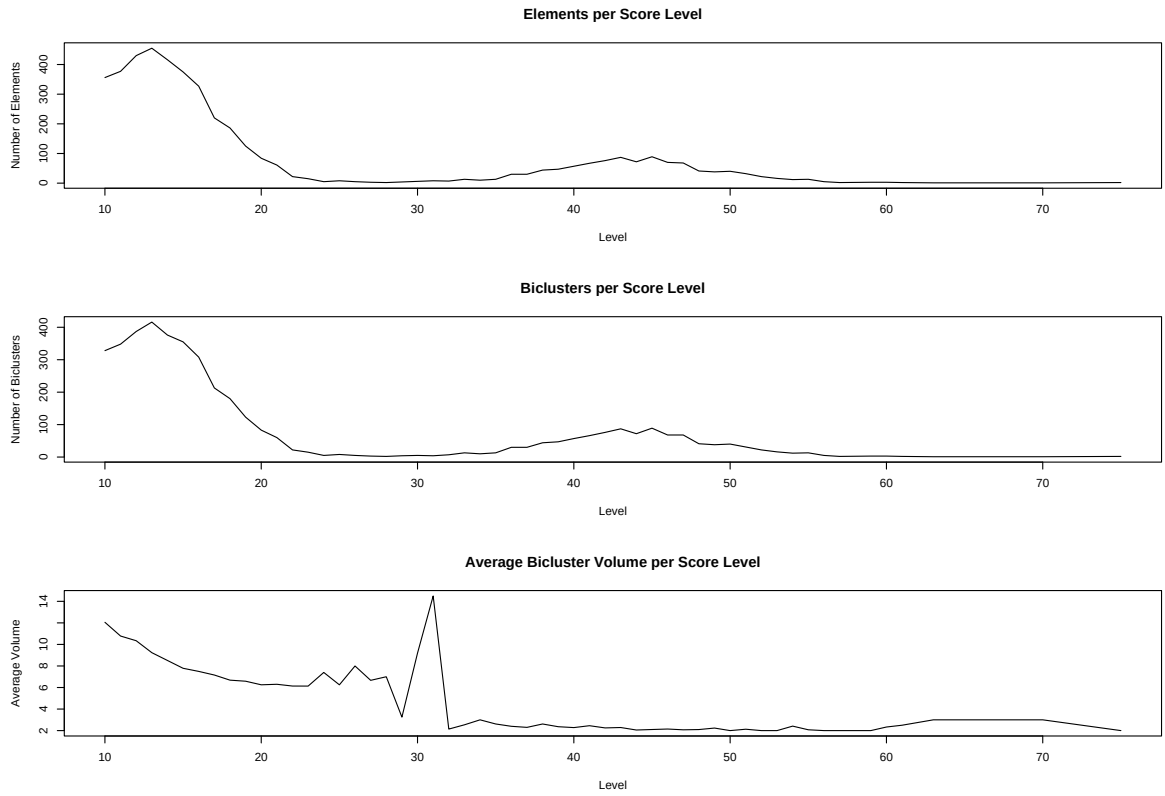

Figure 5.12: Number of Elements, Number of Biclusters and Average Bicluster Volume per Score Level for case E ($\lambda = 4, \varepsilon = 0, |\mathcal{S}| = 100, |S_i| = 600$)

with spurious occurrences, not unlike we have seen in previous cases. What is interesting to observe are the results concerning the planted motifs. The simple motif that was planted (**AAAAA**) was not recovered in isolation. Instead, it appears merged with spurious occurrences of other configurations. This is not surprising since we planted the simple motif in only a few sequences (40), much less than the score level below which many spurious occurrences of 5-mers start to be common, which is around score level 55, as shown in Fig. 5.6.

