Algorithms for Representation and Discovery of Transcription Factor Binding Sites

Elena Zaslavsky

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
By the Department of
Computer Science

January 2006

Abstract

A major objective in molecular biology is to understand how a genome encodes the information that specifies when and where a gene will be transcribed into its protein product. Mediating proteins, known as transcription factors, facilitate this process by interacting with the cell's DNA and the transcription machinery. It is of central importance to identify all sequence-specific DNA binding sites of transcription factors. In this thesis, we consider two relevant computational problems.

The first problem is to develop a representation for a group of known binding sites of a particular transcription factor, in order to facilitate recognition of other binding sites of the same protein. We evaluate the effectiveness of several approaches commonly used for this problem, and show that there are statistically significant differences in their performance. We also consider variants of the basic methods that incorporate pairwise nucleotide dependencies and per-position information content. We find that the use of per-position information content improves all basic methods, and that including local pairwise nucleotide dependencies within binding site models results in better performance for some approaches.

The second problem is that of motif discovery. In this context, given a set of sequences known to contain binding sites of a particular transcription factor, the objective is to identify their locations. We propose a novel combinatorial optimization framework for motif finding, which utilizes both graph pruning techniques and an integer linear programming formulation. Additionally, we introduce a procedure to identify statistically significant motifs. We apply our algorithm to numerous biological datasets as well as to synthetic data, and it performs exceptionally well. Furthermore, we show our framework to be versatile and easily applicable to other variants of the DNA binding site identification problem such as phylogenetic footprinting, the 'subtle' motif formulation and the multiple motifs problem.

Studying the optimization framework in greater depth, we introduce a novel, more compact integer linear program that utilizes the discrete nature of the distance metric imposed on pairs of subsequences. We compare the properties of the two alternate formulations from a theoretical perspective and demonstrate that the compact formulation also leads to a method that is highly effective in practice.

Acknowledgments

I would like to express my sincere appreciation to the many people who have helped me make this thesis a reality. The first order of gratitude goes to my advisor, Professor Mona Singh for her patient guidance, unfailing support and encouragement. It has been a pleasure to work with such a bright and understanding person, and an invaluable experience to learn from her. I would like to thank Professors Bernard Chazelle, Sridhar Hannenhalli, Brian Kernighan and Robert Schapire for taking the time out of their busy schedules to serve as members on my thesis committee, and especially Professors Bernard Chazelle and Sridhar Hannenhalli for reviewing this manuscript and providing insightful comments.

I am grateful to my co-authors, Carl Kingsford and Robert Osada for complementing my abilities and skills, and allowing me to publish our joint work as part of this thesis. My thanks go to prior and current members of the Singh computational biology group, Eric Banks, Jessica Fong, Carl Kingsford, Elena Nabieva and Robert Osada for providing a friendly and intellectually challenging atmosphere. A particular note of appreciation to Jessica Fong, Elena Nabieva and especially Carl Kingsford for critiquing my manuscripts and providing other research related advice.

I am very grateful to my husband, Dima Zaslavsky, for his immeasurable love and patience, his intellectual and emotional support, as well as technical expertise that allowed me to stay sane and focused on my PhD. A most meaningful and wonderful experience during this time has been the birth of our daughter, Racheli, who has opened new dimensions of joy and happiness for me. Her smile makes every difficulty I've encountered on this road a triviality.

My deep appreciation goes to my parents, Vladimir and Dora Oransky, for their boundless love and unwavering belief in me; to R' Dovid and Hindy Sitnick for becoming my second set of parents and always being an invaluable presence in my life. A special thanks to a dear friend, Natalie Zelenko, for lending a listening ear in the

challenging moments[1].

The research conducted for this dissertation was made possible with the funding of Defense Advanced Research Projects Agency (DARPA), grant MDA972-00-1-0031, and Princeton University.

[1]Many things I mention here may sound cliche, but I truly, sincerely mean them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Biological Background

Molecular biology has entered the so-called post genomic era, when scores of genomes have been sequenced, and biological data abounds; yet, many fundamental biological processes are not well understood. For example, how does as simple an organism as the single-celled prokaryote *Escherichia coli* know how to interact with its environment, as each environmental cue generates a specific response, with specific proteins and reactions. Another related question, pertinent for multicellular organisms is this: what is it that can make two cells of the same life form, which possess the same genetic material, so vastly different?

One of the processes, responsible for transforming the static DNA blueprint into a dynamic response and adapting an organism to life's demands, is that of *gene expression*, which selectively switches genes on and off. Only a subset of genes in any cell is active at any given time in the developmental stage of the organism, the metabolic or physiologic state of the cell and under any given set of environmental conditions. Gene expression or the ability of a gene to produce an active protein product is highly regulated. Whereas controls that act in eucaryotic gene expression are very complex and can occur in every stage in the process of transforming a gene into its final product, prokaryotic gene regulation happens mostly during *transcription*, the stage of *messenger RNA (mRNA)* synthesis [Alberts *et al.* 2002]. Transcription is typically facilitated by special purpose proteins, called transcription factors, which carry out their function by binding DNA fragments in the immediate vicinity of the gene being regulated [Alberts *et al.* 2002]. Identifying such DNA binding sites is a very important problem in its own right, as it serves as a first and necessary step in understanding gene regulation. In this thesis we focus on computational techniques for transcription factor binding site discovery. While the methods we develop are broadly applicable, we have experimented primarily on the prokaryotic *Escherichia coli* genome.

In more detail, transcriptional regulation in prokaryotes occurs mainly during transcription initiation, when the transcription enabling complex, RNA polymerase, binds the double-stranded DNA at its *promoter* region. The rate of transcription initiation is modulated by the interaction of transcription factors with the RNA polymerase. Transcription factors can provide *positive* regulation (activation or enhancement) by improving the ability of the RNA polymerase to bind and initiate transcription, or negative regulation (repression) by interfering with the function of the RNA polymerase [Alberts *et al.* 2002]. For instance, a repressor protein may bind DNA and block access of the RNA polymerase to its promoter. Transcription factors bind DNA sequence elements called *operators* or *regulatory binding sites* when forming protein-DNA complexes. The majority of binding sites are located in close proximity to the transcription start site, upstream of it relative to the direction of transcription. The region of DNA containing these sites is called the regulatory region, and it is typically no longer than 1000 nucleotides for prokaryotes, and is often much shorter [Alberts *et al.* 2002].

Regulatory binding sites are usually very specific to a particular protein, and are similar to one another at the sequence level. When binding sites for a single transcription factor protein are aligned together, patterns are readily evident, with conserved and less conserved regions (see Figure 1.1); we call these patterns *motifs*. Since transcription factors typically regulate a number of different genes, collectively referred to as the transcription factor's regulon, a binding signature or motif can be discerned among them. Motif *instances*, the actual binding sites we would like to identify in the context of regulation, correspond to conserved sequence elements, matching to the motif pattern, in the regulatory regions of the protein's regulon. The sites are usually short (up to 30 nucleotides with some exceptions) and often gapless, and instances of a motif are of the same length. Although similar to each other at the sequence level, motif instances do differ in composition to achieve varying degrees of affinity in protein-DNA interactions. Such differences account for better control of gene expression.

As biological approaches to identifying transcription factor binding sites are time-consuming and costly, computational methods are needed to address this very important problem. In finding transcription factor binding sites we can identify two subproblems, those of binding site *representation* and *discovery* [Stormo 2000], both of which we address in this thesis. The first problem is to develop a representation for a group of known binding sites of a particular transcription factor; this involves extracting essential features in order to facilitate recognition of additional binding sites of the same protein. The second problem, motif discovery, is to find hitherto unidentified locations of binding sites (or motif instances) in a given set of sequences known by some (often experimental) means to contain sites for a common factor.

1.2 Representation and Search Problem

In Chapter 2, we address the motif representation problem. The challenge lies in finding a suitable way to represent a set of known binding sites with the goal of

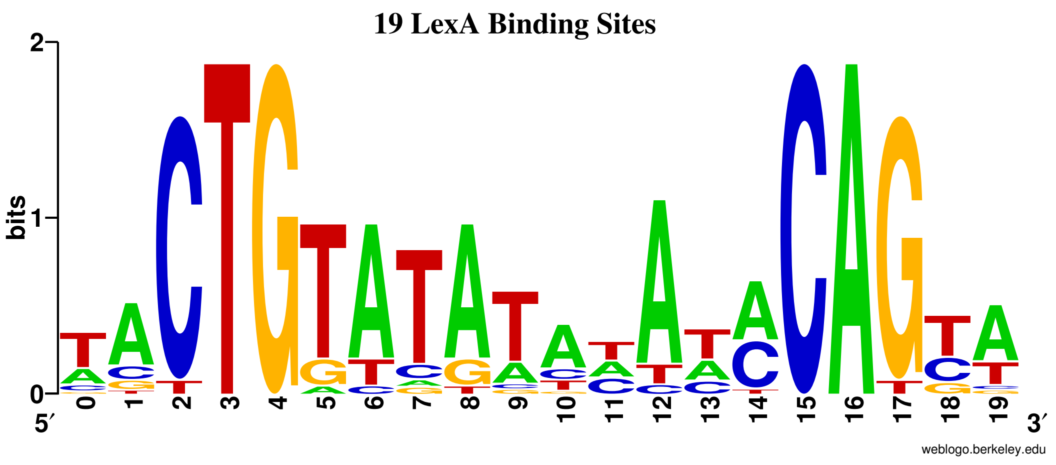


Figure 1.1: Sequence Logo [Schneider and Stephens 1990] of 19 transcription factor LexA binding sites in *E. coli*, created using weblogo.berkeley.edu. The motif is 20 bases long. Overall height of stack shows level of conservation at position (e.g., positions 2–4 and 15–17 are very well conserved). Height of each residue indicates relative frequency and, by assumption, the probability of observing it in the motif.

searching genomic data and discovering novel binding sites of the same transcription factor. Several approaches are commonly used for representing a transcription factor's binding sites, including *consensus* sequences (e.g., [Day and McMorris 1992]) and probabilistic approaches [Staden 1984, Berg and von Hippel 1987], such as *position-specific scoring matrices* (PSSM). A consensus sequence of a group of aligned binding sites is one that contains the most frequently occurring residue in each column, and consensus-based methods compute the degree of similarity between the site in question and the consensus sequence. Probabilistic approaches (e.g., PSSM), which assess the likelihood of observing a given base in a given position of the binding site, lead to most common representations. For instance, the widely-used sequence logo representation [Schneider and Stephens 1990], indicates both the sequence conservation in a position of a group of aligned sequences and the relative frequency of each amino or nucleic acid at that position (see Figure 1.1).

1.2.1 Our contributions: a comparative analysis of representation and search methods

In Chapter 2, we evaluate the effectiveness of several methods for motif representation and search. In addition to consensus sequences and PSSM, we consider methods that compute the average number of nucleotide matches between a putative site and all known sites. Moreover, whereas all above-mentioned methods assume independence between column positions, we extend these basic approaches by incorporating pairwise nucleotide dependencies [Bulyk *et al.* 2002] in our models. Additionally, we explore the effectiveness of integrating per-position information content [Schneider and Stephens 1990] directly in our scoring schemes.

In cross-validation testing on a data set of *E. coli* transcription factors and their binding sites, we show that there are statistically significant differences in how well various methods identify transcription factor binding sites. The use of per-position information content improves the performance of all basic approaches. Furthermore, including local pairwise nucleotide dependencies within binding site models results in improved performance for approaches based on nucleotide matches. Based on our analysis, the best results when searching for DNA binding sites of a particular transcription factor are obtained by methods that incorporate both information content and local pairwise correlations. Software enabling such analysis for specific transcription factors or other genomes is available for download at `http://compbio.cs.princeton.edu/bindsites`.

1.3 Motif Discovery Problem

In Chapters 3 and 4, we address the motif discovery problem: that is, the problem of finding mutually similar patterns in unaligned sequence data. The objective is to identify possible motif instances and their locations in the sequences from a given data set. Motif finding is an important and long-studied problem in computational molecular biology, with applications to both DNA and protein sequences, as short common subsequences in the data may correspond to functionally important elements. In the context of transcriptional networks, motif finding applications arise when identifying shared regulatory signals such as transcription factor binding sites. For protein sequences, motif finding can serve as a tool to identify shared functional and structural elements.

For DNA data, motif finding algorithms have typically been applied to sets of sequences from a single genome that have been identified as possessing a common motif. One source of such data is made available through DNA microarray studies. In this setting the normalized gene expression levels of many genes are determined under a number of experimental conditions in different phases of the cell cycle. These data are then clustered to reveal similar patterns in expression. Under the assumption that co-expressed genes are likely co-regulated, the upstream regions of such co-expressed genes can be subjected to motif finding [Tavazoie *et al.* 1999, Spellman *et al.* 1998]. Another source of data are chromatin immunoprecipitation (ChIP-chip) experiments [Lee *et al.* 2002] and protein binding microarrays [Mukherjee *et al.* 2004]. In these latter approaches the binding of a regulatory protein to DNA is recognized directly via molecular methods. The group of DNA sequences, to which the protein was bound, can be input to a motif finding algorithm to identify the binding sites precisely.

An orthogonal approach, which attempts to identify regulatory sites among a set of orthologous genes across genomes of varying phylogenetic distance, is adopted by [McGuire *et al.* 2000, McCue *et al.* 2001, Blanchette and Tompa 2002, Kellis *et al.* 2003, Cliften *et al.* 2003]. Here it is assumed that the functionally important DNA binding sites are conserved throughout evolution.

There are several variants [Keich and Pevzner 2002] of the motif finding problem, and we address most of them in Chapter 3: (i) the simple sample, where each sequence in the dataset contains exactly one motif instance; (ii) the invaded sample, where more than one instance may exist in some sequences; (iii) the corrupted sample, where a motif instance may not appear in every sequence; (iv) the multiple patterns, where the sequences may contain more than a single common motif.

1.3.1 Previous approaches to motif finding

Numerous approaches to motif finding have been suggested (e.g., [Lawrence and Reilly 1990, Lawrence *et al.* 1993, Bailey and Elkan 1995, Brazma *et al.* 1998, Rigoutsos and Floratos 1998, Hertz and Stormo 1999, Tompa 1999, Hughes *et al.* 2000, Marsan and Sagot 2000, van Helden *et al.* 2000, Workman and Stormo 2000, Pevzner and Sze 2000, Liu *et al.* 2001, Eskin and Pevzner 2002, Buhler and Tompa 2002, Sinha and Tompa 2003, Pavesi *et al.* 2004, Frith *et al.* 2004]). The biological problems addressed by motif finding are complex and varied, and no single currently existing method can solve them completely (e.g., [Tompa *et al.* 2005]).

Motif finding algorithms are based on the representation method chosen for a group of binding sites. Approaches based on the probabilistic representations of binding sites make the assumption that the column distributions of the motif are most different from the distribution of background sequence in which the motif instances are embedded. Accordingly, they attempt to maximize the likelihood ratio of the motif model to the background model.

Consensus representation based methods take an alternate view of motif finding, in which a motif corresponds to a pattern or a regular expression. While exhaustive enumeration of patterns is only feasible for small motif lengths [Tompa 1999, van Helden *et al.* 2000, Sinha and Tompa 2003, Pavesi *et al.* 2004], data sample-driven approximate approaches [Rigoutsos and Floratos 1998, Pevzner and Sze 2000, Buhler and Tompa 2002, Sze *et al.* 2004] have been developed for general lengths.

Comparative tests [Tompa *et al.* 2005] have shown that the two approaches based on different representation of binding sites seem to be complementary, with no definite advantage of either one. Some problem instances are solved correctly by representing motifs with their consensus, while others by using position specific scoring matrices.

1.3.2 Our contributions: a combinatorial optimization approach

Considering a pattern-based approach, in Chapter 3 we introduce a versatile combinatorial optimization framework for the motif finding problem, where the goal is to find a minimum (or maximum) weighted clique in an N-partite graph. Our approach couples graph pruning techniques with a novel integer linear programming formulation. Our method is flexible and robust enough to accommodate several variants of the motif finding problem, including finding both protein motifs and DNA motifs, either in co-regulated upstream region data or in evolutionarily related sequences of varying phylogenetic distance. We further extend our method to discover multiple motifs. In contrast to commonly-used stochastic search methods for the problem, our combinatorial approach yields optimal solutions in most cases. We apply our method to numerous biological sequence datasets, as well as to synthetic data, and in all cases it performs very well, identifying either known motifs or motifs of high conservation. We assess statistical significance of the discovered motifs, and find that in the vast majority of cases such a motif is unlikely to have arisen by chance alone.

In Chapter 4, we consider the integer linear programming formulation for the motif finding problem from a more theoretical perspective. While our previous approach focused on graph pruning and decomposition techniques to reduce the size of the combinatorial problem, here we describe an alternate novel integer linear programming formulation for motif finding. Its effectiveness is based on a key observation that the edge weights in the graph formulation belong to a small set of possibilities due to the discrete nature of the distance metric imposed on pairs of subsequences. We explore the relative advantages and disadvantages of the two integer linear programming formulations and show that our novel formulation leads to an algorithm that is highly effective in practice when applied to problem instances arising from biological sequence data. We are able to solve moderate-sized problems to optimality often many times faster than our earlier mathematical programming approach.

Thesis organization. The remainder of the thesis is organized as follows. In Chapter 2, we discuss the motif representation problem, and present a comparative study of various methods and their extensions. In Chapters 3 and 4, we focus on the motif discovery problem. We first introduce our combinatorial optimization approach in Chapter 3, and describe the basic mathematical programming formulation and graph pruning techniques. In Chapter 4, we introduce an alternative mathematical programming formulation and evaluate its effectiveness. In Chapter 5, we draw conclusions pertaining to this work and present directions for future research.

Chapter 2

Representing and searching for transcription factor binding sites

2.1 Introduction

In this chapter we address the question of how best to represent a group of binding sites for a particular transcription factor with the goal of searching for additional occurrences of such sites in genomic data. At its essence, this task involves extracting essential features from a set of binding sites sequences, and then using these features to search for additional sites. However, since a single transcription factor can bind sites of considerable variability, it is difficult to find a precise set of rules for identification of novel binding sites; as a result, a number of different methods have been proposed for this problem (e.g., [Staden 1984, Schneider *et al.* 1986, Berg and von Hippel 1987, Day and McMorris 1992, Gelfand 1995, Stormo 2000]). Traditionally, a particular transcription factor's preference for binding site composition has been represented by a consensus sequence (e.g., [Day and McMorris 1992]), and more recently as a sequence logo [Schneider and Stephens 1990]. Novel sites for a transcription factor are typically found by either matching to a consensus sequence [Day and McMorris 1992], or using position-specific scoring matrices (PSSMs) [Staden 1984].

While many methods for identification of regulatory binding sites have been proposed, the availability of online data sets of transcription factors and their aligned binding domains (e.g., [Robison *et al.* 1998, Salgado *et al.* 2004]) allows us to quantify the effectiveness of different approaches. In particular, cross-validation testing can be used to quantify how well each method performs in distinguishing between the DNA binding sites for a particular transcription factor and those of other proteins. While there may be some overlap between the binding domains for different transcription factors, the known DNA binding sites for the transcription factor under consideration should be among the top-ranked sites. Such an empirical evaluation is important and timely, as whole-genome scans in search of the binding sites of a particular protein are increasingly used to make functional annotations of uncharacterized proteins, and to infer properties of transcriptional regulatory networks (e.g., [Thieffry *et al.* 1998]). Additionally, the aforementioned methods are the basis for other more sophisticated approaches for predicting transcription factor binding sites, including motif discovery and cross-genomic approaches (e.g., [Hertz and Stormo 1999, Hughes *et al.* 2000, Sinha and Tompa 2000, Gelfand *et al.* 2000, McGuire *et al.* 2000, McCue *et al.* 2001, Tan *et al.* 2001, Blanchette and Tompa 2002]).

In this chapter, we evaluate four basic methods for representing and searching for transcription factor binding sites: consensus sequences [Day and McMorris 1992], two variants of position specific scoring matrices (log-odds matrices, and the statistical mechanics based Berg & von Hippel method [Berg and von Hippel 1987]), as well as a novel method based on nucleotide matches, which we call *Centroid*, that computes the average number of nucleotide matches between a putative site and all known binding sites. In addition, we consider whether these basic methods can be improved using two natural extensions: incorporation of pairwise nucleotide dependencies and per-position information content. Whereas the basic methods assume that each base contributes independently to binding, it has been demonstrated that there are interdependent effects between bases [Man and Stormo 2001, Bulyk *et al.* 2002]. Though the independence assumption has clearly been useful in practice and seems to provide a good approximation to the energetics of DNA-protein binding [Benos *et al.* 2002], here we assess whether improvement is possible by incorporating pairwise correlations. Similarly, the use of per-position information content has already proven to be useful in representing binding sites [Schneider and Stephens 1990] and in motif discovery [Hertz and Stormo 1999]; here, we apply it directly to the problem of searching for a transcription factor's binding sites. In particular, we consider the heuristic of using the information content of a position to weight its contribution towards the overall score.

We compare how well these methods and their extensions perform in identifying the binding sites for a particular transcription factor without erroneously identifying binding sites of other proteins. We assess improvement in performance using the Wilcoxon matched-pairs signed-ranks test, which evaluates whether the frequency with which one method outperforms another is statistically significant, as well as receiver operating characteristic (ROC) curves, which compare the performance of two or more methods over a range of possible false positive rates. Testing on a data set of *E. coli* transcription factor binding sites [Robison *et al.* 1998], our analysis shows that there are statistically significant differences between these methods. In particular, our main findings are:

1. The extension of using per-position information content to weight positional scores improves the performance of all methods, sometimes dramatically. For example, consensus sequences have by far the poorest performance of all basic methods in discriminating between binding sites for the transcription factor of interest and binding sites of other transcription factors; however, weighting each match to a consensus base by the appropriate per-position information content

makes consensus sequences much more competitive with other methods.

2. Methods based on nucleotide matches, such as consensus sequences and Centroid, show statistically significant improvements when incorporating pairwise nucleotide dependencies. Furthermore, in these cases, the choice of which pairs to include in the model is important; in particular, considering all possible pairs of nucleotides in a binding site is not as effective as using just neighboring pairs. On the other hand, probabilistic methods, such as log-odds PSSMs, do not show statistically significant improvements when incorporating pairwise dependencies.

3. The difference in performance between methods decreases substantially once information content and pairwise correlations have been incorporated.

In general, when searching for DNA binding sites of a particular transcription factor, we find that methods incorporating information content and pairwise correlations are most effective. Of course, for a specific transcription factor and its binding sites, an alternate method may perform better in practice. For organisms like *E. coli* with many well-characterized transcription factors and binding sites, analysis similar to the one performed here should aid in choosing a specific method and suitable threshold for a particular transcription factor. Our software enabling such analysis is available for download at `http://compbio.cs.princeton.edu/bindsites`.