Motifs listed in ranks 1010 through 1015 were the only complex motifs recovered by our method. They all resemble the planted complex motifs but none matches any of them exactly. This is, once again, not surprising. If we inspect motif in rank 1015 it is almost an exact superposition of the two complex motifs that were planted in the input sequences. All the other motifs are simply subsets of the configurations induced by our complex motifs. The

#	**Motif**	**Score**
1	AAAAT	75
2	TTTTA	75
3	AAAAAT	70
4	AAAAAC	66
5	CAAAA	63
⋮	⋮	⋮
1010	AAATn_5TTTT	32
1011	AAAATn_5TTTTn_6CCCC	31
1012	AAAAn_6TTTTn_6CCCCT	31
1013	AAAATn_5TTTTAn_{11}TTTA	31
1014	AAAAn_6TTTTAn_6CCCTn_1TTTA	31
1015	AAAATn_5TTTTAn_5CCCCTn_1TTTA	30
⋮	⋮	⋮
4260		

Table 5.6: Top scoring motifs inferred from the top scoring biclusters for case E

explanation for this result is straightforward. The two complex motifs that were planted in the data set were quite similar and induced almost identical configurations, except for those referring to the pairs (AAAA, TTTT), (AAAA, TTTA), (AAAT, TTTT) and (AAAT, TTTA) which differed in their relative distances. A spurious occurrence of one of these configurations in one of the input sequences was sufficient to deprive the other of its status of most common configuration. In this case, configurations (AAAA, TTTT, 24), (TTTT, AAAA, -24), (AAAT, TTTT, 23) and (TTTT, AAAT, -23) were the ones which got discarded making it impossible to fully recover the planted motif AAAATn_{20}TTTTA. On the other hand, the other complex motif was fully recovered. The motif that is listed in rank 1015 was inferred by inspection of the identified bicluster and what is shown does not convey all the information that the bicluster offers.

It is true that many configurations that pertain to motif AAAATn_{20}TTTTA are interwined with those generated by AAAATn_5TTTTAn_5CCCCT but it is still possible to discriminate between the two. We decided to present the reconstructed motif in this way because both configurations

(AAAA, TTTA, 10) and (AAAA, TTTT, 25) are associated with the bicluster, but, for instance, there is only a pair of configurations involving 4-mers TTTA and CCCC which are the ones induced by AAAATn_5TTTTAn_5CCCCT. This allows us to suspect that different signals have combined in this bicluster and we can, in this case, easily distinguish the two.

This case has illustrated some of the more complex situations that we can face using the proposed method. It is, however, a very artificial situation because the planted motifs are quite similar. These situations are unlikely to occur with real data sets if one chooses an appropriate value for λ.

5.2 Biological Data

As we said previously, synthetic data sets are convenient for controlled tests but they fall short of being a reliable model of regulatory regions. In this section we present the results of the application of our method to a real data set [57]. This data set is composed of various σ^{54}-dependent promoter sequences of *Escherichia coli*. The data set is composed of 69 sequences with an average length of 580 nucleotides.

We begin by analyzing the plots obtained for the number of elements, biclusters and average bicluster volume per score level by applying our method using $\lambda = 4$ and $\varepsilon = 0$ shown in Fig. 5.13. These plots are significantly different from the ones obtained for the synthetic data. There are no distinguishable local maxima which suggests that these sequences are not random as we have already argued.

In Tab. 5.7 we list a portion of the list of motifs assembled by inspection of the identified biclusters. The complex motifs that are listed consist of the only complex motifs identified above score level 20. In [57] a consensus sequence for the promoter of these sequences was obtained by a combination of genetic evidence and putative promoters reported in the literature based on sequence similarity. The consensus sequence reported by the authors was: NNNNmrNrYTGGCACGNNNNTTGCWNNwNNNNN where R stands for purines, Y for pyrimidines, W for A or T and, as usual, N stands for any nucleotide.

The complex motifs obtained using our method are in absolute accordance with the consensus sequences reported by the authors.

As we discussed in previous chapters, our method is not guaranteed to find all interesting motifs due to its heuristic nature and is also vulnerable to reporting false positives. It should,

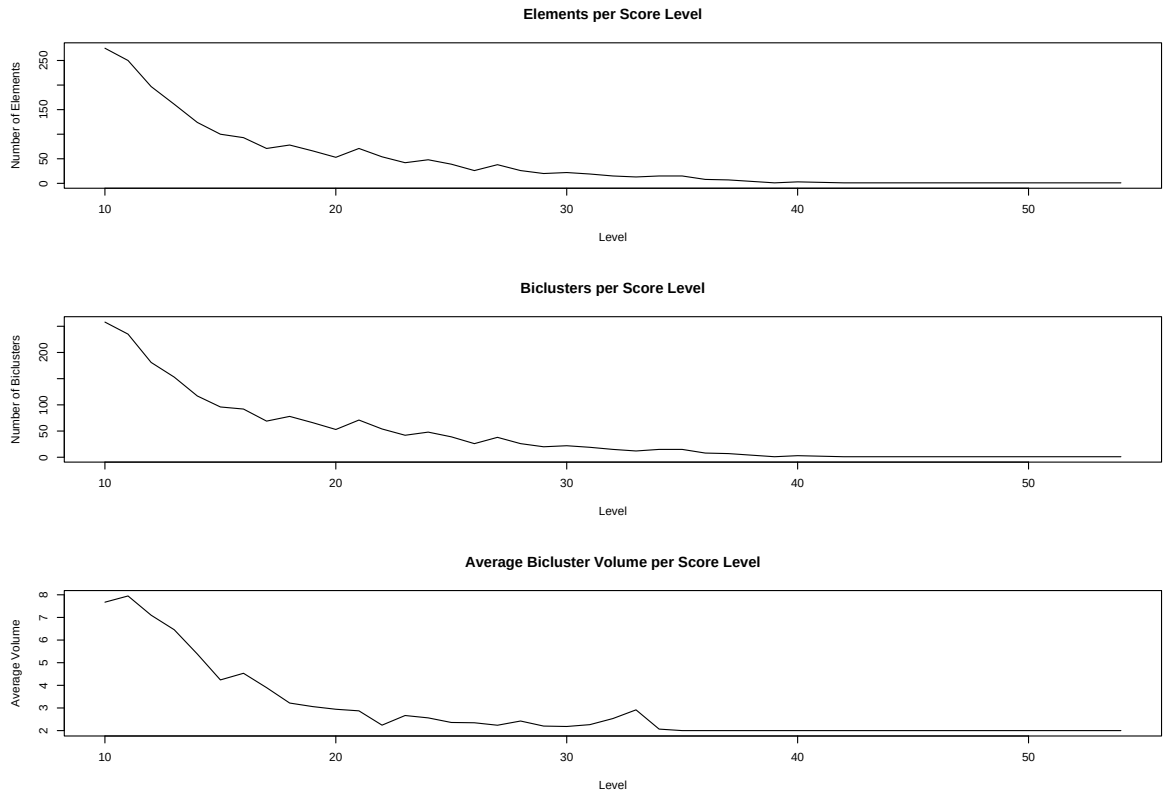

Figure 5.13: Number of Elements, Number of Biclusters and Average Bicluster Volume per Score Level for the σ^{54} data set ($\lambda = 4, \varepsilon = 0, |\mathcal{S}| = 69, average|S_i| = 580$)

therefore, rely on a motif finder to validate its output. The biclusters identified in this case suggest the existence of a complex motif with two components separated by a gap of 5-7 nucleotides. If we use the SMILE algorithm [5] and ask for all complex motifs with a first component between 4 and 6 nucleotides long and a second component between 4 and 5 nucleotides long separated by a gap between 5 and 7 nucleotides long occurring in at least 20 sequences the algorithm reports one motif only:

`TGGC_TTGCT`

This suggests that our method is artificially inflating the scores of the biclusters because it does not perform a cross-check of the sequences in which each configuration occurs. If we lower the number of sequences we require the motif to occur in we recover all the motifs reported by our method (not shown).