2.2 Methods

2.2.1 Data set

The data of [Robison *et al.* 1998,McGuire *et al.* 2000] contains 68 regulatory proteins and their aligned DNA binding sites; we constructed our data set from it as follows. First, only proteins with at least four binding sites were considered. Second, in the original database, occasionally the binding sites for a single regulatory protein were split into multiple groups based on the number of tandem duplications; we chose to include the individual sites for ArgR, MetJ, and PhoB rather than their tandem-repeated counterparts. Third, binding sites from sigma factors were removed, as were binding sites from NarP, since all the latter are also binding sites for NarL. Fourth, duplicate binding sites were removed in order to preserve the integrity of the leave-one-out cross-validation process. Finally, each binding site was located within the *E. coli* K-12 genome (version M54 of strain MG1655 [Blattner *et al.* 1997]), and was extracted along with flanking regions on each side. Binding sites that could not be located unambiguously within the genome were excluded from our study. This process left 35 transcription factors and 410 binding sites, with an average of 11.7 ± 8.5 sites per transcription factor.

2.2.2 Approaches for representing and searching for transcription factor binding sites

Four basic approaches for representing and searching for transcription factor binding sites were evaluated. We found that specific implementation details affect performance, and so each of the methods is described briefly below.

First, a word about notation. Let S be the set of N DNA binding sites for a particular transcription factor. We assume that each binding site has length l and that these binding sites are aligned. Define $n_j(b)$ to be the number of times base b appears in the j-th position of any sequence in S, and $f_j(b)$ to be the corresponding frequency. Similarly, define $n(b)$ to be the number of times base b appears overall in the N binding sites, and $f(b)$ to be the overall frequency for base b. We then consider how each method scores a new DNA subsequence t (also of length l) in an attempt to predict whether t is a binding site of the given protein. Let t_j denote the j-th base of the sequence t to be scored.

Extending the above notation to pairs of positions, let $n_{ij}(b, d)$ be the number of times the ordered pair of bases (b, d) occurs in positions i and j of any sequence belonging to S, and $f_{ij}(b, d)$ be the corresponding frequency. Ideally, a method for incorporating pairwise correlations should only take into account those pairs that are known, perhaps through structural studies, to act together in determining DNA-protein binding specificity. Since such precise information is not always readily available, as a first approximation we focus on considering pairwise correlations between bases that are nearby in sequence, and introduce the notion of *scope* to delimit which pairs are considered correlated. For instance, a scope of one restricts correlated positions to adjacent pairs while a scope of two considers both adjacent pairs and pairs separated by an intermediate base.

Next we define the information content (**IC**) of a position in a set of sequences. Information content is an important concept based on the information theoretic notion of entropy introduced in a seminal paper by Claude Shannon [Shannon 1948]. In the current application, the entropy of a position expresses the number of bits necessary to describe the position in a binding site, and the information content of a position is calculated by subtracting its entropy from the value of the maximum possible entropy. That is, the higher the information content, the more conserved (and presumably more important) the position. More specifically, the information content IC_j of position j in S is defined as $2+\sum_{b\in DNA} f_j(b)\log f_j(b)$, and the information content IC_{ij} of a pair of positions is $4+\sum_{b,d\in DNA} f_{ij}(b,d)\log f_{ij}(b,d)$ [Schneider *et al.* 1986]. Information content is used in Sequence Logos by [Schneider and Stephens 1990] mainly as a visualization tool to identify important positions in a binding site. We propose a different, more direct usage in a scoring scheme, namely by incorporating the IC of a position as a multiplicative factor in scoring a target binding site sequence.

We consider the following methods and their variations (summarized precisely in Table 2.1).

Consensus. These methods vary considerably [Day and McMorris 1992]; we implemented the version of consensus sequences described by [Yamauchi 1991]. For each position j, let b be the most frequent base and d be the second most frequent base. If $f_j(b) > 0.5$, then b is our consensus base for position j (denoted by consensus_j); otherwise if $f_j(b) + f_j(d) > 0.75$ then both b and d are the consensus bases. If neither is true, there is no consensus base for this position. The score of a new sequence t is obtained by counting the number of times t_j agrees with the consensus base for the j-th position.

PSSM. Typically, this method assumes independence between positions, and computes a log-odds score for a potential binding site. We employ a commonly used Bayesian estimate to handle the zero frequency case and replace $f_j(b)$ by $\hat{f}_j(b) = \frac{n_j(b)+\hat{f}(b)}{N+1}$ [Lawrence *et al.* 1993], where $\hat{f}(b)$ is the estimate of overall background frequency of base b, computed as $\frac{n(b)+.25}{N\times l+1}$.

Berg & von Hippel. We conducted the full analysis with a statistical mechanics-based method that makes the connection between base-pair statistics of a set of sites and its binding free energy. Denoting the number of occurrences of the most common base in position j of the set of binding sites by $n_j(0)$, the method scores a new sequence t by computing a per-positional corrected log-odds score of observing a base of t versus the most frequent base in the corresponding position of the sequences [Berg and von Hippel 1987, Berg and von Hippel 1988].

Centroid method. We introduce a method that scores a sequence t by computing the average shared identity between t and every sequence in S.

Next we consider extensions of the above methods that incorporate pairwise dependencies.

Consensus-P. For a sequence t, this method counts both the number of nucleotides matching the consensus sequence and the number of nucleotide pairs within a partic-

Method	Score	Pair Score ($j = i + s$)
Consensus	$\sum_{j=1}^{l} [t_j \in \text{consensus}_j]$	$\sum_{s=1}^{\text{scope}} \sum_{i=1}^{l-s} [t_i \in \text{consensus}_i, t_j \in \text{consensus}_j]$
PSSM	$\sum_{j=1}^{l} \log \frac{\hat{f}_j(t_j)}{\hat{f}(t_j)}$	$\sum_{s=1}^{\text{scope}} \sum_{i=1}^{l-s} \log \frac{\hat{f}_{ij}(t_i,t_j)}{\hat{f}(t_i)\hat{f}(t_j)}$
Berg & von Hippel	$\sum_{j=1}^{l} \log \frac{n_j(t_j)+0.5}{n_j(0)+0.5}$	$\sum_{s=1}^{\text{scope}} \sum_{i=1}^{l-s} \log \frac{n_{ij}(t_i,t_j)+0.5}{n_{ij}(0,0)+0.5}$
Centroid	$\sum_{j=1}^{l} f_j(t_j)$	$\sum_{s=1}^{\text{scope}} \sum_{i=1}^{l-s} f_{ij}(t_i, t_j)$

Table 2.1: Scores computed by various representation methods. The final score for a pair method is the sum of the basic score and the pair score.

ular scope matching the corresponding bases in the consensus sequence.

Centroid-P. This method considers the number of shared bases as well as the number of shared pairs of bases within a particular scope between the sequence t and each sequence in S.

PSSM-P. This method is an extension of the PSSM log-odds method that also accounts for pairwise correlations. [Zhang and Marr 1993] note that although rigorously generalizing PSSM-P beyond adjacent pairs is not difficult in principle, in practice the small number of known sites per transcription factor limits the rigorous probabilistic derivation of the method to only consider adjacent pairs. For example, a derivation of scope two requires calculating triplet frequencies. Instead, in our analysis we evaluate an intuitive definition of the method that considers only pairwise dependencies regardless of scope[1]. We handle the zero frequency case by replacing $f_{ij}(b, d)$ by $\hat{f}_{ij}(b, d) = \frac{n_{ij}(b,d)+\hat{f}(b)\hat{f}(d)}{N+1}$. We tried several different ways of computing the reference "background" pair frequencies; modeling this as the product of single column frequencies had the best overall performance.

We extend the **Berg & von Hippel** method to incorporate pairs of bases in a similar manner, with $n_{ij}(0, 0)$ giving the most frequent pair of bases in positions i and j.

Finally, for every method considered, we also examine its variation in which per-

[1]For scope value of one, the rigorous derivation that assumes that position j depends on position $j + 1$ subtracts single columns log-odds scores; the method described above performs better in our tests without subtracting these singlet scores.

position information content is used to weight the contribution of each position (or pair of positions) towards the overall score[2]. For instance, the score computed by Centroid IC is $\sum_{j=1}^{l} \mathrm{IC}_j \times f_j(t_j)$, and the score computed by its pair counterpart Centroid-P IC with scope parameter *scope* is the sum of the Centroid IC score and $\sum_{s=1}^{\text{scope}} \sum_{i=1}^{l-s} \mathrm{IC}_{ij} \times f_{ij}(t_i, t_j)$, where $j = i + s$.

2.2.3 Cross-validation testing and analysis

The most common usage of any of the methods described above would be to scan non-coding regions in a genome in order to find binding sites for a particular transcription factor. This would entail scoring consecutive windows of appropriate length and considering windows that score above a carefully chosen threshold to be predicted binding sites. However, such a framework is not easily applicable when we wish to evaluate and compare different methods; in particular, the *E. coli* genome contains many yet uncharacterized binding sites, and predicted windows may correspond to true binding sites even if they are not annotated as such in our data set. Instead, we consider a testing framework with carefully pruned sets of positive and negative examples. In particular, we conduct leave-one-out cross-validation studies to evaluate a particular method, and consider each binding site s in turn as follows. Suppose s belongs to a set S of known binding sites, each of length l, for a particular transcription factor TF. The method under consideration then uses all the sites except s, i.e. $S - \{s\}$ to build the binding site representation for TF, and scores s as well as a set of negative examples. The negative examples consist of all binding sites in our data set except those known to be bound by TF. To score a negative binding site t, we examine all possible alignment positions of this binding site against the binding site representation of TF such that either the representation of TF is completely

[2]In [Schneider *et al.* 1986] it is suggested to use a sampling error correction based on the expected information content of n random samples. This correction did not improve performance in our tests and so we only report on uncorrected information content.

contained within t, or t is completely contained within the representation of TF. In the latter case, genomic flanking regions around t are used for scoring. Six pairs of binding sites were found to reside completely inside one another in the genome. In these cases, when scoring the negative binding site, a true binding site for the transcription factor of interest is present; thus these corresponding binding sites were removed from the pool of negative examples during cross-validation testing. The final score for a target sequence is taken to be the higher score when considering both the original sequence and its reverse complement. Of course, it still is possible that transcription factor TF can bind some of the negative examples, but nevertheless s should be among the top scoring sites in the overall pool.

The discriminatory power of each method, exhibited in the relative score of the actual binding site among all scored sites, is analyzed using two data-mining tests: averaged ranks and receiver operating characteristic (ROC) curves (e.g., see [Witten and Frank 2000]). In particular, for each site s of a transcription factor under consideration we compute its rank in cross-validation testing by counting how many negative examples score as well or better than s, with lower rank indicating better performance. Then, to compare how well two methods perform, we use a Wilcoxon matched-pairs signed-ranks test. Briefly, the number of times one method outperforms the other is compared with how many times such an event would happen merely by chance under the assumption that both methods perform equally well. P-values of less than .05 are considered significant. For ROC analysis, we first create a ROC curve for each individual leave-one-out test (i.e., we keep track of whether the binding site was found as a function of the number of false positives allowed) and then average over all sites for that transcription factor. Individual ROC curves for each transcription factor are then further averaged across the various transcription factors to arrive at a final curve for the method.

2.3 Experimental Results

2.3.1 Comparison of basic methods

We first establish the baseline performance of each of the four basic methods described above. Figure 2.1 compares the performance of the Consensus, PSSM, Berg & von Hippel and Centroid methods using ROC analysis. Each curve plots the fraction of correctly classified positive examples (TP rate) as a function of the incorrectly classified negative examples (FP rate).

As expected, Consensus performs markedly the poorest, consistently lying to the lower-right of the other curves. On the other hand, the rest of the methods are comparable, as their curves lie very close to one another and cross at various FP rates.

As evidenced above, and in all of the following testing scenarios, PSSM and its variants perform virtually identically to Berg & von Hippel's method and its variants. Therefore, we describe our results omitting the latter method so as not to overcrowd the graphs.

2.3.2 Influence of pairwise correlations

We next address the question of whether inclusion of pairwise correlation information improves the ability of the basic approaches to identify transcription factor binding sites. Ideally, a method for incorporating pairwise correlations should only take into account those pairs of binding site positions that are known, perhaps through structural studies, to act together in determining DNA-protein binding specificity. Such precise biological information is not readily available, and so, as a first approximation, we focus on considering pairwise correlations between bases that are nearby in sequence. In particular, we compare the results of incorporating pairwise correlations as the positional distance allowed between pairs of bases (i.e., the scope) is varied.

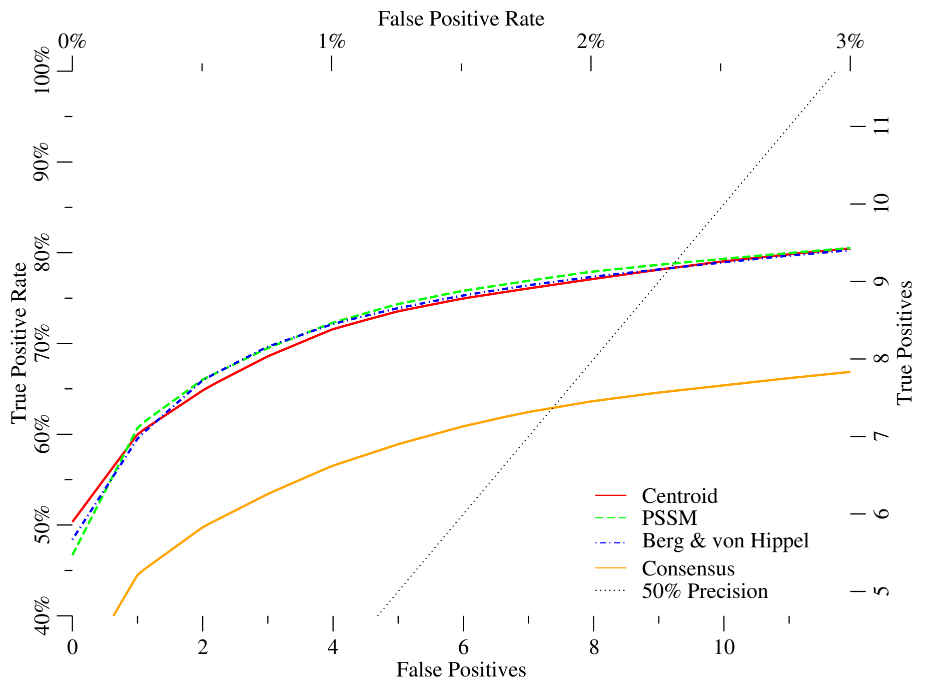
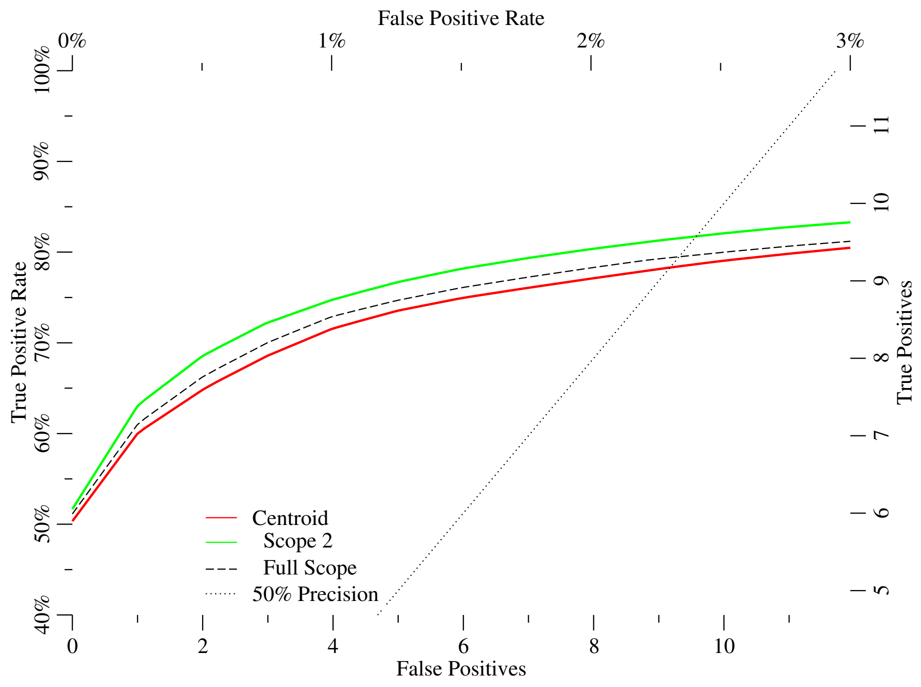
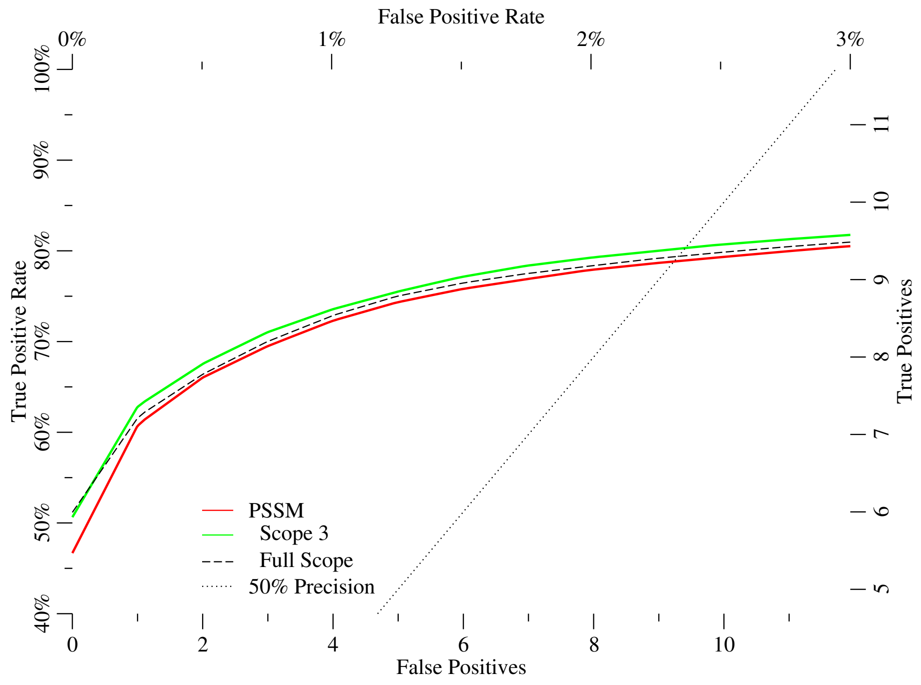


Figure 2.1: ROC curves comparing performance of the four basic methods: Centroid, PSSM, Berg & von Hippel, and Consensus. The top and left scales give the false positive rate and true positive rate, respectively, whereas the bottom and right scales give the corresponding number of binding sites. The 50% precision line indicates the boundary at which the methods would predict as binding sites as many incorrect sites as correct ones. Consensus is clearly outperformed by the other three methods.

Figure 2.2 summarizes the effect of considering various pairwise correlations for centroid and PSSM. We consider scope parameters between zero (where no pairwise dependencies are assumed) and four, as well as full scope. For Centroid-P, neither zero scope nor full scope performs best, whereas curves with scopes in the range of two to four consistently achieve higher TP rates across the relevant range of FP rates (data only shown for scope of two). The improvement for scope two is modest (3% improvement when allowing no false positives, and approximately 5% when allowing one false positive) yet significant with a p-value of less than 10^{-4}, as judged by the Wilcoxon matched-pairs signed-ranks test (see **Statistical significance of methods comparison**). Thus, incorporating pairwise correlations improves the discriminatory ability of the centroid method; however, it is important to consider pairs of positions only within a limited scope. In the remainder of the chapter, Centroid-P is used with



(a) Centroid



(b) PSSM

Figure 2.2: ROC curves comparing performance when pairs are considered for Centroid (a) and PSSM (b). For each, the basic method (scope zero) is shown, along with the method using all possible pairs (full scope) and the method using the best performing scope (scope two for Centroid-P and scope three for PSSM-P).

scope two as it is less computationally intensive than those with scopes three and four and yet performance is comparable. A similar trend is observed for Consensus (data not shown), where a dramatic improvement in performance occurs with the addition of pairwise correlations. At scope two, used for all subsequent analysis, performance increases over scope zero by 25% when allowing no false positives, and 26% when allowing one false positive. In contrast, for PSSM-P, incorporating pairs at small scopes results in performance decline (see **Discussion**). At larger scopes PSSM-P performs very similarly to PSSM. For the remaining analysis, we consider PSSM-P at scope three; performance increases over scope zero by 8% with no false positives and by 3% with one false positive. However, this improvement is not judged to be statistically significant (see below). To summarize, for the methods based on tallying up nucleotide matches, such as Centroid and Consensus, considering pairwise correlations clearly helps; however, we cannot make the same claim for the probabilistic methods such as PSSM.

We further quantify the effect of nearby pairwise correlations by considering the performance of the Centroid-P method in the same cross-validation scenario but on a perturbed data set, produced by randomly shuffling the columns of the binding sites used as positive examples. While shuffling the columns of a set of binding sites preserves per-column nucleotide composition, it also, on average, destroys any local pairwise correlations found in the original alignment. The results are shown in Figure 2.3 where a ROC curve for Centroid-P tested on the original data set is plotted against the same method tested on the shuffled data set. Shuffling and cross-validation are averaged over 1,024 trials producing the solid shuffling curve, whereas the dashed curves show performance out to one standard deviation due to effects of randomness. The benefit of incorporating nearby inter-column correlations is clearly observed, as performance on shuffled sites is consistently worse than performance on the original sites.

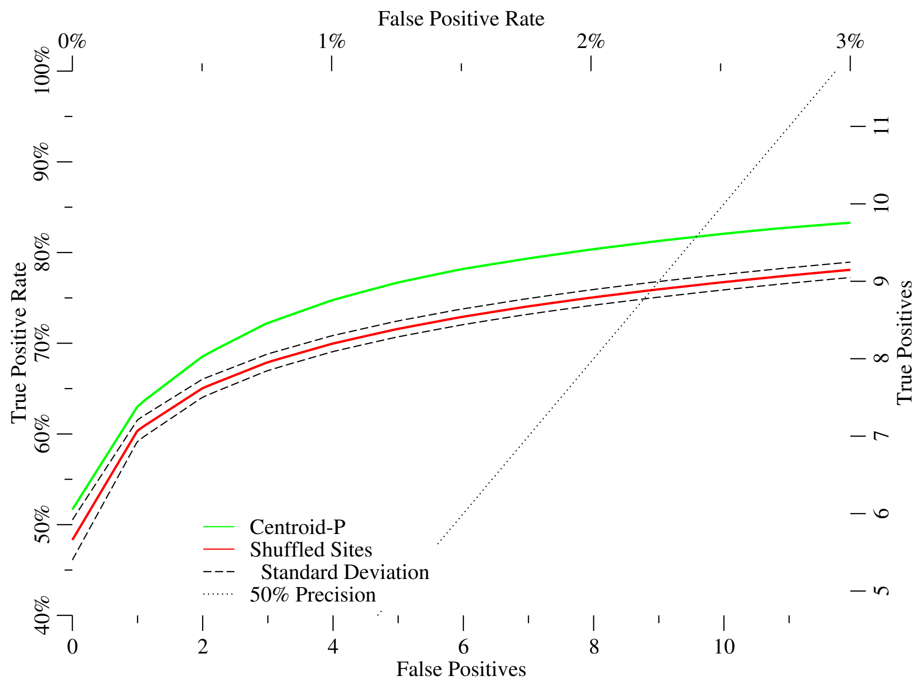


Figure 2.3: ROC curves comparing Centroid-P with scope two using regular sites and sites with columns shuffled. The solid curve is an average over 1,024 different shuffles, while dashed curves show performance out to one standard deviation with random shuffling.