#	**Motif**	**Score**
1	CTGGC	54
2	TGGCA	53
3	GGCAC	49
4	TTGCT	42
5	CCCGC	40
⋮	⋮	⋮
40	TGGCn_7TTGC	35
⋮	⋮	⋮
125	TGGCAn_6TTGC	30
⋮	⋮	⋮
270	TGGCAn_6TTGCT	25
⋮	⋮	⋮
537	GGCACn_5TTGC	20
⋮	⋮	⋮
1887		

Table 5.7: Top scoring motifs inferred from the top scoring biclusters for the σ^{54} data set

Other algorithms, like MEME [23], are, in principle, able to find interesting features in data sets. We have used MEME to analyze this data set. A common way to present results reported by MEME is a multi-level consensus sequence. The nucleotides in the top row are the most likely for each position, and nucleotides at lower rows are decreasingly probable. The multi-level consensus sequences obtained with MEME for the σ^{54} data set was the following:

```
G  C  T  G  G  C  A  C  G  G  C  T  C  T  T  G  C  T
T  T                T  A  C  G  C  G  C           A
```

The multi-level consensus sequence reported by MEME is in accordance with both the documented consensus sequence and the motifs extracted using the method proposed in this thesis. It is not so clear, however, in elucidating the complex structure of the motif which is the main goal of our approach.

Chapter 6

Conclusions and Future Work

In this thesis we have presented an effective method to guide modern combinatorial motif finders. We have seen that, up to a point, our method is capable of reporting the relevant motifs by itself. We have demonstrated this ability in the results presented in chapter 5.

We have already discussed the shortcomings of our approach which refer to situations where the method is unable to detect all the important configurations necessary to assemble a motif present in the input sequences. We argue that most of these situations are unlikely to occur in practice and that this method stands as a useful practical tool to guide the search for motifs, especially those exhibiting a complex nature.

Notwithstanding the fact that most of the biclusters identified by our method suggest the presence of motifs which can easily be assembled by inspection of the configurations associated with the bicluster elements, we still lack an automated procedure to assemble these motifs. This is a task that can be challenging in some special cases where multiple motifs end up grouped in the same bicluster or when some configurations are masked due to signal interference. It is, however, an interesting and useful procedure that deserves attention.

Another issue is the ability to accurately predict the exact number of input sequences where the configurations grouped in a bicluster simultaneously occur. As we have discussed in previous chapters the score level at which a bicluster is identified is an upper bound of the simultaneous occurrences of all associated configurations. Reporting the exact number of sequences involves keeping track of which input sequences each configuration occurs in. This improvement would also be useful to detect and discard spurious biclusters that result from grouping configurations that never occur in the same input sequences.

Another important improvement to our method would be offering support for degenerate motifs, i.e., the ability to cope with nucleotide substitutions as is the case with many modern motif finders. This is not a trivial matter and involves, besides redefining most of the concepts introduced in chapter 4, considering not only co-occurrences of pairs of λ-mers present in the input sequences but also those of λ-mers at some pre-defined Hamming distance of the former.

We have mentioned in chapter 4 that the number of biclusters that our biclustering algorithm is able to identify ($|L(\mathcal{S})|^2$) is much less than the theoretical maximum number of biclusters ($3^{\frac{|L(\mathcal{S})|}{3}}$). This is due to the fact that the algorithm performs a greedy search, missing an undetermined part of the solution. We have argued that this can be mitigated by the fact that, in practice, the generated matrix of co-occurrences is sparse at each level considered. It would be interesting, however, to compare the results obtained using this algorithm with another performing an exhaustive search. It may prove worthwhile to implement a randomized version of the algorithm whereby rows/columns would be randomly targeted for addition, in conjunction with a beam search approach which would combine, at each level, the biclusters obtained by considering initial biclusters in different orderings.

Finally, it is clear from the results presented in chapter 5 that many of the identified biclusters are due to random occurrences of short simple motifs. A user of our method would greatly benefit from a statistical significance assessment of the reported biclusters. Despite the fact that listing the biclusters by decreasing score values already constitutes a significant help in discriminating the output it is insufficient to isolate interesting motifs occurring in few input sequences.

Bibliography

[1] J. van Helden, A. F. Rios, and J. Collado-Vides. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. *Nucleic Acids Research*, 28(8):1808–1818, 2000.