2.3.3 Importance of per-position information content

Next, we evaluate the performance of the basic methods described above and consider the effect of adding per-position information content to each method. We compare via a ranks chart the performance of the Consensus, PSSM and Centroid methods, along with their pair counterparts with and without information content.

Figure 2.4 shows average ranks (as computed over the binding sites for each transcription factor) for both versions of each method and its pair extension side by side. Comparing median performance, it is clear that adding per-position information content results in improved performance in both the original and pairwise versions of the basic methods. Noticeably, the addition of information content to the Consensus method dramatically improves its performance, and in fact makes it much more competitive with the other methods. When considering performance differences upon incorporating both pairwise dependencies and per-position information content at

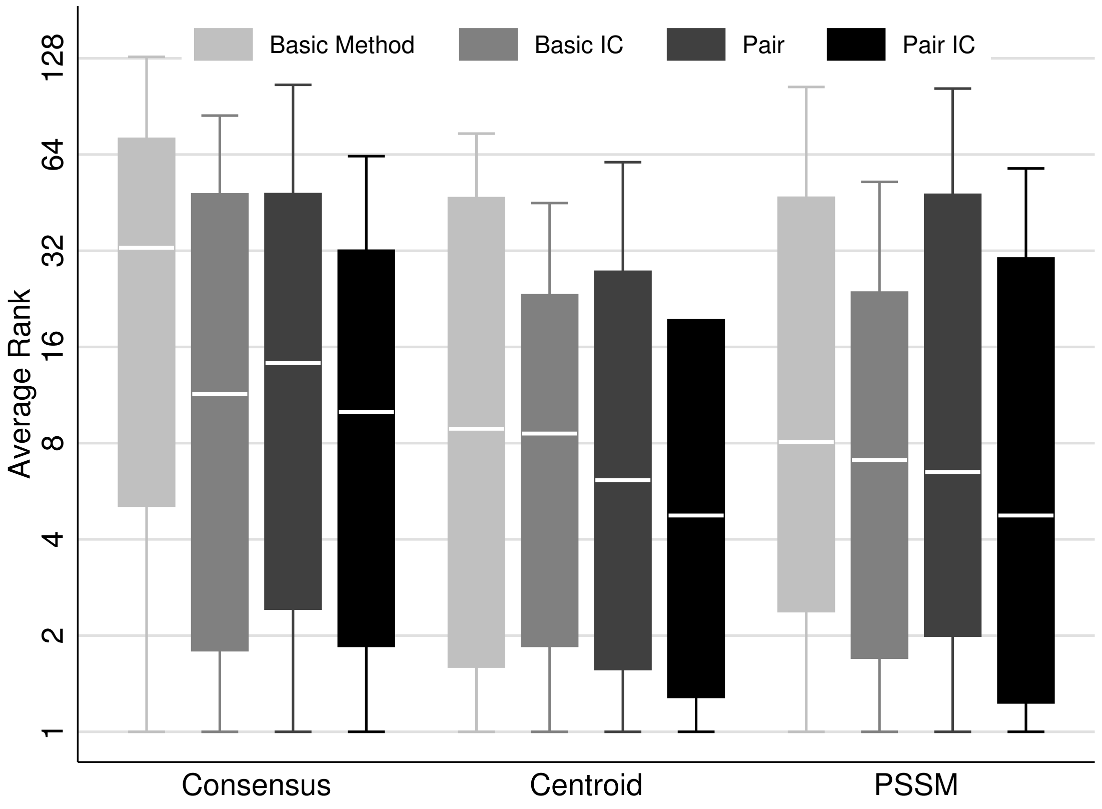


Figure 2.4: Performance of all the methods measured by averaged ranks per transcription factor. For each transcription factor, an average rank is computed from the rank of each of its binding sites in cross-validation testing. The horizontal line in each box is the median transcription factor's average rank while each box shows the 25th–75th percentiles for average ranks.

particular values of false positives, basic Consensus shows a 36% improvement when allowing no false positives and a 37% improvement when allowing one false positive. These values for Centroid are 2% and 9%, and for PSSM are 10.5% and 8%.

2.3.4 Statistical significance of method comparisons

To compare methods and assess whether the differences in performance (partially described above) of various methods are statistically significant, we apply a Wilcoxon matched-pairs signed-ranks test. For every comparison and each cross-validation test, we calculate the change in the rank of the left-out-example. These rank differences are converted into p-values under the assumption that both methods perform equally

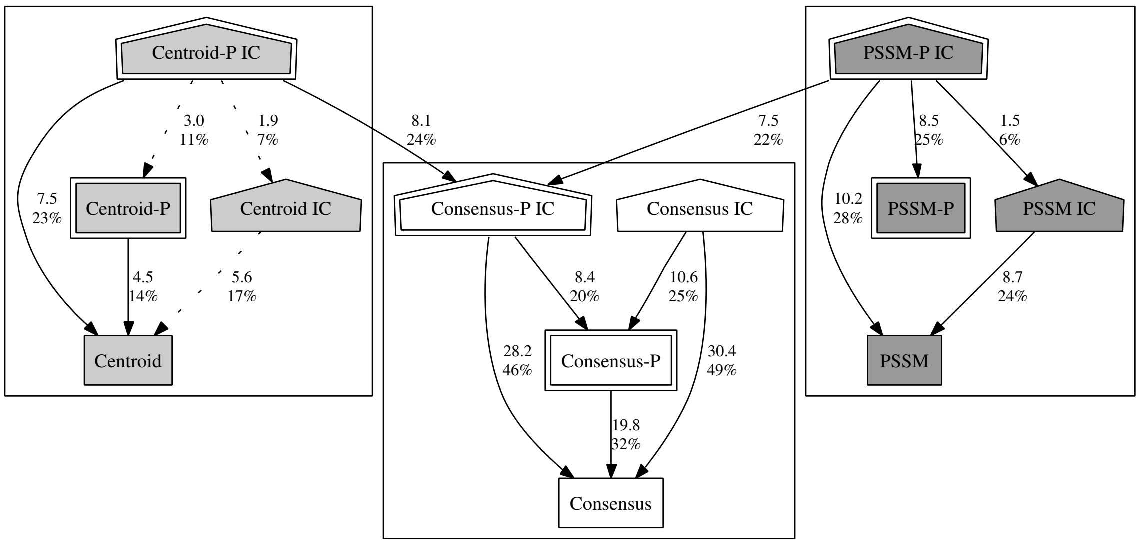


Figure 2.5: Partial ordering (at the 95% significance level) of methods based on a signed ranks test. Arrows point towards worse performing methods, with labels indicating the difference in average rank between the methods, both as an absolute number and as percentage improvement. Pairwise comparisons were performed between all four variations of each method, as well as between the three P IC methods, for a total of 21 tested hypotheses. P-values were adjusted for multiple hypothesis testing using a sequentially rejective Bonferroni test [Holm 1979]. Dotted edges show differences that were not found to be significant with the Bonferroni correction, but have individual p-values $< .05$.

well, and a sequentially rejective Bonferroni test is used to select the most significantly different pairs of methods so that the overall p-value is less than 5% [Holm 1979].

We chose to test a subset of all possible pairs of methods with the goal of identifying the best performing methods and quantifying improvement (if any) resulting from incorporation of information content and pairwise dependencies. In particular, for each basic method, all its variations are compared to each other; additionally, the versions of every method that incorporate both pairs and information content are compared to each other. The results are shown in Figure 2.5, producing a graph in which a directed edge connects a pair of methods, one with a significant performance improvement over the other (arrow pointing toward worse performing methods). The overall conclusion is that the pair and information content incorporating version of each basic method outperforms the other methods in its group (with the exception of Consensus-P IC which did not perform significantly better than Consensus-IC).

As for the overall best method, Centroid-P IC has the best average rank; and both Centroid-P IC and PSSM-P IC statistically outperform the highest number of other methods. All information content weighted methods perform significantly better than their non-weighted counterparts, with a qualification that although Centroid-P IC and Centroid IC perform better than Centroid-P and Centroid at individual p-values of .02, these differences are not statistically significant with the Bonferroni correction. The improvement resulting from adding pairwise dependencies is more modest, as we observe only some of the possible arrows relating a method and its pair-incorporating counterpart.

2.4 Discussion

Based on our findings, we conclude that both pairwise correlations and especially information content can be used to improve the discriminatory power of computational methods for binding site recognition and prediction.

The importance of pairwise correlations for binding site identification has been a topic of debate within the computational biology research community, and numerous papers have been published supporting both viewpoints (e.g., [Bulyk *et al.* 2002, Benos *et al.* 2002, Barash *et al.* 2003]). Our findings show that considering pairwise correlations improves performance for all three methods, though the improvements are very modest and not significant for the statistical methods (e.g., see PSSM-P in Figure 5). Moreover, while the resulting improvement is dramatic for Consensus-P, it is more modest for Centroid-P. Nevertheless, our testing suggests that inter-positional information can provide additional binding domain specificity, particularly when the appropriate pairs are considered. This is demonstrated by the fact that considering some pairs of positions (i.e., those within close sequential proximity) results in improved performance for the nucleotide match methods such as Centroid-P;

it may be possible that more careful selection of pairwise dependencies, perhaps from crystal structures when available, would result in further improvements. We suspect that performance of the statistical methods, such as PSSM-P, do not show statistically significant improvements when pairwise correlations are considered because of the non-occurrence of many base-pair combinations in the small data set; such a case is more severely penalized in PSSM-P scoring than in Centroid-P or Consensus-P scoring.

It is not clear a priori that the addition of information content should improve performance of methods that already incorporate frequency information. Nevertheless, we found that the inclusion of information content benefits all methods tested, and a clear trend is observed when considering the methods and their IC counterparts (Figure 2.4). Additionally, the performance of most methods is comparable once per-position IC weights have been included. This perhaps suggests that information content is the measure that allows us to rigorously identify the highly conserved positions in the binding site; these are presumably the functionally important positions in the interaction between a transcription factor and its DNA binding domain. Given those key positions, the precise way of making use of that information appears less critical; even Consensus, a clearly inferior method utilizing the simplest matching criterion, receives noteworthy improvement from including information content.

Finally, we note that we observe some variation in method performance per transcription factor. This suggests that no single method is optimal for all situations. This is not surprising given the high degree of variation observed in protein-DNA interactions. Whereas in general methods incorporating information content and pairwise nucleotide information are expected to be most effective when searching for DNA binding sites of a particular transcription factor, for a specific transcription factor and its binding sites, an alternate method may perform better. Additionally, in some scenarios it is desirable to allow a higher number of predicted binding sites that can be later eliminated using other approaches (e.g., using cross-genomic information). Analysis similar to the one performed here is likely to prove useful in choosing, for different contexts, a specific method and suitable threshold for finding binding sites of a particular transcription factor.

Chapter 3

Combinatorial Optimization Approach to Motif Finding

3.1 Introduction

In this chapter we consider the problem of motif discovery, or that of finding approximately repeated patterns in unaligned sequence data. It has applications in uncovering transcriptional networks, as short common subsequences in the data may correspond to a regulatory protein's binding sites, and in protein function identification, where short blocks of conserved protein sequence code for important structural or functional elements.

3.1.1 Previous approaches

Motif finding methods are based on the chosen representation method for a group of binding sites. Position specific scoring matrices (PSSMs), which estimate the probability of observing each residue in every position of the motif, are used most commonly to represent binding sites, and they are the basis for many motif finding algorithms. Numerous methods use parameter estimation techniques such as Gibbs

Sampling [Lawrence *et al.* 1993,Hughes *et al.* 2000,Liu *et al.* 2001,Frith *et al.* 2004] and Expectation Maximization (EM) [Lawrence and Reilly 1990, Bailey and Elkan 1995] at their core to arrive at the residue probability settings in the scoring matrices. These approaches commonly make the assumption that the column distributions of the PSSM representing the motif are most different from the distribution of the background sequence. Accordingly, they attempt to maximize the log-likelihood ratio (llr) of the motif model to the background model, defined for a motif of length ℓ embedded in N background sequences as follows:

$$llr = \sum_{j=1}^{\ell} \sum_{b \in \Sigma} n_j(b) \lg \frac{f_j(b)}{f(b)}$$

where $n_j(b)$ is the number of occurrences of base b in column j of the motif, $f_j(b) = n_j(b)/N$ is the corresponding frequency, and $f(b)$ is the probability of observing base b in background.

A method, called CONSENSUS [Hertz and Stormo 1999], is an iterative greedy approach, which attempts to maximize the *relative entropy (RE)*, another popular measure for PSSM based methods, defined as

$$RE = \sum_{j=1}^{\ell} \sum_{b \in \Sigma} f_j(b) \lg \frac{f_j(b)}{f(b)}$$

which is just the normalized log-likelihood ratio. All the above methods are statistical greedy iterative improvement procedures, which use multiple re-starts with random starting points, and often converge to local, rather than global optima.

Consensus representation based methods take an alternate view of motif finding, in which a motif corresponds to a pattern or a regular expression. Ideally, an exhaustive pattern-driven search should be performed to examine every potential motif. While possible for small motif lengths ℓ [Tompa 1999, van Helden *et al.* 2000, Sinha and Tompa 2003,Pavesi *et al.* 2004], that is not a feasible option as ℓ grows. An observation that most of the potential sequence patterns are dissimilar with the true motif has led to the development of sample-driven approximate approaches, which only examine patterns occurring in the data [Rigoutsos and Floratos 1998, Pevzner and Sze 2000, Buhler and Tompa 2002, Sze *et al.* 2004] as well close 'neighbors' of such patterns.

Comparative tests [Tompa *et al.* 2005] have shown that approaches based on different representations of binding sites perform similarly. We take a pattern-based view, and define motif discovery as the problem of finding a set of motif instances whose mutual sequence similarity is maximized.

3.1.2 Combinatorial optimization framework

Here, we consider a combinatorial optimization framework for motif finding that is flexible enough to model several variants of the problem and several types of biological data. For DNA sequences, motif finding algorithms have typically been applied to sets of sequences from a single genome that have been identified as possessing a common motif, either through DNA microarray studies [Tavazoie *et al.* 1999], ChIP-chip experiments [Lee *et al.* 2002] or protein binding microarray experiments [Mukherjee *et al.* 2004]. An orthogonal approach, which attempts to identify regulatory sites among a set of orthologous genes across genomes of varying phylogenetic distance, is adopted by [McGuire *et al.* 2000, McCue *et al.* 2001, Blanchette and Tompa 2002, Kellis *et al.* 2003, Cliften *et al.* 2003]. Yet another formulation of motif finding for DNA sequences, that of 'subtle' motifs, was introduced by [Pevzner and Sze 2000] and tested in simulated data. For protein sequences, motif finding can reveal structural and functional constraints, and especially in the case of divergent sequence motifs, incorporating amino acid substitution matrices [Dayhoff *et al.* 1978, Henikoff and Henikoff 1992] is particularly useful.

Underlying our approach, we consider motif finding as the problem of finding the best gapless local multiple sequence alignment using the sum-of-pairs (SP) scoring scheme, which is one of many reasonable schemes for assessing motif conservation [Osada *et al.* 2004, Schuler *et al.* 1991, Carillo and Lipman 1988]. Recasting motif finding as an equivalent graph problem, we then formulate it as an instance of integer linear programming (ILP), and, since ILP is NP-hard to solve in general, consider its linear programming (LP) relaxation. Typically, the linear programs are very large, numbering in the millions of variables, and prove too difficult even for highly optimized commercial solvers. To reduce the size of the linear programs, we initially employ a number of graph pruning techniques, building upon the ideas of [Gusfield 1993, Vingron and Pevzner 1995, Pevzner and Sze 2000, Lukashin and Rosa 1999]. These fall into the broad category of dead-end elimination (DEE) algorithms (e.g., [Desmet *et al.* 1992]), where sequence positions that are incompatible with the optimal alignment are discarded.

In most cases, the resulting graphs and their linear programs are small, and are easily solved by the CPLEX package [CPLEX 7.1] with AMPL [Fourer *et al.* 2002]. Interestingly, the vast majority of solutions are integral, guaranteeing optimality for the original integer linear program and motif finding problem, and obviating the need to employ ILP solvers. Thus, our approach runs in polynomial time for many practical instances of the problem, which otherwise is known to be NP-hard [Akutsu *et al.* 2000, Wang and Jiang 1994]. In the cases where fractional solutions are found for the linear programs, an ILP solver is applied. In rare more difficult cases, when the graph pruning techniques do not sufficiently reduce the size of the problem, we introduce a heuristic iterative scheme for finding motifs (see section 3.3.5). Finally, given a discovered motif, we test its statistical significance by exactly computing the scores' probability distribution and assessing the number of motifs of the same or better quality that are expected to occur in the data at random. In the cases where optimal solutions are not guaranteed, a bound on the significance value of the potential optimal solution is also given. In practice, the ability of our method to find optimal solutions to large problems, that are also shown to be statistically significant, attests to its overall effectiveness.

We extend our coupled mathematical programming and pruning approach to account for multiple motifs and phylogenetic information, and then test it in various settings. First, we consider the problem of finding shared sequence motifs in protein sequences. Unlike the commonly-used stochastic search methods for motif finding (e.g., [Lawrence *et al.* 1993, Bailey and Elkan 1995], our combinatorial formulation naturally incorporates amino acid substitution matrices, and finds optimal solutions for all the tested datasets. Second, we consider sets of genes known to be regulated by the same *E. coli* transcription factor, and apply our approach to find the corresponding binding sites. Third, we consider the phylogenetic footprinting problem [Blanchette and Tompa 2002], and find shared motifs upstream of orthologous genes. The difficulty of this problem lies in that the sequences may not have had enough evolutionary time to diverge and may share sequence level similarity beyond the functionally important site; incorporation of additional information, in the form of the weightings obtained from a phylogenetic tree relating the species, proves useful in this context. Finally, we consider the 'subtle' motifs formulation of [Pevzner and Sze 2000], where a fixed pattern is inserted into the input sequences with some number of perturbations. In particular, we show that our formulation can be used to find many optimal solutions, thereby retrieving the correct implanted one, along with others that may have occurred by chance. In all scenarios, we show that our method works well in practice, either recovering the known motifs or other motifs of high conservation that are shown to be statistically significant.

3.2 Broad Problem Formulation

The motif finding problem is modeled here as that of finding the ungapped local multiple sequence alignment (MSA) with best sum-of-pairs (SP) score. Informally, given p sequences $\{S_1, \ldots, S_p\}$ and a block length parameter ℓ, the goal is to find an ℓ–long gapless subsequence from each input sequence so that the total similarity among selected blocks is maximized. More formally let s_i^k refer to the ℓ–long block (ℓ–mer) in sequence S_i beginning in position k and let $sim(x, y)$ denote a similarity score between the ℓ–long subsequences x, y. The objective is then to find the set of positions $\{k_1, \ldots, k_p\}$ in each sequence, such that the sum-of-pairs score $\sum_{i<j} sim(s_i^{k_i}, s_j^{k_j})$ is maximized.

It is convenient to consider a graph-theoretic formulation of this problem [Reinert *et al.* 1997]. Let G be an undirected p–partite graph with node set $V_1 \cup \ldots \cup V_p$, where V_i includes a node u for each ℓ–long subsequence s_i^k in the i-th sequence. Note that the subsequences corresponding to two consecutive vertices overlap in $\ell - 1$ positions, and that the V_i's may have varying sizes. Each pair of nodes $u \in V_i$ and $v \in V_j$ $(i \neq j)$, corresponding to subsequences s_i^k and $s_j^{k'}$ in S_i and S_j respectively, is joined by an edge with weight of $w_{uv} = sim(s_i^k, s_j^{k'})$. By this construction G is a complete p–partite graph (see Figure 3.1). The MSA is achieved by picking the highest weight p–partite clique (denoted p–clique) in graph G.

The rest of this section describes the combinatorial optimization framework for the MSA problem. We first explain our approach to the basic formulation of the problem, and then consider extensions. Our method consists of two components: the mathematical programming formulation of the maximum-weight clique problem in G, and graph pruning techniques. Though the problem can theoretically be solved using mathematical programming tools directly, biologically relevant instances are typically very large, numbering in the millions of variables, and would take a prohibitively long time to solve. To reduce the running-times we employ a number of pruning

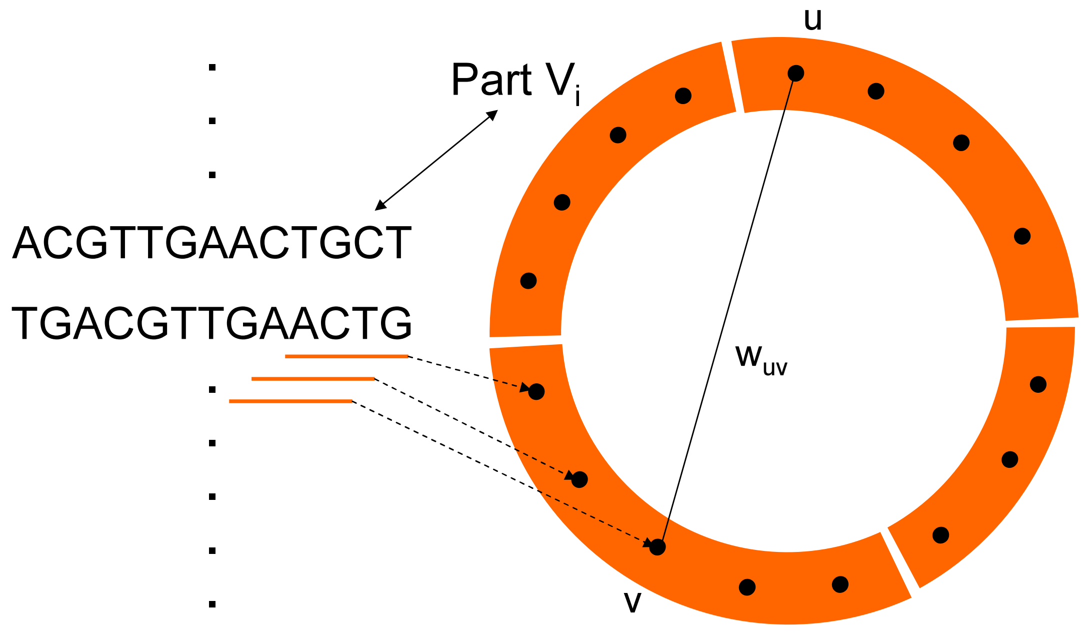


Figure 3.1: Graph representation for the multiple sequence alignment problem. A graph part V_i corresponds to every sequence, and a vertex corresponds to every possible motif position. If the motif length ℓ is four, the picture matches up the last few subsequences to their graph vertices. Every pair of vertices u and v in distinct parts are connected by an edge w_{uv}.

techniques, generally referred to as dead-end elimination (DEE) in the protein design community, which discard vertices and/or edges that cannot possibly be part of the optimal solution[1].

3.3 Basic Motif Finding Framework

3.3.1 Similarity scores

To fully specify the graph as above, we need to define its edge weights. For the DNA motif finding problem we can use the simplest 1/0 similarity score for match/mismatch between pairs of bases, and sum the scores for the ℓ–long blocks in pairs of sequences to derive the weights. When the background distribution of the input sequences is far

[1]Our framework can be recast as a minimization problem as well. In that case the graph edge weights are derived from a distance measure between ℓ–long subsequences. The MSA is then achieved by picking the lowest-weight clique in the graph, and the described DEE techniques can be easily adjusted.

from uniform, it is important for the scoring scheme to reflect the bias in composition. In order to reward matches of more infrequent bases, instead of using 1 for a match, we assign a score of $log(1/f(b))$ for a base b pairing[2], where $f(b)$ is the zero-corrected frequency of base b in the background, and 0 for any mismatch. In practice, we work with integral scores by scaling the floating point numbers to the desired degree of accuracy and rounding (here, we use the scale factor of 100).

We also apply the same basic problem formulation to the protein motif finding problem. Since background correction is less important for protein sequences, our scoring scheme does not reflect compositional bias in this case. Rather, we compute the weights based on amino acid substitution matrices, which assign higher scores to more favorable substitutions and better reflect biochemical properties of such pairings. We experiment with both PAM [Dayhoff *et al.* 1978] and BLOSUM [Henikoff and Henikoff 1992] matrix families. To calculate the edge weights, we sum the matrix entries for amino acid pairs in each position of the ℓ–long block.

3.3.2 Integer linear programming formulation

For graph $G = (V, E)$, where $V = V_1 \cup \ldots \cup V_p$ and $E = \{(u, v) : u \in V_i, v \in V_j, i \neq j\}$, we introduce a binary decision variable x_u for every vertex u, and a binary decision variable y_{uv} for every edge (u, v). Setting x_u to 1 corresponds to selecting vertex u for the p–clique and thus choosing the sequence position corresponding to u in the alignment. Setting variable y_{uv} to 1 corresponds to choosing both the vertices u and v for the p–clique.

The following integer linear program solves the motif finding problem formulated above:

[2]We also experimented with a scheme that assigns a score of $1/f(b)$ for a base b match, which performed similarly.

$$
\begin{array}{lll}
\text{Maximize} & \sum_{(u,v)\in E} w_{uv} \cdot y_{uv} & \\
\text{subject to} & & \\
\quad \sum_{u\in V_j} x_u = 1 & \text{for } 1 \le j \le p & (\textit{node} \text{ constraints}) \\
\quad \sum_{u\in V_j} y_{uv} = x_v & \text{for } 1 \le j \le p, v \in V \setminus V_j & (\textit{edge} \text{ constraints}) \\
\quad x_u, y_{uv} \in \{0,1\} & \text{for } u \in V, (u,v) \in E &
\end{array}
$$

The first set of constraints ensures that exactly one vertex is picked from every graph part, corresponding to one position being chosen from every input sequence. The second set of constraints relates vertex variables to edge variables, allowing the objective function to be expressed in terms of finding a maximum edge-weight clique. An edge is chosen only if it connects two chosen vertices. This formulation is similar to that used by [Kingsford *et al.* 2005] for fixed-backbone protein design and homology modeling.

ILP itself is NP-hard, but replacing the integrality constraints on the x and y variables with $0 \le x_u, y_{uv} \le 1$ allows for a polynomial-time heuristic for the problem. It is important to note that should a linear programming solution happen to be integral, it is guaranteed to be optimal for the original ILP and motif finding problem. Non-integral solutions, on the other hand, are not feasible for the ILP and do not translate to a selection of positions for the MSA problem. Those instances need to be solved by other means, such as using an ILP solver. Interestingly, we find integral solutions in an overwhelming majority of instances (especially after applying our pruning techniques).

3.3.3 Graph pruning techniques

Basic clique-bounds DEE. The idea of our first pruning technique is as follows. Suppose there exists a clique of weight C^* in G. Then a vertex u, whose participation in any possible clique in G reduces the weight of that clique below C^*, is incompatible with the optimal alignment and can be safely eliminated (similar to [Lukashin and Rosa 1999]).

For $u \in V_i$ define $star(u)$ to be a selection of vertices from every graph part other than V_i. Let F_u be the value induced by the edge weights for a $star(u)$ that form best pairwise alignments with u:

$$F_u = \sum_{j \neq i} \max_{v \in V_j} w_{uv} \tag{3.1}$$

If u were to participate in any clique in G, it cannot possibly contribute more than F_u to the weight of the clique. Similarly, let F_i^* be the value of the best possible $star(u)$ among all $u \in V_i$:

$$F_i^* = \max_{u \in V_i} F_u \tag{3.2}$$

F_i^* is an upper bound on what any vertex in V_i can contribute to any alignment.

Now, if F_z, the most a vertex $z \in V_k$ can contribute to a clique, assuming the best possible contributions from all other graph parts, is insufficient compared to the value C^* of an existing clique, i.e. if

$$F_z < 2 \times C^* - \sum_{i \neq k} F_i^*, \tag{3.3}$$

z can be discarded. The clique value C^* is used with a factor of 2 since two edges are accounted for between every pair of graph parts in the above inequality.

In fact, the values of F_i^* are further constrained by requiring a connection to z when z is under consideration. That is, when considering a node $z \in V_k$ to eliminate, and calculating F_i^* according to Equation 3.2 among all possible $u \in V_i$, the F_u of Equation 3.1 is instead computed as:

$$F_u = w_{zu} + \sum_{j \neq i,k} \max_{v \in V_j} w_{uv} \tag{3.4}$$

The value of C^* can be computed from any "good" alignment. We use the weight of the clique imposed by the best overall *star*.

Tighter constraints for clique-bounds DEE. For a vertex $u \in V_i$ and every other V_j, an edge has to connect u to some $v \in V_j$ in any alignment. When calculating F_u, we can constrain its value by considering three-way alignments and requiring that the vertices in the best $star(u)$ chosen as neighbors of u in graph parts other than V_j are also good matches to v. Performing this computation for every pair of u, V_j and considering every edge incident on u would be too costly. Therefore, we only consider such three-way alignments for every vertex $u \in V_i$ and the next part V_{i+1} of the graph (with the last and first parts paired). Essentially, this procedure shifts the emphasis onto edges, allowing better alignments and bounds, and yet eliminates vertices by considering the best edge incident on it.

For a given edge (u, v) with endpoints $u \in V_i$ and $v \in V_{i+1}$ we consider an adjacent *double star* with two centers at u and v, and sharing all the endpoints x_j in the other graph parts, denoted as $dstar(u, v)$; the weight of such a $dstar(u, v)$ is $2w_{uv} + \sum_{\substack{j \neq i \\ j \neq i+1}} (w_{ux_j} + w_{vx_j})$. Now consider a clique $\{u_1 \in V_1, \ldots, u_p \in V_p\}$ of some value C^*, and the sum of its *double stars*:

$$\sum_{i=1 \ldots p} (2w_{u_i u_{i+1}} + \sum_{\substack{j \neq i \\ j \neq i+1}} (w_{u_j u_i} + w_{u_j u_{i+1}})) = 2 \sum_{i=1 \ldots p} \sum_{j \neq i} w_{u_j u_i} = 4C^* \qquad (3.5)$$

The sum is equal to $4C$, since the weights between adjacent graph parts i and $i+1$ are counted twice directly as the first term of the above equation and once each when considering pairs $(i-1, i)$ and $(i+1, i+2)$. Edge weights between non-adjacent graph parts i and j are counted once each when considering adjacent parts' pairs $(i-1, i)$, $(i, i+1)$, $(j-1, j)$, and $(j, j+1)$.

Now we proceed to define F_{uv} with for an edge (u, v) with endpoints $u \in V_i$ and

$v \in V_{i+1}$ as

$$F_{uv} = 2w_{uv} + \sum_{\substack{j \neq i \\ j \neq i+1}} \max_{x \in V_j}(w_{xu} + w_{xv}) \tag{3.6}$$

F_{uv} can be viewed as the weight of the best *double star* centered at the pair of vertices u, v (or edge (u, v)) and it is the best possible contribution to any alignment, if the edge (u, v) was required to be a part of the alignment.

We define F_u for $u \in V_i$ and F_i^* for part i similarly to the above definitions as

$$\begin{aligned} F_u &= \max_{v \in V_{i+1}} F_{uv} && (3.7) \\ F_i^* &= \max_{u \in V_i} F_u && (3.8) \end{aligned}$$

F_u is the value of the best double star centered on vertex $u \in V_i$ and some vertex $v \in V_{i+1}$, and F_i^* is the value of the best double star centered on any pair of vertices $u \in V_i$ and $v \in V_{i+1}$.

For any clique $\{u_1 \in V_1, \ldots, u_p \in V_p\}$ of value C^* in the graph, by Equations 3.5–3.8 we have

$$4C^* = \sum_{i=1\ldots p} (2w_{u_i u_{i+1}} + \sum_{\substack{j \neq i \\ j \neq i+1}} (w_{u_j u_i} + w_{u_j u_{i+1}})) \leq \sum_{i=1\ldots p} F_{u_i u_{i+1}} \leq \sum_{i=1\ldots p} F_{u_i} \leq \sum_{i=1\ldots p} F_i^*$$

Then Equation 3.3, with $2C^*$ replaced by $4C^*$, can be used to eliminate vertices in the same way as before, eliminating a vertex z in a particular graph part if F_z, the value of its best adjacent double star, is insufficient considering best possible contributions from all other graph parts. For best pruning results the value of C^* should be as high as possible; we choose C^* as the clique weight induced by the best overall double star.

Graph Decomposition. DEE techniques work well for simpler instances of the motif finding problem (see Results), but tend to be inadequate, leaving a large final graph, for more complex cases. To overcome this difficulty we propose a divide-and-conquer graph decomposition approach. For every graph part i and vertex $u \in V_i$ we consider induced subgraphs $G^u = (V^u, E^u)$ in turn, where $V^u = u \cup V \setminus V_i$. Application of the *clique-bounds DEE* technique to graphs G^u is very effective since one of the graph parts, G_i^u contains only one vertex, u, and all the F and F^* values that need to be recomputed for the new graph G^u are greatly constrained. The process of updating the F and F^* values is efficient as the changes are localized to one part in the graph. Importantly, the C^* remains intact, since the clique of that larger value exists in the original graph and can be used for the decomposed one, helping to eliminate vertices. For some of the vertices u, iterative application of the DEE criterion and re-computation of the F and F^* values causes G^u to become disconnected, implying that vertex u cannot be part of the optimal alignment. Such a vertex u is marked for deletion, and that information is propagated to all subsequently considered induced subgraphs, further constraining the corresponding F and F^* values and helping to eliminate other vertices in turn.

3.3.4 Statistical significance

While the method described above finds the optimal solution for the underlying graph problem with the corresponding value of the objective function, it is important to assess how likely it is for such a motif to have occurred at random. To that end, we propose a strategy to calculate statistical significance, in our case measured by the number of motifs of equal or better quality expected to occur in random data with the same characteristics [Altschul 2005].

Let the score of the motif in question be denoted by S. Recall that $f(b)$ is the zero-corrected background frequency of nucleotide b in the input sequences, and $sim(b_1, b_2)$ is the integral score computed for all residue pairs as above.

We compute $P_i(X)$, the probability of observing a motif of length i and of score

X in N sequences, in the first two steps of the following, and infer the e-value of score S in the last two:

1. Calculate the exact probability distribution $P_1(x)$ for all possible SP-scores for a single column of N random residues. We use the multinomial distribution to compute the probability of observing every combination of residues in the column according to the background distribution, and calculate the corresponding SP-score by summing appropriate $sim(b_1, b_2)$ values. We then add probabilities for the same scores resulting from different residue combinations. To make the computation feasible for the protein alphabet and for large numbers of sequences, we calculate the scores and probabilities in such an order that every new score and probability is computable from the previous one by a local update operation.

2. Calculate the exact probability distribution $P_l(x)$ for all possible SP-scores for ℓ random columns of letters. This can be done by convolution of $P_1(x)$ as in [Tatusov *et al.* 1994], where we inductively construct a distribution for i columns based on the distribution for $i-1$ columns, $P_{i-1}(x)$, and the single column distribution $P_1(x)$.

3. For a given score of interest S, we calculate the probability $R_l(S)$ that an ℓ–long pattern has score greater than or equal to S by chance alone. This is simply $R_l(S) = \sum_{x>=S} P_l(x)$.

4. Finally, we compute the total number of possible motifs in the data. If the sequences have lengths $L_1, \ldots, L_N$, then the search space size $L = \prod_i (L_i - l + 1)$, where ℓ is the fixed length of the motif. The expected number of alignments with score at least S by chance along, or the e-value, is then $L * R_l(S)$.

3.3.5 Algorithm description

We combine the various elements described above and apply them in the order of increasing complexity (See Figure 3.2). At every juncture a decision is made whether to send the problem in its current state to the LP solver, and this decision hinges on the size of the graph. If the graph is "small" enough for some suitable definition of small (currently set at 800 vertices), we submit the appropriate linear program to the LP solver and, if necessary, to the ILP solver. To reduce the graph to that necessary small size, we apply the DEE variants, running each one of them to convergence so that no further pruning is possible. The process is terminated when the specified graph size has been reached. First, we attempt to prune the graph using *basic clique-bounds DEE*; then we consider tighter bound computations; lastly we employ *graph decomposition* in conjunction with the DEE methods.

In the rare cases when the above procedure is unable to sufficiently prune the graph, we perform what we call speculative pruning using higher C^* values, which do not necessarily correspond to known cliques in graph G. Three outcomes of such pruning are possible: (i) The graph is eliminated completely. This guarantees that a clique of value C^* does not exist in G. (ii) The pruning is once again inadequate to reduce the size of the graph sufficiently. (iii) The pruning procedure converges to a small graph that can be piped to the solver. We search the space of possible C^* values until we find one that produces outcome (iii). To identify such a value we first translate the possible clique scores into their corresponding e-values, and then perform binary search on the e-value exponent, transforming the current e-value back to its score equivalent. This method converges quickly, typically locating an effective C^* in fewer than 10 iterations.

If the optimal solution for the final reduced graph is better than the C^* used in pruning, then it is also optimal for the original graph. Otherwise, using speculative pruning no longer guarantees optimality of the final motif, as a vertex that is a member

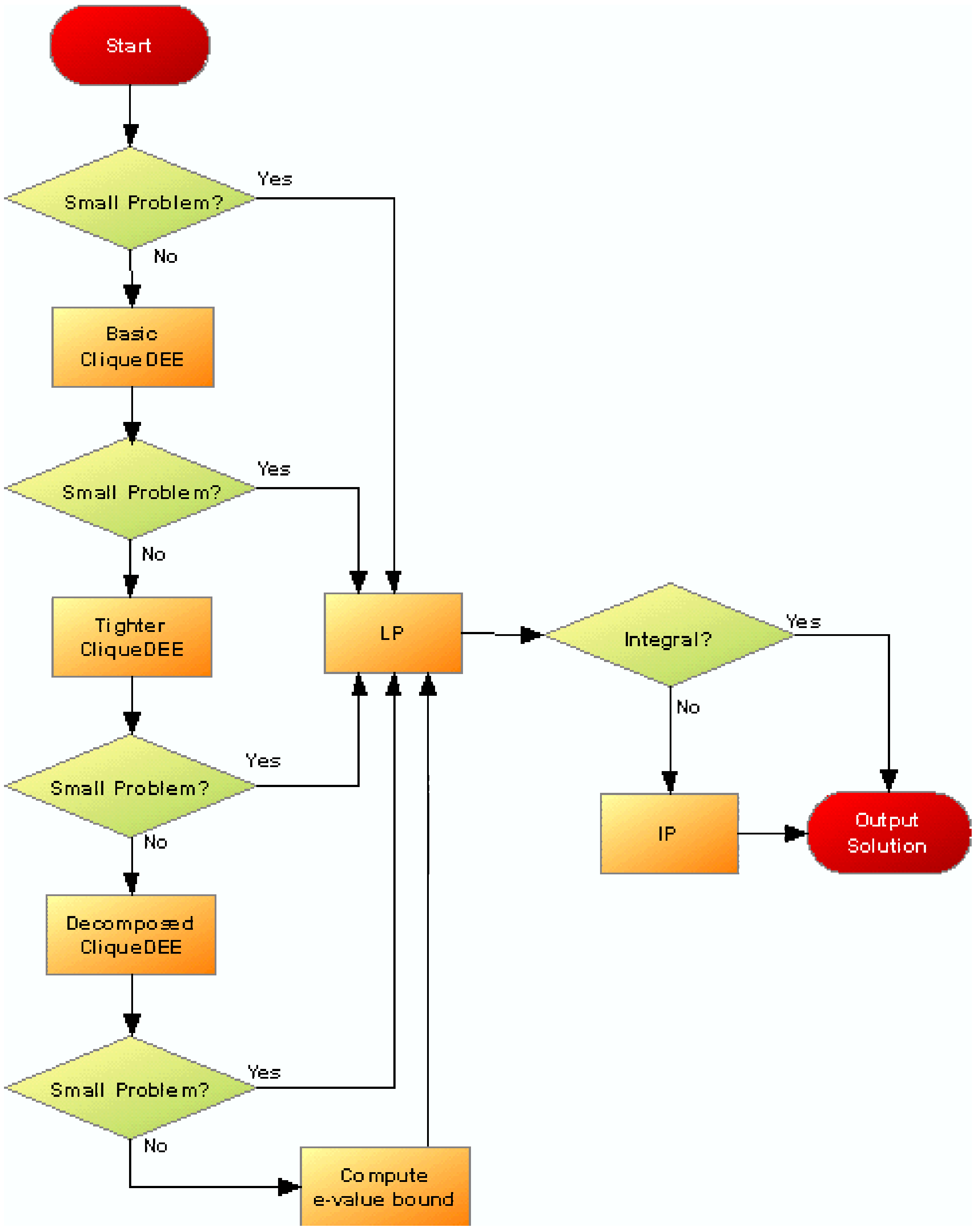


Figure 3.2: Algorithm flow chart. Every diamond-shaped node is a decision node, rectangular nodes are various execution steps, and oval nodes are head and terminal nodes.

of the optimal solution may have been eliminated in the pruning step. However, we can place an e-value bound on the actual optimal solution. Since the optimal solution is no better than the value of C^* used to obtain the pruned graph, the e-value corresponding to C^* provides us with the bound.

3.4 Subtle Motifs Framework

Pevzner and Sze [Pevzner and Sze 2000] introduced the 'subtle' motifs version of the motif finding problem. They formulate it as a signal finding problem in which an unknown pattern of a given length is inserted with modifications into each of the input sequences. The positions of insertion are unknown, as are the modifications in the instances of the pattern. Pevzner and Sze focus on what they call the (l, d)-signal version, in which the pattern is a string of length ℓ and each pattern instance differs from the pattern in exactly d positions. The mutations are allowed to occur anywhere in the pattern, and thus any two instances of the pattern may differ from each other in as many as $2d$ positions. On the other hand, if two subsequences differ in more than $2d$ positions, they cannot possibly be instances of the implanted pattern.

We can solve the subtle motifs problem just as above by formulating it as a multiple sequence alignment with the sum-of-pairs score. The graph version of the problem remains largely the same except that it is no longer a complete p–partite graph. By definition, vertices that correspond to subsequences whose Hamming distance is greater than $2d$ should not be connected by an edge, as such an edge cannot possibly be part of the optimal clique. The weights on the edges, as suggested in [Pevzner and Sze 2000], are computed by considering the number of matches and mismatches.

It is straightforward to adjust our linear program to reflect the fact that graph G is no longer a complete graph by removing variables corresponding to non-existent edges.

$$\begin{array}{llll}
\text{Maximize} & \sum_{(u,v)\in E} w_{uv} \cdot y_{uv} & & \\
\text{subject to} & & & \\
& \sum_{u\in V_j} x_u = 1 & \text{for } 1 \le j \le N & (\textit{node} \text{ constraints}) \\
& \sum_{u\in V_j,(u,v)\in E} y_{uv} = x_v & \text{for } 1 \le j \le N, v \in V \setminus V_j & (\textit{edge} \text{ constraints}) \\
& 0 \le x_u, y_{uv} \le 1 & \text{for } u \in V, (u,v) \in E &
\end{array}$$

The main difference is in the *edge* constraints, in that the summation is made over the existing edges only.

3.4.1 Graph pruning and decomposition

We begin by noting that the subtle motifs graphs may be pruned using any of the methods introduced by previous authors (e.g., in [Pevzner and Sze 2000, Sze *et al.* 2004]), and our LP/ILP formulation can be applied whenever the graph size has decreased sufficiently. Additionally, DEE methods introduced above can be adjusted for these graphs as well. We chose to experiment with two pruning and decomposition procedures.

The first technique we apply is a dead-end elimination routine, suggested in [Vingron and Pevzner 1995, Pevzner and Sze 2000]. Here we can use connectivity properties of $G = (V, E)$, which no longer is a complete p–partite graph, to prune "dead-end" vertices and edges. Following the notation of [Pevzner and Sze 2000], let vertex $u \in V_i$ be a *neighbor* of vertex $v \in V_j$ if (u, v) is an edge in the graph, and vertex x be a *neighbor* of an edge (u, v) if $\{u, v, x\}$ is a connected triangle in the graph. The strategy we call *vertex removal* is to delete any vertex u that does not have at least one neighbor in every part of G other than V_i. *Edge removal* eliminates any edge (u, v) that does not have at least one neighbor in every part of G excluding V_i and V_j. These two strategies, collectively referred to as *connectivity DEE*, are applied iteratively until no further vertices or edges are detected for removal.

We also introduce a second *graph decomposition II* technique for solving more difficult instances of the subtle motifs problem. Though any DEE routine can be employed in conjunction with graph decomposition, we concentrate on the iterative application of *vertex removal* and *edge removal* procedures. The idea of decomposition becomes more effective if modified slightly from its version above. Here we choose an arbitrary part V_i of V, and consider all the vertices $u \in V_i$ in turn. The intuition is that some vertex u in V_i has to be in the optimal alignment, and considering each one exhausts the possibilities. As before, we consider induced subgraphs $G^u = (V^u, E^u)$ and process them with the *connectivity DEE* routines. For some u, their corresponding graphs G^u become disconnected, and those vertices can be discarded; for others, typically, the graph remaining after this processing is small. As a final step we solve a number of these small linear programs, and select the optimal solution to the original problem among them. Note that for some difficult and dense instances of the subtle motifs problem, we can extend this *graph decomposition II* technique to multiple graph parts, considering pairs or even triplets of vertices for elimination.

3.5 Other Motif Finding Frameworks

3.5.1 Phylogenetic footprinting

As mentioned earlier, one way of finding regulatory sites is to look for them among a set of homologous genes across species. In this case additional data, in the form of the phylogenetic tree relating the species, is available and should be exploited. It is especially important when closely related species are part of the input, and, unweighted, they contribute duplicate information and skew the alignment. We use a phylogenetic tree and branch lengths when calculating the edge weights in the graph, with highly diverged sequence pairs getting larger weights. The precise weighting scheme follows the ideas of weighted progressive alignment [Feng and Doolittle 1987], in which weights α_i are computed for every sequence i. The calculation sums branch lengths along the path from the tree root to the sequence at the leaf, splitting shared branches among the descendant leaves, and thereby reducing the weight for related sequences. In essence, we solve a multiple sequence alignment problem with weighted SP-score using match/mismatch, where the computed weight for a pair of positions in sequences i and j is multiplied by $\alpha_i \times \alpha_j$. The rest of the algorithm operates as in the basic motif finding case above, employing the same linear programming formulation and DEE techniques.