[2] P. A. Pevzner and S. H. Sze. Combinatorial approaches to finding subtle signals in DNA sequences. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 8:269–278, 2000.

[3] E. Eskin and P. A. Pevzner. Finding composite regulatory patterns in DNA sequences. *Bioinformatics*, 18(1):354–363, 2002.

[4] E. Eskin, U. Keich, M. S. Gelfand, and P. A. Pevzner. Genome-wide analysis of bacterial promoter regions. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 29–40, 2003.

[5] L. Marsan and M.-F. Sagot. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. *Journal of Computational Biology*, 7(3/4):345–360, 2000.

[6] A. M. Carvalho, A. T. Freitas, A. L. Oliveira, and M.-F. Sagot. A highly scalable algorithm for the extraction of cis-regulatory regions. In *Proceedings of the 3rd Asia Pacific Bioinformatics Conference*, volume 1 of *Advances in Bioinformatics and Computational Biology*, pages 273–282. Imperial College Press, 2005.

[7] J. Watson and F. Crick. A structure for deoxyribose nucleic acid. *Nature*, 171:737–738, 1953.

[8] G. M. Cooper and R. E. Hausman. *The Cell: A Molecular Approach, Third Edition.* Sinauer Associates, Inc., 2003.

[9] M. Ptashne and A. Gann. *Genes and Signals.* Cold Spring Harbor Laboratory Press, 2001.

[10] M. Blanchette. Algorithms for phylogenetic footprinting. In *Proceedings of RECOMB*, pages 49–58, 2001.

[11] T. Werner. Models for prediction and recognition of eukaryotic promoters. *Mammalian Genome*, 10(2):168–175, 1999.

[12] A. Vanet, L. Marsan, and M.-F. Sagot. Promoter sequences and algorithmical methods for identifying them. *Research in Microbiology*, 150:779–799, 1999.

[13] G. D. Stormo, T. D. Schneider, L. Gold, and A. Ehrenfeucht. Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli. *Nucleic Acids Research*, 10(9):2997–3011, 1982.

[14] R. Harr, M. Häggström, and P. Gustafsson. Search algorithm for pattern match analysis of nucleic acid sequences. *Nucleic Acids Research*, 11(9):2943–2957, 1983.

[15] M. E. Mulligan, D. K. Hawley, R. Entriken, and W. R. McClure. Escherichia coli promoter sequences predict in vitro RNA polymerase selectivity. *Nucleic Acids Research*, 12(1 Pt 2):789–800, 1984.

[16] R. Staden. Computer methods to locate signals in nucleic acid sequences. *Nucleic Acids Research*, 12(1 Pt 2):505–519, 1984.

[17] T. D. Schneider, G. D. Stormo, J. S. Haemer, and L. Gold. Information content of binding sites on nucleotide sequences. *Journal of Molecular Biology*, (188):415–431, 1986.

[18] J. M. Heumann, A. S. Lapedes, and G. D. Stormo. Neural networks for determining protein specificity and multiple alignment of binding sites. *Proceeding of the International Conference on Intelligent Systems for Molecular Biology*, 2:188–194, 1994.

[19] G. Z. Hertz and G. D. Stormo. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15(7/8):563–577, 1999.

[20] O. D. King and F. P. Roth. A non-parametric model for transcription factor binding sites. *Nucleic Acids Research*, 31(19), 2003.

[21] E. Segal and R. Sharan. A discriminative model for identifying spatial cis-regulatory modules. *Journal of Computational Biology*, 12(6):822–834, 2005.

[22] J. Buhler and M. Tompa. Finding motifs using random projections. *Journal of Computational Biology*, 9(2):225–242, 2002.

[23] T. L. Bailey and C. Elkan. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning*, 21(1-2):51–80, 1995.

[24] T. Bailey and C. Elkan. 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*, pages 28–36, 1994.

[25] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: A Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, 1993.

[26] A. V. Favorov, M. S. Gelfand, A. V. Gerasimova, D. A. Ravcheev, A. A. Mironov, and V. J. Makeev. A Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length. *Bioinformatics*, 21(10):2240–2245, 2005.