3.5.2 Multiple motifs

Here we give several extensions to address the issue of multiple motifs existing in a set of sequences. First, distinct multiple motifs, such as sets of binding sites for two different transcription factors, can be found iteratively by first locating a single optimal motif, masking it out from the problem instance, and then looking for the next one. We mask the previous motif by deleting its solution vertices from the original graph, and then reapplying the DEE/LP techniques to locate the next optimal motif.

Second, it is possible to solve iteratively several ILPs in order to find multiple near-optimal solutions, corresponding to the best cliques of successively decreasing total weights. At iteration t, we add a constraint to the integer linear programming formulation so as to exclude all previously discovered solutions:

$$\sum_{u \in S_k} x_u \leq p - 1 \quad \text{for } k = 1, \ldots, t-1, \tag{3.9}$$

where S_k contains the optimal set of vertices found in iteration k. This requires that the new solution differs from all previous ones in at least one graph part. We note that to use this constraint for the basic formulation of the motif finding problem, the DEE methods given above have to be modified so as not to eliminate nodes taking part in near-optimal but not necessarily optimal solutions. For the subtle motifs problem, the DEE methods only eliminate nodes and edges based on whether they can take part in any clique in the graph, and thus constraint 3.9 can be immediately applied to iteratively find all possible cliques.

Finally, finding *repeated* motifs where a single pattern occurs in m distinct positions in each sequence, requires a slight modification to the construction of the graph $G = (V, E)$. The set of vertices remains the same, but the edge set is modified in two ways. First, we introduce edges between vertices corresponding to positions in the same sequence, since the instances of the motif are all mutually similar. Secondly, to address the issue of low complexity regions, such as poly-A repeats, being selected as a solution, we would like the motif instances to be non-overlapping. Thus, vertices corresponding to overlapping ℓ–mers in each sequence are not connected by an edge. The ILP formulation to address this problem is similar to the case of single motif; additional constraints are needed, though, to ensure a proper vertex selection since the graph now has edges within each part.

$$\begin{array}{ll}
\text{Maximize} & \sum_{(u,v)\in E} w_{uv} \cdot y_{uv} \\
\text{subject to} & \\
\quad \sum_{u\in V_j} x_u = m & \text{for } 1 \leq j \leq p \\
\quad \sum_{u\in V_j} y_{uv} = m \times x_v & \text{for } 1 \leq j \leq p, v \in V \setminus V_j \\
\quad \sum_{u\in V_j} y_{uv} = (m-1) \times x_v & \text{for } 1 \leq j \leq p, v \in V_j \\
\quad x_u, y_{uv} \in \{0,1\} & \text{for } u \in V, (u,v) \in E
\end{array}$$

The difference with the single motif ILP in the *node* constraints is in that m of the vertices are chosen in each graph part, corresponding to m positions in each input sequence. The next two sets of constraints ensure a proper edge selection between chosen vertices both inside and between partitions, the first taking care of inter-part edges, and the second intra-part edges.

3.6 Experimental Analysis

We apply the combinatorial optimization framework to several motif finding problems. We attempt to discover motifs in instances arising from both DNA and protein sequence data, and compare them with known motifs. We then consider the phylogenetic footprinting problem, and demonstrate multiple motifs' discovery. Finally, we discuss subtle motif finding in simulated data.

3.6.1 Protein motifs

We study the performance of our algorithm on a number of protein datasets with different characteristics. The datasets, summarized in Table 3.1, were constructed using the SwissProt [Boeckmann *et al.* 2003] database from the descriptions of [Lawrence *et al.* 1993, Lukashin and Rosa 1999, Neuwald *et al.* 1995]. These datasets are highly variable in the number and length of their input sequences as well as the degree of motif conservation. The motif length parameters are set based on the lengths described by the above authors. The amino acid substitution matrix we used for all the datasets reported below is BLOSUM62. For more closely related proteins, like the human tumor necrosis factor (TNF) proteins, we also experimented with the BLOSUM80 and PAM100 matrices. However, choice of substitution matrix and slight variations in sought motif length do not substantially affect the results, as similar motifs are found. The five datasets are of varying difficulty to solve, with some employing the basic *clique-bounds DEE* technique to prune the graphs, while others require more elaborate pruning that is constrained by three-way alignments (see Table 3.1).

In all the test datasets our algorithm recovers the motifs found by [Lawrence *et al.* 1993, Lukashin and Rosa 1999, Neuwald *et al.* 1995] and reported in the literature. As described by [Lawrence *et al.* 1993], the HTH dataset is very diverse, and

Dataset	**Seq**	**Len**	$\lvert V\rvert$	**DEE**	$\lvert V_{\mathbf{DEE}}\rvert$	**E-value**
Lipocalin	5	16	844	(1)	5	3.80×10^{-16}
Helix-Turn-Helix	30	20	6870	(1,2,3)	260	3.88×10^{-67}
Tumor Necrosis Factor	10	17	2329	(1)	10	1.50×10^{-40}
Zinc Metallopeptidase	10	12	7761	(1,3)	10	5.82×10^{-23}
Immunoglobulin Fold	18	10	7498	(1,2,3)	187	3.04×10^{-24}

Table 3.1: Descriptions of the protein datasets. The first two datasets are from [Lawrence *et al.* 1993], the next two from [Lukashin and Rosa 1999], and the last one from [Neuwald *et al.* 1995]. ***Seq*** is the number of input protein sequences; ***Len*** is the length of the protein motif searched for; $|V|$ is the number of vertices in the original graph constructed from the dataset; ***DEE*** gives the methods involved in pruning the graph and are denoted by (1) *clique-bounds DEE*, (2) *graph decomposition*, and (3) tighter constrained bounds. $|V_{DEE}|$ is the number of vertices in the graph after pruning; ***E-value*** lists the e-value of the found motif.

the detection of the motif is a difficult task. Nonetheless, our HTH motif is identical to that of [Lawrence *et al.* 1993], and agrees with the known annotations in every sequence. We likewise find the lipocalin motif; it is a weak motif with few generally conserved residues that is in perfect correspondence with the known lipocalin signature. We also precisely recover the immunoglobulin fold, TNF and zinc metallopeptidase motifs. In contrast to [Lukashin and Rosa 1999], who limit sequence lengths to 500, we retain the original protein sequences, making the problem more difficult computationally as the average sequence length in the zinc metallopeptidase dataset is approximately 800, and some sequences are as long as 1300 residues[3]. Nonetheless, we recover the zinc metallopeptidase motif using our method. It is identical to the motif reported by [Lukashin and Rosa 1999] in nine of ten sequences (see Table 3.2); yet, with the difference in the last sequence, the motif discovered by our method is superior both in terms of sequence conservation and statistical significance at e-value of $5.7729e^{-023}$ vs $1.12155e^{-021}$ for [Lukashin and Rosa 1999].

[3]Our implementation of the method of [Lukashin and Rosa 1999] applied to the zinc metallopeptidase dataset with full length sequences failed to converge.

ID	**LP/DEE Motif**		**Lukashin Motif**	
	Position	Motif	Position	Motif
ACET	411	`VAHHEMGHIQYF`	411	`VAHHEMGHIQYF`
AMPN	384	`VIAHELAHQWFG`	384	`VIAHELAHQWFG`
BMP1	210	`IVVHELGHVVGF`	210	`IVVHELGHVVGF`
MM01	215	`VAAHELGHSLGL`	215	`VAAHELGHSLGL`
MM02	400	`VAAHEFGHAMGL`	400	`VAAHEFGHAMGL`
MM03	215	`VAAHEIGHSLGL`	215	`VAAHEIGHSLGL`
LKHA	292	`VIAHEISHSWTG`	292	`VIAHEISHSWTG`
MEPA	152	`IIEHEILHALGF`	152	`IIEHEILHALGF`
PSA	349	`VVGHELAHQWFG`	349	`VVGHELAHQWFG`
ACE	985	`VAHHEMGHIQYF`	387	`TVHHEMGHIQYY`

Table 3.2: Human zinc metallopeptidase motif as found by the LP/DEE method and [Lukashin and Rosa 1999]. All sequences are designated by their SwissProt entries with extension *HUMAN* omitted.

3.6.2 DNA motifs

We analyze the performance of our method on a set of sequences consisting of DNA binding sites embedded in their respective upstream regions (up to 600 bp) for a number of regulatory proteins of *E. coli.* Our dataset was constructed from the data of [Robison *et al.* 1998, McGuire *et al.* 2000], as described in [Osada *et al.* 2004]. In short, we remove sigma factors; binding sites that could not be unambiguously located in the genome; and transcription factors with fewer than three known sites. Additionally, we include only one copy of a sequence containing multiple known binding sites. Motif length parameters are set according to [Robison *et al.* 1998], based on the length of the consensus binding site determined in biological studies. The final dataset consists of 36 transcription factors, with the number of sequences for each factor ranging between 3 and 34, and the length of the binding site ranging between 11 and 48 (see Table 3.4).

To evaluate performance and determine the extent of agreement between the motif predictions versus known motifs, we use two statistics: nucleotide level *performance coefficient*, abbreviated as *nPC* [Pevzner and Sze 2000], and site level *site sensitivity* or *sSn* [Tompa *et al.* 2005]. We use the notation of [Tompa *et al.* 2005] to define these statistics. Let *nTP*, *nFP*, *nTN*, *nFN* refer to nucleotide level true positives, false positives, true negatives and false negatives respectively. For example, *nTP* is the number of nucleotides in common between the known and predicted motifs. The first statistic we employ, the nucleotide level performance coefficient (*nPC*), measures the degree of overlap between known and predicted motifs, and is defined as $nPC = nTP/(nTP + nFN + nFP)$. The performance coefficient is a stringent statistic, penalizing a method for both failing to identify any nucleotide belonging to the motif as well as falsely predicting any nucleotide not belonging to the motif. The second statistic we employ is site level sensitivity (*sSn*). Following [Tompa *et al.* 2005] we consider two sites to be overlapping if they overlap by at least one-quarter the length of the site. Defining site level statistics similarly to the nucleotide level statistics above (e.g., site level true positives, sTP, is the number of known sites overlapped by predicted sites), site level sensitivity is $sSn = sTP/(sTP + sFN)$.

Biased-Composition Data

We evaluate the effectiveness of our scoring scheme in finding motifs embedded in compositionally-biased background sequences. We use a selection of five transcription factor datasets of varying levels of conservation, measured by their information content, and perturb the background sequences to reflect increasingly biased probability distributions. For each background position, a base is selected at random according to a probability distribution in which base G is chosen with some probability $pr(G)$ and the other bases with probability $(1 - pr(G))/3$ each. Measurement of performance in terms of the nucleotide performance coefficient is summarized in Table 3.3 for various values of $pr(G)$. We find motifs of very high *nPC* values; all the motifs also exhibit *sSn* values of identically 1.0 for all datasets and all $pr(G)$ settings (data not shown), attesting to the fact that our scoring scheme is successfully able to

TF	IC	Bias				
		0.25	0.5	0.75	0.9	1.0
araC	1.00041	0.8113	0.9592	0.9592	0.9592	0.9592
cpxR	1.17034	1.0000	0.8261	0.9811	0.9811	0.9811
dnaA	1.45351	1.0000	0.7647	0.7647	1.0000	1.0000
galR	1.34756	0.8824	0.8824	1.0000	1.0000	1.0000
narP	1.40273	1.0000	1.0000	1.0000	1.0000	1.0000

Table 3.3: Scoring method evaluation in terms of performance coefficient in biased-composition simulated data. The first column identifies the transcription factor dataset; the second column measures degree of conservation (information content) of the known motif. The rest of the columns list *nPC*, the nucleotide level performance coefficient of our scoring scheme, in compositionally biased background sequences with $pr(G)$ indicated in the column heading and the frequencies of all other bases split equally.

correct for bias in sequence composition. We are able to identify the implanted motifs with better performance statistics than in other simulated data studies (e.g., [Tompa *et al.* 2005]) largely due to the background sequence model we employ. Whereas we assume independence between positions, others have applied second or third order Markov chains, which better simulate biological sequences, to generate their data.

Biological Data

We summarize the findings of our algorithm in Table 3.4. Of the 36 transcription factors we considered, 25 were solved in seconds with the application of *clique-bounds DEE*, some using tighter bounds constrained by three-way alignments; seven required the application of *graph decomposition* with tighter *clique-bounds DEE*, and took a few minutes to three hours to solve. For the remaining four datasets we use speculative pruning, finding highly significant solutions for two of them, albeit without the guarantee of optimality, and no significant solution for one; the final dataset, *crp*, proved too large to be pruned effectively, and we consider no solution to have been found for it. In the entire data collection, all but one of the problems resulted in integral solutions to their linear programs, which immediately translated to a motif selection. The one instance with the fractional solution was easily solved by the ILP solver. Setting the e-value threshold at 1.0, we find 32 statistically significant motifs.

TF	**Seq**	**Len**	**IC**	**RE**	**E-value**	***nPC***	***sSn***
ada	3	31	1.3000	1.0846	9.16×10^{-1}	0.1341	0.33
araC	4	48	1.1437	0.9940	1.15×10^{-3}	0.3474	0.50
arcA	11	15	1.2505	1.1992	4.31×10^{-6}	0.4224	0.73
argR	8	18	1.2990	1.2149	1.30×10^{-7}	0.2857	0.50
cpxR	7	15	1.3290	1.2337	1.09×10^{-5}	0.5556	0.71
cytR	5	18	1.2317	1.1069	2.48×10^{-1}	0.0588	0.20
dnaA	6	15	1.4535	1.3300	6.12×10^{-6}	1.0000	1.00
fadR	5	17	1.3466	1.2074	1.33×10^{-2}	0.5455	0.80
fis*	8	35	0.8927	0.8376	1.37×10^{-6}	0.1966	0.38
flhCD	3	31	1.3942	1.1656	4.79×10^{-3}	0.0000	0.00
fnr	10	22	1.1025	1.0476	1.85×10^{-9}	0.6176	0.80
fruR	10	16	1.2094	1.1491	5.52×10^{-8}	0.8182	0.90
fur	7	18	1.3285	1.2332	1.28×10^{-8}	0.4237	0.71
galR	7	16	1.5445	1.4347	1.52×10^{-16}	0.5034	0.71
glpR	4	20	1.4227	1.2441	2.63×10^{-2}	0.5534	0.75
hns	5	11	1.5175	1.3660	2.25	0.0000	0.00
ihf*	19	48	0.3932	0.3859	$2.26 \times 10^{+8}$	0.0381	0.16
lexA	17	20	1.1481	1.1192	1.01×10^{-40}	0.7215	0.88
lrp	4	25	1.2879	1.1237	6.44×10^{-2}	0.0989	0.25
malT	6	10	1.5071	1.3815	1.73×10^{-1}	0.0000	0.00
metJ	5	16	1.6842	1.5195	3.37×10^{-12}	0.6495	1.00
metR	6	15	1.3097	1.1970	6.57×10^{-2}	0.0000	0.00
modE	3	24	1.5618	1.3145	3.95×10^{-4}	1.0000	1.00
nagC	5	23	1.2795	1.1462	1.03×10^{-3}	0.0360	0.20
narL	10	16	1.1391	1.0828	8.06×10^{-4}	0.8182	0.90
narP	4	16	1.4534	1.2737	7.48×10^{-4}	0.0000	0.00
ntrC	4	17	1.6621	1.4605	1.28×10^{-8}	0.6386	1.00
ompR	4	20	1.3566	1.1860	4.27×10^{-6}	0.0000	0.00
oxyR	4	39	1.0965	0.9521	2.64	0.0796	0.25
phoB	8	22	1.1567	1.0835	4.14×10^{-9}	0.8051	1.00
purR	20	26	0.8305	0.8147	1.53×10^{-37}	0.7247	0.95
soxS*	11	35	0.7771	0.7453	1.26×10^{-9}	0.0815	0.27
trpR	4	24	1.4069	1.2291	3.74×10^{-6}	0.8462	1.00
tus	5	23	1.5839	1.4276	1.05×10^{-17}	0.8400	1.00
tyrR	10	22	1.0693	1.0159	3.63×10^{-9}	0.5017	0.70

Table 3.4: Listing of complete results for the transcription factors dataset. ***Seq*** is the number of input sequences; ***Len*** is the length of the motif searched for. The rest of the listed measures refer to the motif discovered by the algorithm: ***IC*** is the information content; ***RE*** is the relative entropy; ***E-value*** is the e-value, computed according to out statistical significance assessment; ***nPC*** is the nucleotide level performance coefficient; and ***sSn*** is the site level sensitivity. The three starred entries indicate potentially non-optimal solutions. For them we provide a bound on the significance value of a potential optimal solution according to the method detailed in the **Algorithm Description** section above. We list the e-value of the obtained motif and its bound in parentheses: fis $1.37 \times 10^{-6} (2.29 \times 10^{-7})$; ihf $2.26 \times 10^{+8} (3.98 \times 10^{-31})$; soxS $1.26 \times 10^{-9} (5.25 \times 10^{-14})$. The final dataset, *crp*, with no identified solution, was the largest at 34 sequences, and the site length of 22.

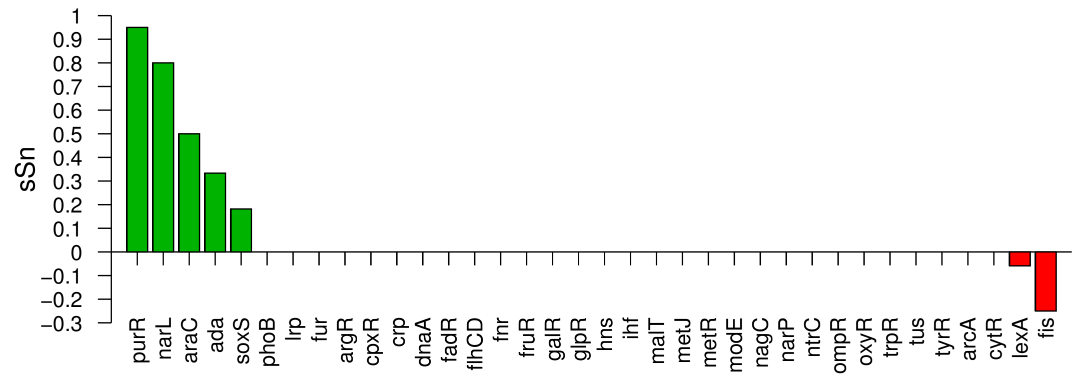
Of the three transcription factor datasets with no significant solutions, one, *hns*, is a short motif, and the other two, *oxyr* and *ihf*, are very poorly conserved motifs with low information content[4]. In general, motifs found by our method exhibit both equal or higher information content (measuring motif conservation) and relative entropy (measuring the difference with the background distribution) than that of the sets of known transcription factor binding sites.

Evaluating performance of any motif finding algorithm is not a straightforward task, as other, better-conserved, biologically-relevant motifs may exist in the data. To that end, we compute the degree of agreement between the motifs discovered by our method and the known binding sites, and also compare them with motifs found by widely used stochastic-search motif finders, Gibbs Motif Sampler [Thompson *et al.* 2003] and MEME [Bailey and Elkan 1995]. We run these two methods, requiring one motif instance per sequence, 20 random restarts, and using motif length parameters of [Robison *et al.* 1998], while leaving other parameters at their defaults.

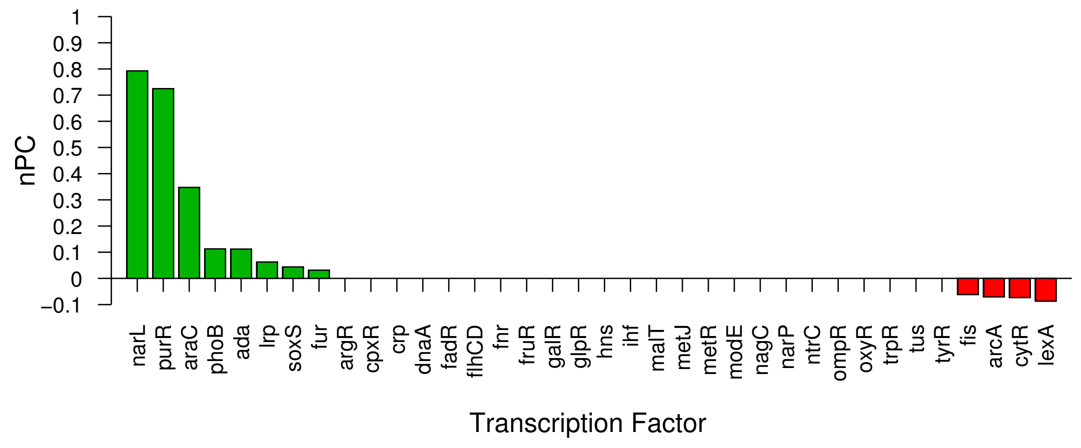
Gibbs Motif Sampler comparison. Gibbs Motif Sampler discovers motifs for 31 transcription factors; no statistically significant motif is reported for the other five. In two of these five datasets, *ihf* and *crp*, the LP/DEE method likewise fails to discover a significant motif; in one other dataset (*flhCD*), surprisingly, a motif of high information content and relative entropy does exist (see Table 3.4).

We chart the performance of our algorithm versus Gibbs Motif Sampler in Figure 3.3. Each bar in the chart measures the difference in site level sensitivity (Figure 3.3(a)) and nucleotide performance coefficient (Figure 3.3(b)) between the two methods. While the absolute values for the two statistics differ, the qualitative results are similar. Very large differences in performance are observed for three transcription factors, *narL*, *purR*, and *araC*, with our method identifying them almost completely, and the Gibbs method entirely misidentifying the motif in *narL*, and reporting no

[4]The information content for the known and discovered motifs respectively is: *oxyr* (0.89, 0.95), *ihf* (0.36, 0.39).



(a) Difference in site sensitivity between LP/DEE and Gibbs.



(b) Difference in nucleotide performance coefficient between LP/DEE and Gibbs.

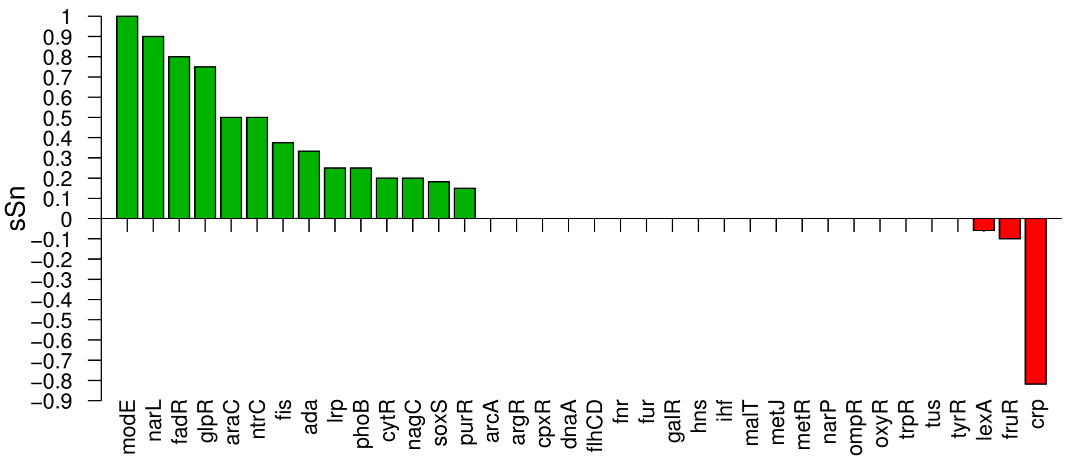
Figure 3.3: Performance comparison between the LP/DEE method and Gibbs Motif Sampler [Thompson *et al.* 2003] when identifying known regulatory sites [Robison *et al.* 1998]. For every transcription factor dataset the height of the bar indicates the difference in the metric, with green bars specifying better performance for our method and red bars otherwise. When a method fails to report a statistically significant motif, the number of correctly identified nucleotides and sites is set to zero.

significant motif for the other two datasets. For the rest of the datasets, the methods perform similarly, either both identifying extensive overlap with the biological motif or both reporting near zero values. Overall our method discovers motifs that exhibit a slightly higher overlap with the known transcription factor binding sites, such that the average nPC for our method over the entire dataset is 0.395, versus 0.344 for the Gibbs Motif Sampler. For site level sensitivity, the average sSn for our method is 0.533 versus 0.472 for the Gibbs Motif Sampler. It is also interesting to note that the average relative entropy of the discovered motifs, a measure related to one maximized by the Gibbs algorithm, is essentially identical for the two algorithms, with the difference being smaller than 0.001 for the majority of the datasets.

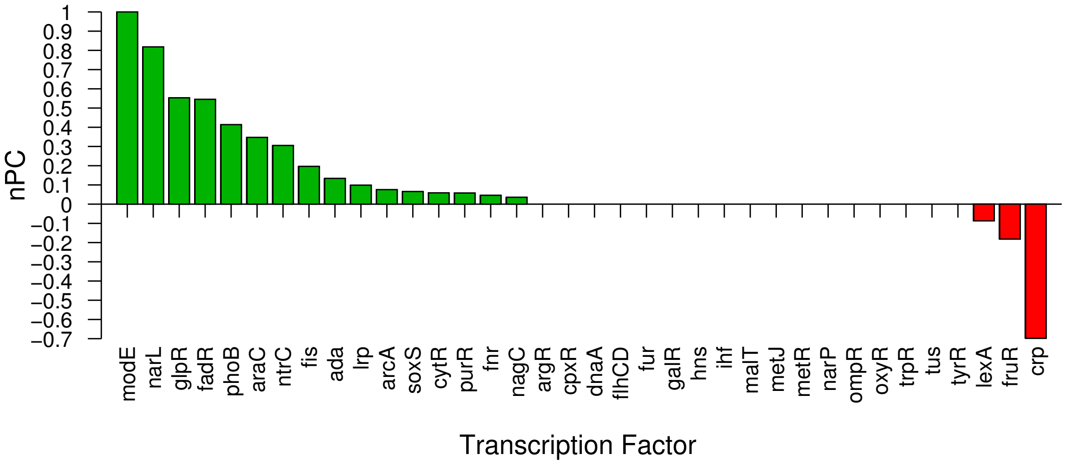
MEME comparison. MEME reports motifs and their corresponding e-values for all 36 transcription factor datasets. Using an e-value cutoff of 1.0 for both LP/DEE algorithm and MEME, and setting all the overlap numbers for datasets with no reported significant motifs to zero, we chart the performance of the methods in Figure 3.4.

As before, each bar in the chart measures the difference in site level sensitivity (Figure 3.4(a)) and nucleotide performance coefficient (Figure 3.4(b)) between the two methods. With the exception of *crp*, the LP/DEE method significantly outperforms MEME, reporting both higher site level sensitivity and nucleotide performance coefficient values for approximately half of the datasets. Average overlap statistics differ significantly as well: average nPC for MEME is 0.290 versus 0.395 for LP/DEE and average sSn is 0.383 versus 0.533.

Note, however, that the large difference in performance is mainly due to MEME's significance assessment of the discovered motifs. It is possible that MEME's significance computation is unnecessarily conservative for our dataset. If we entirely disregard MEME's reported e-values, and focus exclusively on the raw coefficients for overlap with the known motifs, the difference in performance between LP/DEE and MEME is not as skewed (see Figure 3.5). Very large differences are observed for

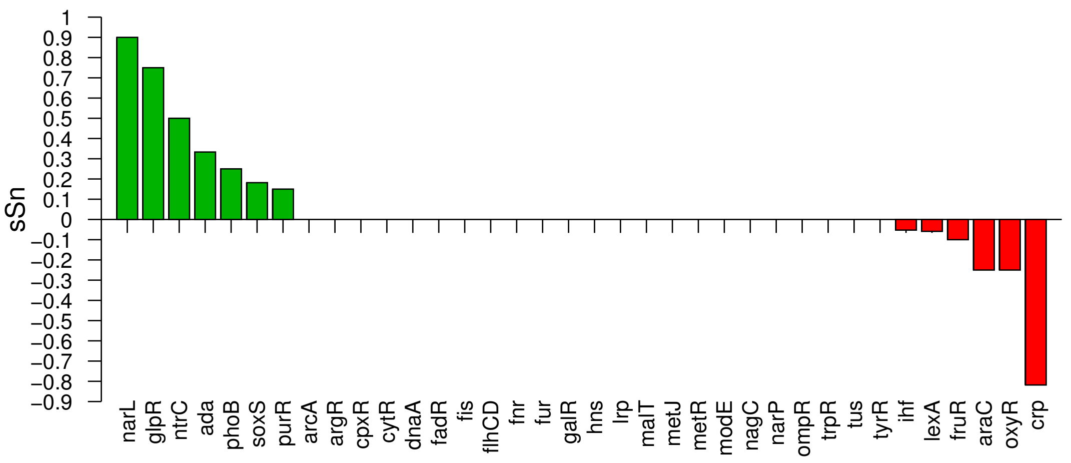


(a) Difference in site sensitivity between LP/DEE and MEME using a significance threshold.

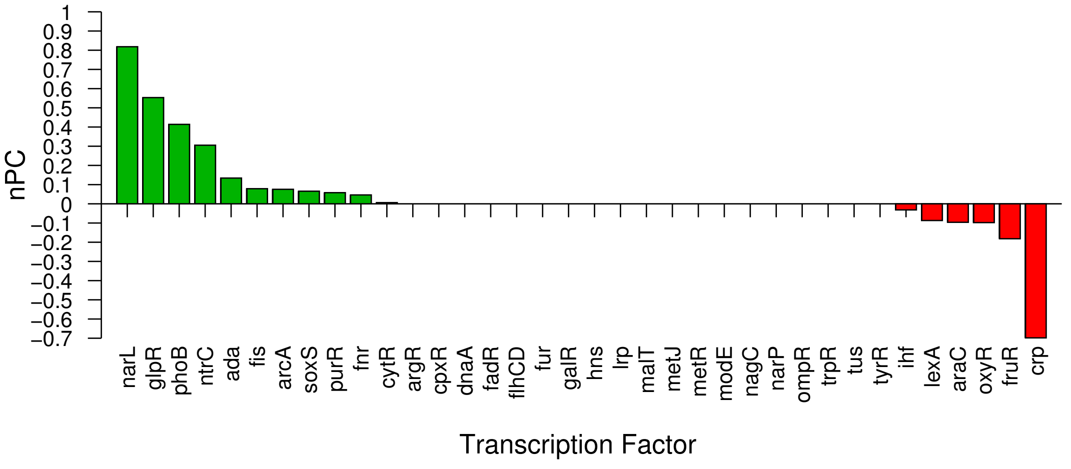


(b) Difference in nucleotide performance coefficient between LP/DEE and MEME using a significance threshold.

Figure 3.4: Performance comparison between the LP/DEE method and MEME [Bailey and Elkan 1995] when identifying known regulatory sites [Robison *et al.* 1998], using a significance threshold. For every transcription factor dataset the height of the bar indicates the difference in the metric, with green bars specifying better performance for our method and red bars otherwise. When a method fails to report a statistically significant motif with an e-value cutoff of 1.0, the number of correctly identified nucleotides and sites is set to zero.



(a) Difference in site sensitivity between LP/DEE and MEME ignoring motif significance.



(b) Difference in nucleotide performance coefficient between LP/DEE and MEME ignoring motif significance.

Figure 3.5: Performance comparison between the LP/DEE method and MEME [Bailey and Elkan 1995] when identifying known regulatory sites [Robison *et al.* 1998], ignoring motif significance. For every transcription factor dataset the height of the bar indicates the difference in the metric, with green bars specifying better performance for our method and red bars otherwise.

four transcription factors, with our method identifying *narL*, *glpR*, and *ntrC* almost completely, and MEME entirely misidentifying the motifs, while the opposite holds true for *crp*. For the rest of the datasets, the methods perform similarly. Average performance values for MEME are slightly lower than for LP/DEE, at average *nPC* being 0.360 and average *sSn* being 0.501 for MEME, whereas the corresponding values[5] for LP/DEE are 0.398 and 0.544. Ignoring e-values, of course, is not the correct approach to evaluating the performance of a method; true performance of MEME, thus, lies somewhere between the two extremes of disragrding e-values, on the one hand, and considering a stringent e-value threshold, on the other.

Lastly, we mention that for some datasets (e.g., *narP*), exhibiting zero or near-zero overlap with the known motif for all three methods, the Gibbs, MEME and LP/DEE algorithms identically agree on the choice of the discovered motif, and such a motif is typically highly conserved. This may be an indication that other, yet unknown, biologically relevant motifs exist in the data.

3.6.3 Phylogenetic footprinting

We experiment with motif discovery among sets of upstream regions of orthologous genes in a number of genomes, having incorporated phylogenetic information in constructing our graphs. The phylogenetic trees, the sequence datasets, varied in size and genome selection, and the motif length parameters come from [Blanchette and Tompa 2002]. All eight datasets contain vertebrate sequences; some (Interleukin-3 and Insulin datasets) consist of only mammalian genomes, while others contain members from more diverse animal phyla. The number of sequences in the datasets ranges between 4 and 16, and most sequences are shorter than 1000 residues in length.

We identify well-conserved interesting motifs in every dataset. We note that the discovered motifs are deemed to be statistically significant by our evaluation (e-values

[5]We now include *oxyr* and *ihf*, the two datasets with no significant motif that exhibited non-zero overlap values for LP/DEE, in the averaging computation.

lie in the range of 10^{-18} to 10^{-5}). Though the notion of significance according to our method merely rejects the hypothesis that all the motif instances are unrelated, and a scheme that takes phylogeny into account such as [Prakash and Tompa 2005] is better suited for this problem in general, our significance evaluation attests to the presence of a highly conserved motif instance in every input sequence.

The consensus sequences for the discovered motifs are listed in Table 3.5 along with the description of their DNA regions and source species[6]. All the motifs we find have been documented in the TRANSFAC database [Wingender *et al.* 1996], and the majority of them correspond to those that have been reported by [Blanchette and Tompa 2002]. Two motifs differ from those of [Blanchette and Tompa 2002]: the first, a *c-fos* motif, shares its consensus sequence with a known *c-fos* regulatory element, the binding site of the serum response factor (SRF) protein (accession number R02246). The second, a *c-myc* motif, also corresponds to a known *c-myc* binding site in the *P1* promoter (accession number R04621).

This dataset is also an excellent testing ground for finding distinct multiple motifs using our method, as such motifs exist and have been reported in previous studies. We iteratively identify motifs and remove their corresponding vertices from the constructed graphs. As proof of principle, we find multiple motifs for the insulin dataset. In this case, we successfully identify all four motifs reported by [Blanchette and Tompa 2002]. Since our objective function differs from that of [Blanchette and Tompa 2002] and we require motif occurrences in every input sequence, we recover the motifs in a different order. Of coarse, we identify numerous shifts of these motifs in successive iterations. In practice, therefore, it may be more beneficial to remove a number of vertices corresponding to subsequences overlapping the optimal solution before attempting to find the next motif.

[6]Motif reported for the C-fos promoter dataset was discovered second, after having discarded the poly-A repeat region.

DNA region	**Species**	**Motif (id)**
Growth-hormone 5' UTR + promoter (380 bp)	Salmon, trout, white fish, seriola, lates, tilapia, fugu, grass carp, catfish, chicken, rat, mouse, dog, sheep, goat, human	`TATAAAAA` (7)
Histone H1 5' UTR + promoter (650 bp)	Chicken, duck, frog, mouse	`AAACAAAAGT` (2)
C-fos 5' UTR + promoter (800 bp)	Tetraodon, chicken, mouse, hamster, pig, human	`CCATATTAGG`
C-fos first intron (376 to 758 bp)	Fugu, tetraodon, chicken, pig, mouse, hamster, human	`AGGGATATTT` (3)
Interleukin-3 5' UTR + promoter (490 bp)	Rat, mouse, cow, sheep, human, macaca	`TGGAGGTTCC` (3)
C-myc second intron (971 to 1376 bp)	Chicken, pig, rat, marmoset, gibbon, human	`TTTGCAGCTA` (5)
C-myc 5' promoter (1000 bp)	Goldfish, frog, chicken, rat, pig, marmoset, human	`GCCCCTCCCG`
Insulin family 5' promoter (500 bp)	Human, chimp, aotus, pig, rat (I, II), mouse (I, II)	`GCCATCTGCC` (2) `TAAGACTCTA` (1) `CTATAAAGCC` (3) `CAGGGAAATG` (4)

Table 3.5: Motifs identified with use of phylogenetic information. All datasets tested are from [Blanchette and Tompa 2002]. ***DNA region*** details the DNA regions considered; ***Species*** lists the species and isoforms considered; ***Motif (id)*** identifies the consensus sequence of the discovered motif and its correspondence with the motifs of [Blanchette and Tompa 2002] where applicable. All listed motifs have been documented as regulatory elements in TRANSFAC [Wingender *et al.* 1996]. For datasets other than the *insulin* dataset only the best motif is reported, and for the *insulin* dataset multiple motifs are reported in order of discovery.

3.6.4 Subtle motifs

We test our algorithm's performance in finding subtle motifs on synthetically generated data. Following the terminology of [Pevzner and Sze 2000], we produce the problem instances according to the *FM* (fixed mutation) model. For the (l, d)-signal finding problem we first randomly select a pattern M of length ℓ. For each of t background sequences ($t = 20$ for all test cases) exactly d random positions are chosen in M, and the pattern is then implanted with each of these d positions mutated to a different, randomly chosen base. Both the background sequence and the position of the implanted pattern instance are selected randomly. All random choices above are independent and drawn uniformly.

The original challenge problem proposed by [Pevzner and Sze 2000] is the $(15, 4)$ motif finding problem with background sequence length 600. Various algorithms [Buhler and Tompa 2002,Keich and Pevzner 2002,Eskin and Pevzner 2002,Price *et al.* 2003] have extended the length to 1500 and beyond and considered other combinations of the ℓ and d parameters. Previous approaches have reported the performance coefficient nPC defined earlier, averaged over several generated problem instances (denoted aPC) for every set of parameters considered. As reported by [Price *et al.* 2003] the above algorithms all perform with nearly 100% success rate when recovering the implanted pattern, and we include a representative method, *Projection* by [Buhler and Tompa 2002] in our comparison charts.

The performance of various methods in terms of their aPC for background sequence lengths N ranging from 100 to 1000 in presented in Table 3.6. Most of that data was collected and summarized by [Pevzner and Sze 2000], with each entry averaged over eight problem instances. Our method compares favorably to all the listed general purpose motif finders as well as specialized subtle motif finders. Its performance is matched only by Winnower (k = 3), which does so at the cost of higher complexity. We were further able to test our algorithm's ability to handle increasingly

Sequence Length	100	200	300	400	500	600	700	800	900	1000
CONSENSUS	0.92	0.94	0.53	0.31	0.29	0.07	0.15	0.09	0.01	0.04
GibbsDNA	0.93	0.96	0.51	0.46	0.29	0.12	0.09	0.34	0.00	0.12
MEME	0.91	0.78	0.59	0.37	0.17	0.10	0.02	0.03	0.00	0.00
WINNOWER(2)	0.98	0.98	0.97	0.95	0.97	0.92	0.58	0.02	0.02	0.02
WINNOWER(3)	0.98	0.98	0.97	0.94	0.97	0.92	0.90	0.93	0.90	0.88
SP-STAR	0.98	0.98	1.00	0.96	0.96	0.84	0.83	0.69	0.64	0.23
LP/DEE	0.99	0.96	0.98	0.96	0.93	0.96	0.88	0.90	0.89	0.91

Table 3.6: Comparison of performance of the various algorithms in terms of the *aPC* for different length samples with implanted $(15, 4)$ motif, reprinted from [Pevzner and Sze 2000] for all earlier methods. WINNOWER is listed along with its k parameter in parentheses. LP/DEE method is averaged over twenty instances. All other algorithms are averaged over eight instances.

noisier problems by implanting the motif in larger amounts of background sequence. Of the algorithms listed in Table 3.6, only Winnower ($k = 3$) was able to find motifs in sequences up to length 1300, and as stated in [Keich and Pevzner 2002] became "very time-consuming and unusable" beyond that. We carry out the same *aPC* analysis for N ranging from 1000 to 1500, and find that performance degrades very slowly from *aPC* of about 0.9 to 0.8, and roughly corresponds to the results reported by [Buhler and Tompa 2002] for *Projection*. To recall, dead-end elimination was used to reduce the sizes of the graphs. We found that *connectivity DEE* alone was sufficient to solve problems up to $N = 600$. Beyond that, we employed *connectivity DEE* in conjunction with *graph decomposition II*, where vertices were tested for potential membership in the optimal solution one at a time. Of the possibly many resulting subgraphs per problem instance, typically only one or two were output.

We also considered other combinations of the ℓ and d parameters in this signal finding problem for background sequence length $N = 600$. Table 3.7 lists our findings together with those of the earlier algorithms (reprinted from [Buhler and Tompa 2002]). Just as above, our method performs similarly to Projection, and outperforms the others. It is interesting to note the varying connectivity properties of the graphs. As the fraction of mutated positions to total motif length increased, the number of connected subgraphs, resulting from graph decomposition, grew as well.

l	d	GibbsDNA	WINNOWER	SP-STAR	PROJECTION	**LP/DEE**
10	2	0.20	0.78	0.56	0.80 ± 0.02	0.85
11	2	0.68	0.90	0.84	0.94 ± 0.01	0.91
12	3	0.03	0.75	0.33	0.77 ± 0.03	0.80
13	3	0.60	0.92	0.92	0.94 ± 0.01	0.91
14	4	0.02	0.02	0.20	0.71 ± 0.05	0.68
15	4	0.19	0.92	0.73	0.93 ± 0.01	0.96
16	5	0.02	0.03	0.04	0.67 ± 0.06	0.66
17	5	0.28	0.03	0.69	0.94 ± 0.01	0.89
19	6	0.05	0.03	0.40	0.94 ± 0.01	0.91

Table 3.7: Comparison of performance of the various algorithms in terms of the aPC on instances with background length $N = 600$ and various combinations of ℓ and d parameters, reprinted from [Buhler and Tompa 2002] for all earlier methods. WINNOWER is used with $k = 2$.

Whereas one or two subgraphs remained for most problems, that number became 4–16 for the $(19, 6)$-signal and hundreds for the $(16, 5)$-signal, as second and higher levels of *graph decomposition* were employed.

We note that failures of any of these algorithms are due to multiple cliques in the graph. In contrast to local search techniques, our algorithm is not subject to finding suboptimal local maxima as judged by the sum-of-pairs score. Lack of direct correspondence between implanted motifs and highest weight cliques suggests considering a number of near-optimal solutions. We use the constraint in Equation 3.9 to find up to the first 1000 heaviest weight cliques, if that many exist, by enumerating near-optimal solutions. In each case, one of the cliques output corresponds to the actual implanted pattern. Since edges between vertices are weighted by the total number of matches between the corresponding subsequences, the implanted motif typically corresponds to one of the higher scoring ones. For the simpler case of the $(15, 4)$ motif, the implanted pattern is often found among the optimal-weight cliques. However, in some of the $(14, 4)$ and $(12, 3)$ cases, there appear to be many higher weight cliques occurring by chance, suggesting that these patterns are too hard to distinguish from background. Nevertheless, our approach of finding multiple, successive near-optimal solutions allows us to retrieve all valid possible implanted patterns.

3.7 Discussion

We have described a versatile mathematical programming framework for the motif finding problem. In order to solve the linear programs, numbering sometimes in millions of variables, we employ graph decomposition and pruning techniques to identify vertices guaranteed to be excluded from the optimal solution. While these algorithms tremendously reduce the sizes of the problems, some datasets are more challenging and computationally expensive than others. Clearly, the difficulty is correlated with the total number of vertices in the graphs (see Table 3.1), and the average sequence length appears to be the dominant factor, as has also been found by [Hu *et al.* 2005]. It is also noteworthy that presence of a well conserved motif allows the pruning to be especially effective. For example, the human zinc metallopeptidase dataset contains a very highly conserved motif, a well-recognized zinc-binding region signature; albeit with application of computationally involved decomposition and pruning techniques, its reduced graph size is ten vertices, a mere one vertex per graph part.

A major advantage of our algorithm over previous approaches for motif finding is the ability to find optimal local alignments for many practical problems. In the future, we hope to extend the capabilities of our approach by incorporating features common to more widely-used motif finding algorithms. A basic improvement would be to automatically decide on motif length. It can be done using a standard method, such as choosing the length that corresponds to the highest average information content (or relative entropy) in the motif. Alternatively, the relative success of the pruning procedure can be a gauge, as the inability to eliminate many vertices can indicate potential absence of a motif. For example, when pruning the immunoglobulin fold dataset (see Table 3.1) with the length parameter equal to 10, the graph is highly reduced in size (from 7498 to 187 vertices); when attempting to prune it with the length parameter set to 14, no such substantial reduction occurs and the graph remains large at 2322 vertices. Another useful feature is to allow zero occurrences of a motif in some of the input sequences; this may be possible to accomplish within our framework by including an additive term in the objective function that creates a tradeoff between the weight of the induced clique-like structure that is being maximized and the potential motif absence. It is also interesting to investigate other types of useful constraints that can be included in our linear programs. For example, we may want to look for two shorter motifs that are within some distance of one another, modeling cooperative binding of transcription factor proteins; there are natural linear constraints that can enforce such a condition.

In summary, the described optimization framework provides a flexible approach to tackle many important issues in motif finding. We have successfully applied it to a variety of problems, including DNA motifs, protein motifs, and subtle motifs, and have been able to incorporate phylogenetic information in the context of cross-species motif discovery, as well as to find multiple near-optimal solutions. We hope in the future to extend its capabilities to model more complex types of motif finding problems.