[27] J. Schug and G. C. Overton. Modeling transcription factor binding sites with Gibbs Sampling and Minimum Description Length encoding. *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 5:268–271, 1997.

[28] U. Keich and P. A. Pevzner. Finding motifs in the twilight zone. In *Proceedings of RECOMB*, 2002.

[29] A. Price, S. Ramabhadran, and P. A. Pevzner. Finding subtle motifs by branching from sample strings. *Bioinformatics*, 19 Suppl 2:149–149, 2003.

[30] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[31] D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, 1999.

[32] Kandel, Matias, Unger, and Winkler. Shuffling biological sequences. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 71:171–185, 1996.

[33] S. F. Altschul and B. W. Erickson. Significance of nucleotide sequence alignments: a method for random sequence permutation that preserves dinucleotide and codon usage. *Molecular Biology and Evolution*, 2(6):526–538, 1985.

[34] E. Coward. Shufflet: shuffling sequences while conserving the k-let counts. *Bioinformatics*, 15(12):1058–1059, 1999.

[35] S. C. Madeira and A. L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[36] R. Peeters. The maximum edge biclique problem is NP-Complete. *Discrete Applied Math.*, 131(3):651–654, 2003.

[37] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12(1):61–86, 2002.

[38] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. *Journal of Computational Biology*, 10(3-4):373–384, 2003.

[39] J. A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association (JASA)*, 67(337):123–129, 1972.

[40] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences USA*, 97(22):12079–12084, 2000.

[41] C. Tang, L. Zhang, I. Zhang, and R. Ramanathan. Interrelated two-way clustering: an unsupervised approach for gene expression data analysis. In *Proceeding of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering*, pages 41–48, 2001.

[42] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: capturing subspace correlation in large data sets. In *Proceedings of the 18th IEEE International Conference on Data Engineering*, pages 517–528, 2002.

[43] J. Yang, W. Wang, H. Wang, and P. Yu. Enhanced biclustering on expression data. In *Proceeding of the 3rd IEEE International Symposium on Bioinformatics and Bioengineering*, pages 321–327, 2003.

[44] T. M. Murali and S. Kasif. Extracting conserved gene expression motifs from gene expression data. *Proceedings of the Pacific Symposium on Biocomputing*, pages 77–88, 2003.

[45] A. Tanay, R. Sharan, and R. Shamir. Discovering statistically significant biclusters in gene expression data. *Bioinformatics*, 18 Suppl 1:136–144, 2002.

[46] J. Liu and W. Wang. Op-cluster: Clustering by tendency in high dimensional spaces. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 187–194, 2003.

[47] H. Wang, W. Wang, and P. S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, pages 394–405, 2002.

[48] S. Busygin, G. Jacobsen, and E. Kramer. Double conjugated clustering applied to leukemia microarray data. In *Proceedings of the 2nd SIAM International Conference on Data Mining, Workshop Clustering High Dimensional Data*, 2002.

[49] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13(4):703–716, 2003.

[50] A. Califano, G. Stolovitzky, and Y. Tu. Analysis of gene expression microarrays for phenotype classification. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, number 8, pages 75–85, 2000.

[51] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue coclustering of gene expression data. In *Proceedings of the 4th SIAM International Conference on Data Mining*, 2004.

[52] D. Duffy and A. Quiroz. A permutation based algorithm for block clustering. *J. Classification*, 8:65–91, 1991.

[53] Y. Cheng and G. M. Church. Biclustering of expression data. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, volume 8, pages 93–103, 2000.

[54] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17 Suppl 1:243–252, 2001.

[55] Q. Sheng, Y. Moreau, and B. De Moor. Biclustering microarray data by Gibbs sampling. *Bioinformatics*, 19 Suppl 2:196–196, 2003.

[56] J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23–28, 1965.

[57] H. Barrios, B. Valderrama, and E. Morett. Compilation and analysis of σ^{54}-dependent promoter sequences. *Nucleic Acids Research*, 27(22):4305–4313, 1999.

[58] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2002.