Chapter 4

Improving a Mathematical Programming Formulation for Motif Finding

4.1 Introduction

Chapter 3 introduced an integer linear programming (ILP) formulation for the motif finding problem. A difficulty, encountered in solving real biological samples, is the size of the integer linear programs that can have millions of variables. While in the previous chapter we tackle such potentially large ILPs by preprocessing the graph with pruning and decomposition techniques, here we take an alternate direction and propose a novel, more compact integer linear program.

As in Chapter 3, we formulate the motif discovery problem as that of finding the best gapless local multiple sequence alignment using the sum-of-pairs (SP) scoring scheme based on a similarity measure. Here, we switch to an equivalent minimization formulation for the combinatorial problem of finding an optimum weight clique of size p in a p-partite graph (e.g., [Reinert *et al.* 1997, Pevzner and Sze 2000, Sze *et al.* 2004]), as we focus on a distance metric for the edge weights. For general notions of distance, this graph problem is NP-hard to approximate within any reasonable factor [Chazelle *et al.* 2004]. In the motif finding setting, where edge weights obey the triangle inequality, the problem remains NP-hard [Akutsu *et al.* 2000], yet amenable to approximation. And, while constant-factor approximation algorithms do exist [Gusfield 1993, Bafna *et al.* 1997], the ability to find the optimal solution in practice is preferable.

Our novel integer programming formulation utilizes the discrete nature of the distance metric imposed on pairs of subsequences. We present a class of constraints to make the linear programming relaxation of the new formulation provably as tight as that given in the previous chapter. Furthermore, rather than introducing all of these additional constraints to find a solution in practice, we provide a separation algorithm. We also describe and test a heuristic approach to solve the LP relaxation of our novel ILP formulation that, in all observed cases, finds a solution of the same objective value as the LP relaxation of the previous ILP, often an order of magnitude faster. Moreover, we observe that in fact, the LP relaxations for both of the ILP formulations often obtain integral optimal solutions, making solving the LP relaxations sufficient for solving the original ILP. Even if this were not the case, the ability to find faster solutions to the relaxations may translate into significant speed-ups in branch-and-bound approaches for ILP solving.

4.2 Formal Problem Specification

We are given p sequences, which are assumed without loss of generality to each have length N', and a motif length ℓ. The goal is to find a substring s_i of length l in each sequence i, such that the sum of the pairwise distances between the substrings (i.e., $\sum_{i<j} distance(s_i, s_j)$) is minimized. The distance between substrings may be defined

in several ways. The simplest measure, and the one we restrict ourselves to in this chapter is the Hamming distance.

Recall that we recast the motif finding problem into a graph problem. We define a complete, weighted p-partite graph, with a part V_i for each sequence. In V_i, there is a node for every possible window of length ℓ in sequence i. Thus there are $N := N' - \ell + 1$ nodes in each V_i, and the vertex set $V = V_1 \cup \cdots \cup V_p$ has size Np. For every pair u and v in different parts there is an edge $(u, v) \in E$. Letting *seq*(u) denote the subsequence corresponding to node u, the weight w_{uv} on edge (u, v) equals *distance*(*seq*(u), *seq*(v)). The goal in motif finding is to choose a node from each part so as to minimize the weight of the induced subgraph.

4.3 Integer and Linear Programming Formulations

4.3.1 Original integer linear programming formulation

In the previous chapter we introduced an integer linear programming (ILP) formulation for the motif finding problem. In this ILP formulation, there is a binary variable x_u for every node u in the above defined graph. The variable x_u is set to 1 if node u is chosen as part of the optimal solution, and 0 otherwise. Additionally, there is a variable y_{uv} for each edge in the graph (the edges are undirected and hence y_{uv} is the same as y_{vu}). These edge variables are set to 1 if both vertices incident on the edge are chosen. In the integer programming setting all variables are constrained to take values from $\{0, 1\}$. We repeat the ILP formulation from Chapter 3 for ease of reference:

$$\begin{array}{lll}
\text{Minimize} & \sum_{\{u,v\}\in E} w_{uv} \cdot y_{uv} & \\
\text{subject to} & & \\
\quad \sum_{u\in V_i} x_u = 1 & \text{for } 1 \leq i \leq p & \\
\quad \sum_{u\in V_i} y_{uv} = x_v & \text{for } 1 \leq i \leq p, v \in V \setminus V_j & \\
\quad x_u, y_{uv} \in \{0,1\} & \text{for } u \in V, (u,v) \in E &
\end{array} \qquad \text{(IP1)}$$

The first set of constraints ensures that one node is chosen from each part, and the second set requires that an edge is chosen if its end points are.

4.3.2 New integer linear programming formulation

Since the alphabet and the length of the sequences are finite, there are only a finite number of possible pairwise distances. For example, in the case of Hamming distances, edge weights can only take on $\ell + 1$ different values. We take advantage of the small number of possible weights and the fact that the edge variables of IP1 are only used to ensure that if two nodes u and v are chosen in the optimal solution then w_{uv} is added to the cost of the clique. We introduce a second ILP in which we no longer have edge variables X_{uv}. Instead, in addition to the node variables X_u, we have a variable Y_{ujc} for each node u, each graph part j such that u is not in V_j, and each possible edge weight c. These Y variables model groupings of the edges by cost into *cost bins*. The intuition is that Y_{ujc} is 1 if node u and some node $v \in V_j$ are chosen such that $w_{uv} = c$.

Formally, let D be the set of possible edge weights and let $W = \{(u, j, c) : c \in D, u \in V, j \in 1, \ldots p \text{ and } u \notin V_j\}$ be the set of triples over which the Y_{ujc} variables are indexed. Then the following ILP models the motif-finding graph problem:

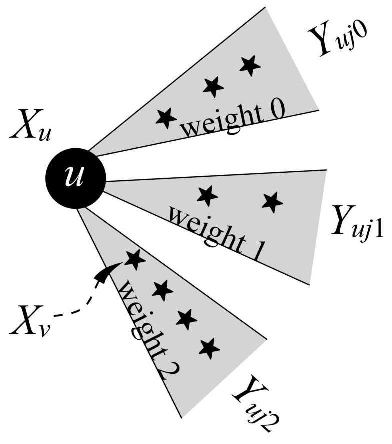


Figure 4.1: Schematic of IP2. Adjacent to each node u there are at most $|D|$ cost bins, each associated with a variable Y_{ujc}. Associated with each cost c are the nodes $v \in V_j$ for which $w_{uv} = c$ (represented in the figure by stars). Constraints (IP2b) say that we must spread a total of X_u weight apportioned over the bins, while constraints (IP2c) limit us to choosing cost bin variables where there is some node $v \in V_j$ chosen such that $w_{uv} = c$.

$$\begin{array}{lll}
\text{Minimize} & \sum_{(u,j,c)\in W:\text{part}(u)<j} c \cdot Y_{ujc} & \\
\text{subject to} & & \\
\quad \sum_{u \in V_i} X_u = 1 & \text{for } 1 \le i \le p & \text{(IP2a)} \\
\quad \sum_{c \in D} Y_{ujc} = X_u & \text{for } 1 \le j \le p, u \in V \setminus V_j & \text{(IP2b)} \\
\quad \sum_{v \in V_j : w_{uv} = c} Y_{vic} \ge Y_{ujc} & \text{for } (u,j,c) \in W \text{ s.t. part}(u) < j & \text{(IP2c)} \\
\quad X_u, Y_{ujc} \in \{0,1\} & &
\end{array} \qquad \text{(IP2)}$$

As in the previous formulation, the first set of constraints forces a single node to be chosen in each part. The second set of constraints says that if a node u is chosen, for each j, one of its "adjacent" cost bins must also be chosen (Figure 4.1). The third set of constraints ensures that Y_{ujc} can be chosen only if some node $v \in V_j$ is also chosen such that $w_{uv} = c$. We discard variables Y_{ujc} if there is no $v \in V_j$ such that $w_{uv} = c$. Figure 4.1 gives a schematic drawing of these constraints.

It is straightforward to see that IP2 correctly models the motif-finding problem if the variables are $\in \{0, 1\}$. For any choice of p-clique $\{u_1, \ldots, u_p\}$ of weight $\gamma = \sum_{i<j} w_{u_i u_j}$, a solution of cost γ to IP2 can be found by taking $X_{u_i} = 1$ for $i = 1 \ldots, p$, and taking $Y_{u_i jc} = 1$ for all $1 \le j \le p$ such that $w_{u_i u_j} = c$. This solution is easily seen to be feasible, and between any pair of graph parts i, j it contributes cost $w_{u_i u_j}$; therefore, the total cost is γ. On the other hand, consider any solution (X, Y) to IP2 of objective value γ. Consider the clique formed by the nodes u such that $X_u = 1$. Between every two graph parts $i < j$, the constraints (IP2a) and (IP2b) imply that exactly one Y_{ujc} and one Y_{vid} are set to 1 for some $u \in V_i$ and $v \in V_j$ and costs c, d. Constraint (IP2c) corresponding to (u, j, c) with Y_{ujc} on its right-hand side can only be satisfied if the sum on its left-hand side is 1, which implies $c = d = w_{uv}$. Thus, a clique of weight γ exists in the motif-finding graph problem.

4.3.3 Advantages of new IP formulation

In practice, IP2 has many fewer variables than IP1. Letting $d = |D|$, the number of kinds of weights, IP2 has $Np((p-1)d + 1)$ variables in the case that a Y_{ujc} variable exists for every allowed choice of (u, j, c), while IP1 has $Np(N(p-1)/2+1)$ variables. If $d < N/2$, the second IP will have fewer variables. In practice, d is expected to be much smaller than N, and while N could reasonably be expected to grow large as longer and longer sequences become practical to study, d is constrained by the geometry of transcription factor binding and will remain small. Also, in practice, it is likely that many Y_{ujc} variables are removed because *seq*(u) does not have matches of every possible weight in each of the other sequences. IP2, on the other hand, will have $O(d)$ times more constraints than IP1, with the number of constraints being $p + Np(p-1)(d/2 + 1)$ for IP2, and $p + Np(p-1)$ for IP1.

While the space requirement for the simplex algorithm is related to the number of constraints and variables, running time is not necessarily directly related. Smaller integer programs with weaker LP relaxations are often less useful for branch-and-bound approaches to IP solving. Thus, we seek the tightest, smallest IP possible. In the end, experiments must be performed to gauge the efficacy of various formulations on practical problems. We present experiments below which suggest IP2 can be more than an order of magnitude faster than IP1.

The recasting of the problem such that the number of edge weights becomes the determining factor of problem size suggests several avenues for practical performance enhancements. For example, it is often the case that one is interested in an optimal solution only if it is of high enough quality, meaning that no motif instance in the solution is further than α away from any other (the *diameter* of the solution is $\leq \alpha$). If this is the case, edges of weight $> \alpha$ can be deleted. Such a requirement reduces d and makes IP2 still smaller. In many applications, even if large diameter solutions are acceptable, there is an expectation that the diameter is likely small, and a solution with a small diameter may be preferred to one with a lower sum-of-pairs score but more outliers. In such a case, the IP2 formulation may allow one to check for low diameter solutions quickly.

4.3.4 Linear programming relaxation

The typical approach to solving an ILP is to solve the linear program derived from the ILP by dropping the requirement that the variables be in $\{0, 1\}$, and instead requiring only that the variables lie in the continuous range $[0, 1]$. This modified problem is called the linear programming (LP) relaxation. Efficient algorithms are known for solving linear programs. In the case of minimization, an ILP formulation is *weaker* than another if the corresponding LP relaxation of the first admits a solution of lower objective value than is possible with the second (*stronger* formulation is defined accordingly). Weaker relaxations are often less useful in solving the corresponding ILP.

The LP relaxation of IP1, which we refer to as LP1, is stronger than the LP relaxation of IP2 as stated. In this section, we present a fairly natural (though exponential) class of constraints that, if added to the LP relaxation of IP2, makes the two formulations equivalent. We refer to this fully constrained relaxation of IP2 as LP2. In the subsequent sections, we provide a separation algorithm and also show that in practice we can focus on just two types of constraints in LP2, and we are able to solve the original ILP iteratively by adding cutting planes corresponding to violated constraints of these types.

Additional constraints. Focus on a single pair of graph parts i and j. The bipartite graph that exists between V_i and V_j is explicitly modeled in IP1 by the edge variables. In IP2, however, the bipartite graph is only implicitly modeled by an understanding of which Y variables are compatible to be chosen together. We study this implicit representation by considering the bipartite *compatibility* graph $\mathcal{C}_{ij}$ between two parts i and j. Intuitively, we have a node in this compatibility graph for each Y_{ujc} and Y_{vic}, and there is an edge between the nodes corresponding to Y_{ujc} and Y_{vic} if $w_{uv} = c$. These two Y variables are compatible in that they can both be set to 1 in IP2.

More formally, let $\mathcal{C}_{ij} = (A_{ij}, A_{ji}, F)$ be a bipartite graph, where $A_{ij} = \{(u, j, c) : u \in V_i, c \in D\}$ is the set of indices of Y variables adjacent to a node in V_i, going to part j, and A_{ji} is defined analogously, going in the opposite direction. The edge set F is defined in terms of the neighbors of a triple (u, j, c). Let $\mathcal{N}(u, j, c) = \{(v, i, c) : u \in V_i, (v, i, c) \in A_{ji} \text{ and } w_{uv} = c\}$ be the *neighbors* of (u, j, c). They are the indices of the Y_{vic} variables adjacent to part j going to part i so that the edge $\{u, v\}$ has weight c. There is an edge in F going between (u, j, c) and each of its neighbors. We call c the *cost* of triple (u, j, c). All this notation is summarized in Figure 4.2.

In any feasible integral solution, if $Y_{ujc} = 1$, then some Y_{vic} for which $(v, i, c) \in \mathcal{N}(u, j, c)$ must also be 1. Extending this insight to subsets of the Y_{ujc} variables yields a class of constraints which will ensure that the resulting linear programming formulation is as tight as LP1. If Q_{ij} is a subset of A_{ij}, then let $\mathcal{N}(Q_{ij}) = \bigcup_{(u,j,c) \in Q_{ij}} \mathcal{N}(u, j, c)$

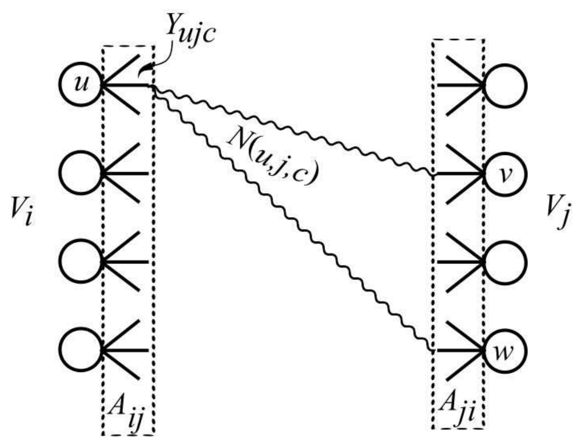


Figure 4.2: Compatibility graph $\mathcal{C}_{ij}$. The neighbor set $\mathcal{N}(u, j, c)$ is shown assuming that v and w are the only nodes in V_j that have cost c with u. Each circle represents a node in part V_i or V_j of the graph. The solid lines incident on the nodes represent Y_{ujc} or Y_{vic} variables associated with the node. A_{ij} and A_{ji} indicated by dotted lines are the sets of Y variables associated with the pair of graph parts i and j. Lastly, the function $N(u, j, c)$ maps a variable Y_{ujc} to a set of compatible Y_{vic} variables by squiggly lines.

be the set of indices that are neighbors to any vertex in Q_{ij}. If $Q_{ij} \subseteq A_{ij}$ then $\mathcal{N}(Q_{ij}) \subseteq A_{ji}$. The following constraint is true in IP2 for any such Q_{ij}:

$$\sum_{(u,j,c)\in Q_{ij}} Y_{ujc} \leq \sum_{(v,i,c)\in\mathcal{N}(Q_{ij})} Y_{vic}\,. \tag{4.1}$$

That is, choose any set of Y_{ujc} variables adjacent to part i that go to part j. The sum of the Y variables for their neighbors must be greater than or equal to the sum of the variables originally chosen. Notice that the third set of constraints in IP2 are of the form (4.1), taking Q_{ij} to be the singleton set $\{(u, i, c)\}$.

4.3.5 Equivalence of linear programming relaxations

Theorem 1 *If for every pair $i < j$, constraints of the form* (4.1) *are added to IP2 for each $Q \subseteq A_{ij}$ such that all triples in Q are of the same cost then the resulting LP relaxation LP2 is as strong as the relaxation LP1 of IP1.*

Proof. It is clear that the linear programming relaxation LP2 described in Theorem 1 is no stronger than LP1 as any solution to LP1 can be converted to a solution of LP2 by making the node variable weights the same and putting the weight of edge variables y_{uv} onto Y_{ujc} and Y_{vic}, where $w_{uv} = c$. This solution to LP2 will satisfy all the constraints in the theorem, and be of the same objective value.

The rest of the proof will involve showing that for any feasible solution for LP2, there is a feasible solution for LP1 with the same objective value, thereby demonstrating that the optimal solution to LP2 is not weaker than the optimal solution to LP1. In particular, fix a solution (X, Y) to LP2 with objective value γ. We need to show that for any feasible distribution of weights on the Y variables a solution to LP1 can be found with objective value γ.

In order to reconstruct a solution $\hat{x}$ for LP1 of objective value γ, we will set $\hat{x}_u = X_u$, using the values of the node variables X_u in the optimal solution to LP2. We must assign values to $\hat{y}_{uv}$ to complete the solution. Recall the compatibility graph $\mathcal{C}_{ij}$ described above. Because all edges in $\mathcal{C}_{ij}$ are between nodes of the same cost, $\mathcal{C}_{ij}$ is really $|D|$ disjoint bipartite graphs $\mathcal{C}_{ij}^c$, one for each cost. Let $A_{ij}^c \cup A_{ji}^c$ be the node set for the subgraph $\mathcal{C}_{ij}^c$ for cost c. Each edge in a subgraph $\mathcal{C}_{ij}^c$ corresponds to one edge in the graph G underlying LP1. Conversely, each edge in G corresponds to exactly one edge in one of the $\mathcal{C}_{ij}^c$ graphs (if edge $\{u, v\}$ has cost c_1, it corresponds to an edge in $\mathcal{C}_{ij}^{c_1}$). We will thus proceed by assigning values to the edges in the various $\mathcal{C}_{ij}^c$, and this will yield values for the $\hat{y}_{uv}$ variables.

If we define $z(A) := \sum_{(u,j,c)\in A} Y_{ujc}$, by the first two sets of constraints (IP2a and IP2b), $z(A_{ij}) = z(A_{ji}) = 1$. Since the constraints (4.1) are included with $Q = A_{ij}^c$ for each cost c, by the pigeonhole principle, $z(A_{ij}^c) = z(A_{ji}^c)$ for every cost c. Thus, for each subgraph $\mathcal{C}_{ij}^c$, the weight placed on the left graph part equals the weight placed on the right graph part. We will consider each induced subgraph $\mathcal{C}_{ij}^c$ separately.

We modify $\mathcal{C}_{ij}^c$ as follows to make it a directed, capacitated graph. Direct the edges of $\mathcal{C}_{ij}^c$ so that they go from A_{ji}^c to A_{ij}^c, and set the capacities of these edges

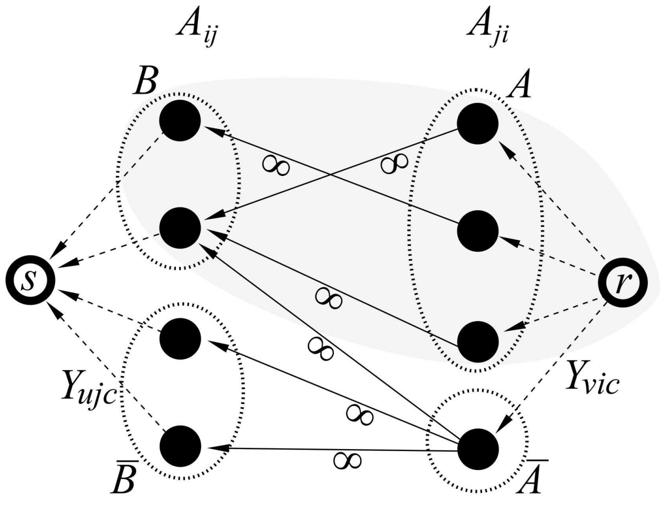


Figure 4.3: Directed, capacitated graph used to show (4.1) are sufficient. Edges incident on the source r and sink s have capacity equal to the Y variable adjacent to r and s respectively. All other edges are of infinite capacity. The shading gives an $r - s$ cut.

to be infinite. Add dummy source (r) and sink (s) nodes, and connect them to the graph by edges directed from r to each node in A^c_{ji} and edges from each node in A^c_{ij} to s. Every edge incident on r and s is also incident on some node representing a Y variable. Put capacities on these edges equal to the value of the Y variable to which they are adjacent (see Figure 4.3).

The desired solution to LP1 can be found if the weight of the nodes (Y variables) in each compatibility subgraph can be spread over the edges. In other words, a solution to LP1 of weight γ can be found if, for each c, there is a flow of weight $z(A^c_{ij})$ from r to s in the augmented flow graph $\mathcal{C}^c_{ij}$. The assignment to $\hat{y}_{uv}$ will be the flow crossing the corresponding edge in the $\mathcal{C}^c_{ij}$ of appropriate cost. Below we show that the set of constraints described in the theorem ensure that the minimum cut in the constructed graph is greater than or equal to $z(A^c_{ij})$, and thus there is a flow of the required weight. The proof of this fact in the context of flow feasibility problems can be found in [Cook *et al.* 1997] on page 54-55, and is reproduced in our notation in the following lemma.

Lemma 1 *The minimum cut of the augmented flow graph $\mathcal{C}^c_{ij}$ (shown in Figure 4.3) is $z(A^c_{ij})$.*

Proof. Recall that the capacities of the edges connecting nodes in A^c_{ji} with nodes in A^c_{ij} are infinite; capacities of the edges leaving r are Y_{vic} and those entering s are Y_{ujc}, and that the total capacity leaving r equals that of entering s, and this total capacity equals $z(A^c_{ij})$. We want to show that the minimum $r - s$ cut in this graph is greater than or equal to $z(A^c_{ij})$.

Consider an $r - s$ cut $\{r\} \cup A \cup B$ where $A \subseteq A^c_{ji}$ and $B \subseteq A^c_{ij}$. (Such a cut is shaded in Figure 4.3.) Define $\bar{A} = A_{ji} \setminus A$ and $\bar{B} = A_{ij} \setminus B$. If any edges connect A to $\bar{B}$ then the capacity of the cut is infinite, and the lemma is proven. Otherwise the value of the cut is the sum of the capacities of the edges leaving r and going to $\bar{A}$ plus the sum of the capacities of the edges entering s from a node in B. We will now show that

$$z(\bar{A}) \geq z(\bar{B}) \,, \tag{4.2}$$

which implies that the value of the cut is $> z(A^c_{ij})$.

Assume for the moment that all nodes in $\bar{A}$ have a neighbor in $\bar{B}$. Then $\mathcal{N}(\bar{B}) = \bar{A}$ because there are no edges between A and $\bar{B}$ by assumption. By (4.1), $z(\bar{B}) \leq z(\mathcal{N}(\bar{B})) = z(\bar{A})$. On the other hand, if there is a node in $\bar{A}$ that does not have a neighbor in $\bar{B}$ then we can add that node to A to make A' (without increasing the cost of the cut), and the above argument shows that $z(A_{ji} \setminus A') \geq y(\bar{B})$, which implies $z(\bar{A}) \geq z(\bar{B})$ since $A_{ji} \setminus A' \subseteq \bar{A}$. This ends the proof of the lemma. Thus, we have shown LP1 and LP2 to be equivalent. $\Box$

4.3.6 Separation algorithm and heuristic solution

It is possible to solve LP2, a linear program with an exponential number of constraints, in polynomial time by the ellipsoid algorithm [Grötschel*et al.* 1993] provided that a separation algorithm exists. In this section we describe such a separation algorithm, which finds a violated constraint, if one exists, in polynomial time or reports that no constraints are violated. The proof below formalizes the intuition in the proof of Theorem 1, by which all constraints are satisfied in a compatibility graph only if a large enough maximum flow exists. Otherwise, the minimum cut identifies a violated constraint.

Theorem 2 *There is a polynomial-time algorithm that can find a violated constraint in LP2 or report that no such constraint exists.*

Proof. Because each constraint in (4.1) involves variables of a single cost, if (4.1) is violated for some set Q, then Q is a subset of an A^c_{ji} for some i, j, c, and so we can consider each subgraph $\mathcal{C}^c_{ij}$ independently. The proof of Theorem 1 shows that there is a violated constraint of the form (4.1) between graph parts i and j involving variables of cost c if and only if the maximum flow in the augmented graph C^c_{ij} is less than $z(A^c_{ij})$. Thus, the minimum cut can be found for each triple i, j, c, and, if a triple i, j, c, with the minimum cut being less than $z(A^c_{ij})$, is identified, one knows that a violated constraint exists between parts i and j with $Q \subset A^c_{ij}$.

The minimum cut can then be examined to determine the violated constraint explicitly. Let $\{r\} \cup A \cup B$ be the minimum $r - s$ cut in C^c_{ij}, with $A \subseteq A^c_{ji}$ and $B \subseteq A^c_{ij}$, and let $\bar{A} = A^c_{ji} \setminus A$ and $\bar{B} = A^c_{ij} \setminus B$ (following the notation of Lemma 1). Such a cut is shaded in Figure 4.3. Let m be the capacity of this cut, and assume, because we are considering a triple i, j, c that was identified as having a violated constraint, that $z(A^c_{ij}) > m$. Because $m < \infty$ there are no edges going from A to $\bar{B}$, and hence two things hold: (1) $m = z(B) + z(\bar{A})$ and (2) $\mathcal{N}(A) \subseteq B$, and therefore $y(\mathcal{N}(A)) \leq y(B)$. Chaining these facts together, we have

$$z(A) = z(A^c_{ij}) - z(\bar{A}) > m - z(\bar{A}) = z(B) \geq z(N(A)),$$

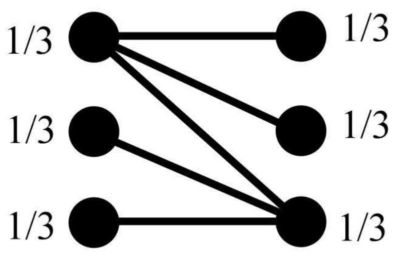


Figure 4.4: Example graph $\mathcal{C}^c_{ij}$, for which the added constraints are insufficient to make LP2 as tight as LP1. All the constraints with $|Q| = 1$ or $|Q| = |A_{ij}|$ are satisfied, but a flow of value 1 does not exist in the augmented version of this graph (as in Figure 4.3).

Thus, the set A is a set for which the constraint of the form (4.1) is violated. $\square$

In practice the ellipsoid algorithm is often slower than the well optimized simplex algorithm. We found that not all of the exponential number of constraints are necessary to solve real problems with the simplex algorithm. Some particular choices of Q_{ij} yield constraints that are intuitively very useful and are usually enough in practice. The constraints with the largest Q_{ij}, that is $Q_{ij} = A_{ij}$, were used in the proof of Theorem 1, and we have found them to be effective in practice. In fact, for this Q_{ij}, the inequality (4.1) is an equality. The relaxation of IP2 already includes all the constraints with $Q_{ij} = \{Y_{ujc}\} \subset A^c_{ij}$. Rather than including constraints with $1 < |Q_{ij}| < |A^c_{ij}|$, we include the constraints with i and j reversed, taking $Q_{ji} = \{Y_{vic}\} \subseteq A^c_{ji}$ for $v \in V_j$, which can be seen to be weaker versions of constraints (4.1) with larger Q_{ij} sets. More detail about our approach to and experiences with real problems can be found in Section 4.4.

Examples can be constructed for which the constraints we use are insufficient to make LP2 as tight as LP1. For instance, Figure 4.4 gives a graph $\mathcal{C}^c_{ij}$ for a single weight c for which all the constraints (4.1) hold but no solution to IP1 can be constructed. However, we have not encountered such pathological cases in practice, and so we do not explore adding constraints with $|Q| > 1$.

4.4 Computational Results

4.4.1 Methodology

We have found the following methodology to work well in practice. We first solve the LP relaxation of IP2, and stop if the solution is integral. Otherwise, we add any violated constraint of the form (4.1) with i and j reversed and with $|Q_{ji}| = 1$, and resolve. We use the optimal basis of the previous iteration as a starting point for the next, setting the dual variables for the added constraints to be basic. This strategy eliminates the need to resolve using an arbitrary starting solution and provides a significant speedup. In the rest of this section we refer to this heuristic strategy of solving the relaxation of IP2 as LP2.

Because the variant of the simplex algorithm can make a large difference in running time in practice, LP1 was solved using two different variants. In the first (`primal dualopt`), the primal problem was solved using the dual simplex algorithm. In the second (`dual primalopt`), the dual problem was solved using the primal simplex algorithm. While, in theory, these two variants should perform similarly, in practice running times can differ significantly. Applying the dual simplex method to the dual problem or the primal simplex to the primal problem are not expected to perform as well because of the relationship between the number of variables and number of constraints in the linear programs, and small scale testing confirms this intuition. LP2 was always solved using the dual simplex method applied to the primal problem so that resolving with additional constraints was faster.

The linear and integer programs were specified with Ampl and solved using CPLEX 7.1 [CPLEX 7.1]. All experiments were run on a public 1.2 GHz SPARC workstation shared by many researchers, using a single processor. All the timings reported are in CPU seconds on this machine. For any run, any problem taking longer than five hours was aborted.

4.4.2 Test datasets

We present results on identifying the binding sites of 50 transcription factor families. We construct our dataset from the data of [Robison *et al.* 1998, McGuire *et al.* 2000] in a fashion similar to the dataset of used in Chapter 3, but with generally shorter sequences. We remove all sites for sigma-factors, duplicate sites, as well as those that could not be unambiguously located in the genome. For each transcription factor under consideration, we extract the genes it is regulating, and gather at least 300 base pairs of genomic sequence upstream of the transcription start sites. In those cases where the binding site is located further upstream, we extend the sequence to include the binding site. The motif length for each family was chosen as before based on the length of the consensus binding site. Unlike in the previous chapter, we do not exclude datasets with fewer than three sequences. The families, their sizes and the length of the binding site are summarized in Table 4.1.

4.4.3 Performance of the LP relaxations

We solved LP1 and LP2 relaxations for the 50 transcription factors listed in Table 4.1. Results, summarizing running times, matrix sizes and speed-up factors are shown in Figure 4.5. For five problems, each LP failed to find a solution in the allotted five hours; these are omitted from the figure. Generally, the initial set of constraints was sufficient to get a tight solution, one that is at least as good as the solution for LP1. Six problems required additional constraints to LP2 in order to make it as tight as LP1. The problems flhCD, torR, and hu required two cutting planes iterations, ompR required three, oxyR required four, and nagC required five. Running times reported in Figure 4.5(b) are the sum of the initial solve times and of all the cutting planes iterations.

Of the problems solvable in less than five hours, only three were not integral. This is somewhat surprising. Of course, there is much structure to real problems, which

TF	ℓ	p	n	*TF*	ℓ	p	n	*TF*	ℓ	p	n
ada	31	3	810	fruR	16	11	4082	metR	15	8	3312
araC	48	6	1715	fur	18	9	3182	modE	24	3	934
arcA	15	13	4790	galR	16	7	2188	nagC	23	6	1870
argR	18	17	5960	gcvA	20	4	1234	narL	16	10	3301
carP	25	2	552	glpR	20	11	3829	ntrC	17	5	1516
cpxR	15	9	2614	hipB	30	4	1084	ompR	20	9	3057
cspA	20	4	1410	hns	11	5	1485	oxyR	39	4	1048
cynR	21	2	854	hu	16	2	571	pdhR	17	2	568
cysB	40	3	783	iclR	15	2	588	phoB	22	14	4618
cytR	18	5	1695	ilvY	27	2	1079	purR	26	20	5856
dnaA	15	8	2381	lacI	21	2	560	rhaS	50	2	502
fadR	17	7	2122	lexA	20	19	5554	soxS	35	13	4004
farR	10	3	873	lrp	25	14	4090	torR	10	4	2198
fhlA	27	2	731	malT	10	10	3410	trpR	24	4	1108
fis	35	18	5371	marR	24	2	813	tus	23	5	1390
flhCD	31	3	810	melR	18	2	717	tyrR	22	17	5258
fnr	22	12	3705	metJ	16	15	5754				

Table 4.1: Characteristics of the 50 problems considered. For each transcription factor dataset *TF*, listed are: motif length (ℓ), number of sequences (p), and the total number of vertices in the underlying graph (n).

may make them less susceptible to the worst-case analysis. The success of the LP relaxations in finding integral solutions suggests that handling non-integral cases may not be as pressing a problem as one would think.

When comparing the running times of LP2 with those of LP1, we take as the running time of LP1 that of the better performing simplex variant. In order words, we take the speed-up factor to be:

$$\frac{\min(\texttt{primal dualopt}, \texttt{dual primalopt})}{\text{LP2}} \tag{4.3}$$

Taking the minimum running time gives LP1 the benefit of the doubt. In practice, always achieving the runtime used in the numerator would require computing each variant in parallel using two processors. The speed-up factors for these problems are shown in Figure 4.5(a). For 10 problems, neither simplex variant completed in

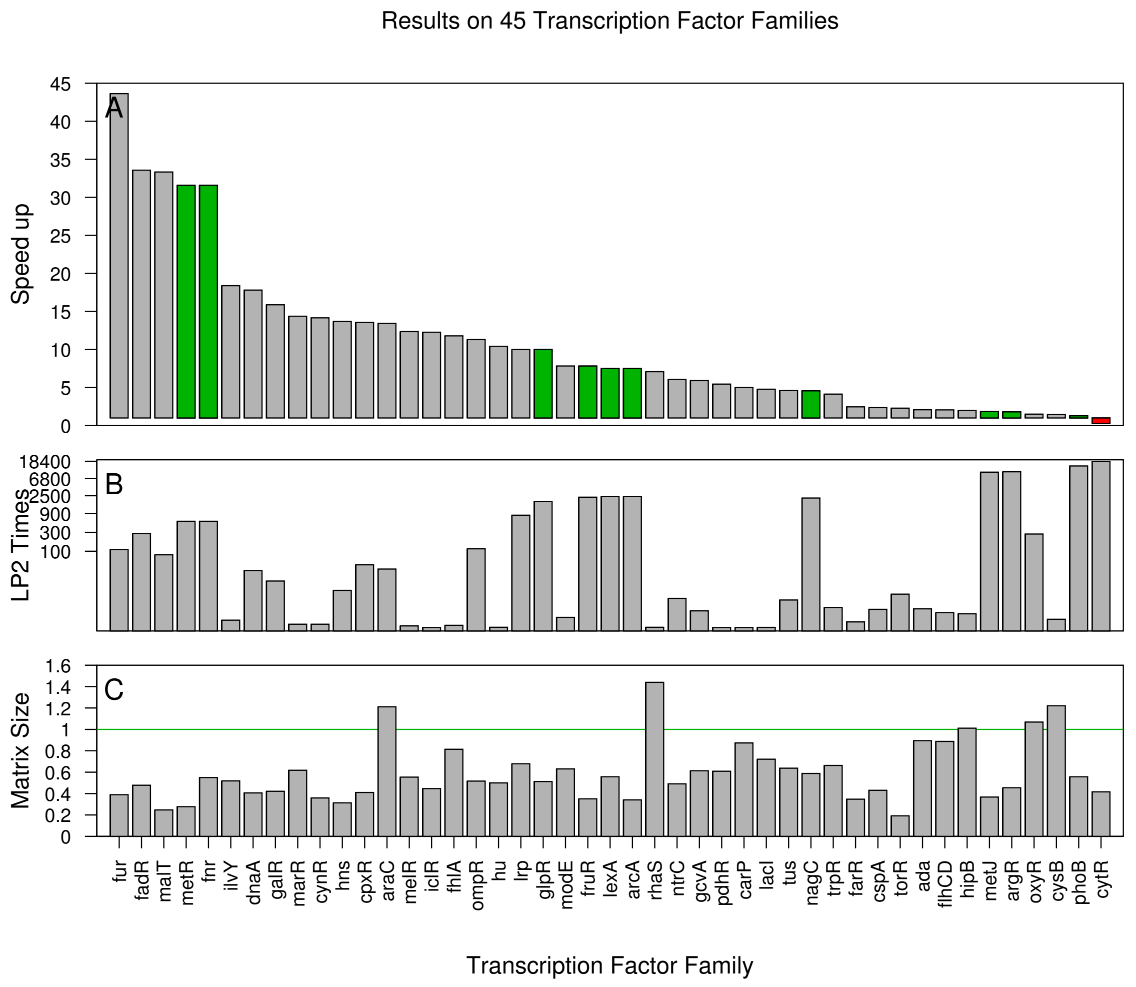


Figure 4.5: (a) Speed-up factor of LP2 over LP1 as defined in equation 4.3. Green bars correspond to problems for which LP1 did not finish in less than five hours. The red bar (far right) marks the problem for which LP2 did not finish in less than five hours, but LP1 did. (b) Running times in seconds for LP2. The y-axis is in log scale. (c) Matrix size for LP2 divided by the matrix size for LP1.

less than five hours when applied to LP1, whereas LP2 did. For these problems, the numerator of (4.3) was taken to be five hours. Such as setting gives a lower bound on the speed up. For one problem, cytR, the reverse was true and LP2 could not finish within five hours, while both simplex variants of LP1 successfully solved the problem. For this dataset, the denominator was taken to be five hours, and equation (4.3) gives an upper bound. For the rest of the datasets an order of magnitude increase in speed is common when using LP2 compared with LP1.

As expected, the size of the constraint matrix (defined as the number of constraints times the number of variables) is often smaller for LP2. Figure 4.5(c) plots (size for LP2)/(size for LP1). While in five cases the matrix for LP2 is larger, in many cases it is $< 50\%$ the size of the matrix for LP1. A smaller constraint matrix can often lead to faster simplex iterations.

We also compared the motifs found by our approach to the set of known transcription factor binding sites existing in the data. We identify motifs that do not always correspond to the known binding sites, but in all cases are at least as well conserved as the actual binding site motifs (measured by average information content). Since our test data are real genomic sequences, co-regulated genes may in fact have multiple shared binding sites for a number of transcription factors. The better conserved set of motif instances we identify may be the binding site signature of a different transcription factor. Alternatively, since the Hamming distance metric is oblivious to the nucleotide distribution in the background, some of our discovered motifs may be the very well conserved low-complexity regions.

4.5 Discussion

In this paper, we introduced a novel integer linear programming formulation for the motif finding problem that works well in practice. In particular, it is significantly faster in finding optimal solutions to moderate-sized motif finding instances than the previous ILP formulation introduced in Chapter 3. We note that a variety of graph pruning and decomposition techniques that have been introduced for motif finding in Chapter 3 and by other authors (e.g., [Reinert *et al.* 1997, Pevzner and Sze 2000, Lukashin and Rosa 1999]), are applicable here as well. It is likely that, in conjunction with those techniques, our novel formulation will be able to tackle problems of significantly larger sizes. Additionally, in contexts where every motif instance is required to be a good match to the motif consensus, our methodology can be applied in an iterative fashion, solving successive subproblems with increasingly higher allowed edge weights until a good solution is found.

There are many interesting avenues for future work. While the underlying graph problem is essentially identical to that of [Chazelle *et al.* 2004], where arbitrary weights are allowed on edges, one central difference is that when minimizing distance in the motif finding application based on nucleotide matches and mismatches, the triangle inequality is satisfied. The current ILP formulations do not exploit this, and as a result, works in its absence. Another feature commonly present in motif finding that is not used here is that the edge weights in the graph are not independent, as each node represents a subsequence from a window sliding along the DNA. Incorporating either the triangle inequality or the correlation between edge weights into the ILP or its analysis may lead to further advances in computational methods for motif finding.

Chapter 5

Conclusions and Future Work

This thesis provides a contribution towards solving a problem essential to the inner workings of a cell, as gene expression and regulation enable a cell to function properly, responding to various life cycle's demands and environmental circumstances. A vital first step in understanding the circuitry of the transcription regulatory network of an organism, with its complex operational patterns, is identification of transcription factor binding sites.

We addressed two subproblems in DNA binding site prediction: those of representation and discovery. In Chapter 2, we presented a comprehensive study of various binding site representation methods, evaluating their ability to identify additional binding sites of transcription factors when given a group of known sites. We note the benefit of incorporating information content into all scoring schemes, and pairwise correlations for some methods. Analysis similar to the one performed in Chapter 2 is likely to prove useful in choosing, for different contexts, a specific method and suitable threshold for finding binding sites of a particular known or unknown transcription factor.

In Chapters 3 and 4, we tackled the problem of motif discovery. We introduced two different mathematical programming formulations for the problem, and developed a flexible optimization framework, whose major advantage over other methods is in finding provably optimal solutions for most problems, and being able to incorporate additional relevant biological information such as protein substitution matrices and phylogenetic tree data. We also describe a procedure for statistical significance evaluation of our results, and find that motifs we discover are unlikely to have occurred in random data.

We have built a prototype system that implements our algorithm, and have successfully discovered known protein and DNA motifs in numerous datasets. We would like to scale up our system to run on larger numbers of longer sequences, overcoming computational efficiency difficulties, as well as to extend our method to incorporate other features common to motif finding algorithms and useful in the context of transcription factor binding sites' discovery. A few basic improvements include automated setting of the motif length parameter and the ability to allow zero or more occurrences of a motif in each of the input sequences. Another interesting capability would be to look for two motifs that are within some distance of one another, modeling cooperative binding of transcription factor proteins. These may be possible to implement in our framework through augmentations of the linear program by altering the objective function and adding linear constraints. Lastly, any motif finding system should be subjected to thorough testing on a diverse and large-scale dataset, such as that of [Tompa *et al.* 2005], and evaluated against many other motif finders under standardized conditions.

An alternate avenue for research, and one that would likely create a more computationally efficient motif finding method, able to tackle and solve to optimality larger sized problems, is to combine the graph pruning methods with the LP/ILP formulation of Chapter 4. Finally, the following constitutes an exciting open question for future research: why is it that integral solutions are observed so frequently for both linear programming formulations? We note that integral solutions often occur regardless of problem size. It would be interesting to explore the structure of the convex polytope enclosing the space of all integral solutions to answer this question.

In conclusion, we have introduced several new methods for transcription factor binding sites' representation and discovery, and established their merits with extensive and thorough testing. Our results have been promising, and suggest that our underlying methodology may be useful as a basis for a comprehensive system for identifying novel binding sites of transcription factor proteins.

Bibliography

[Akutsu *et al.* 2000] Akutsu, T., Arimura, H., and Shimozono, S. On approximation algorithms for local multiple alignment. In Proceedings of the Fourth Annual International Conference on Research in Computational Molecular Biology, pages 1–7. ACM Press, 2000.

[Althaus *et al.* 2000] Althaus,E., Kohlbacher,O., Lenhof,H.-P. and Müller,P. A combinatorial approach to protein docking with flexible sidechains. RECOMB, 2000, pp. 15–24.

[Altschul 2005] Altschul, S. Personal communication.

[Bafna *et al.* 1997] Bafna, V., Lawler, E., Pevzner P. A. Approximation algorithms for multiple alignment. Theoretical Computer Science, 182:233–244, 1997.

[Bailey and Elkan 1995] Bailey, T. and Elkan, C. 1995. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. Machine Learning, 21: 51–80.

[Barash *et al.* 2003] Barash, Y. and Elidan, G. and Friedman, N. and Kaplan, T. Modeling dependencies in protein-DNA binding sites. In: Proceedings Seventh Annual International Conference on Computational Molecular Biology. 28-37.

[Benos *et al.* 2002] Benos, P. and Bulyk, M. and Stormo, G. Additivity in protein-DNA interactions: how good an approximation is it? Nucl. Acids Res. 30(20):4442-4451.

[Berg and von Hippel 1987] Berg, O. and von Hippel, P. 1987. Selection of DNA binding sites by regulatory proteins. Statistical-mechanical theory and application to operators and promoters. J. Mol. Biol. 193: 723–750.

[Berg and von Hippel 1988] Berg, O. and von Hippel, P. 1988. Selection of DNA binding sites by regulatory proteins. II. The binding specificity of cyclic AMP receptor protein to recognition sites. J. Mol. Biol. 200: 709–723.

[Blanchette and Tompa 2002] Blanchette, M. and Tompa, M. 2002. Discovery of regulatory elements by a computational method for phylogenetic footprinting. Genome Res 12: 739–748.

[Blattner *et al.* 1997] Blattner, F., Plunkett G. 3rd, Bloch, C., Perna, N., Burland, V., Riley, M., Collado-Vides, J., Glasner, J., Rode, C., Mayhew, G. et al. 1997. The complete genome sequence of Escherichia coli K-12. Science 277: 1453–1474.

[Boeckmann *et al.* 2003] Boeckmann B., Bairoch A., Apweiler R., Blatter M.-C., Estreicher A., Gasteiger E., Martin M.J., Michoud K., O'Donovan C., Phan I., Pilbout S., Schneider M. 2003. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res. 31: 365–370.

[Brazma *et al.* 1998] Brazma, A., Jonassen, I., Eidhammer, I., and Gilbert, D. 1998. Approaches to the automatic discovery of patterns in biosequences. J. Comput Biol. 5(2): 279–305.

[Buhler and Tompa 2002] Buhler, J. and Tompa, M. 2002. Finding motifs using random projections. J. Comput Biol. 9(2): 225–242.

[Bulyk *et al.* 2002] Bulyk, M. L., Johnson P. L., Church, G. M. 2002. Nucleotides of transcription factor binding sites exert interdependent effects on the binding affinities of transcription factors. Nucl. Acids Res. 30(5): 1255-61.

[Carillo and Lipman 1988] Carillo, H. and Lipman, D. 1988. The multiple sequence alignment problem in biology. SIAM Journal on Applied Math. 48: 1073–1082.

[Chazelle *et al.* 2004] Chazelle, B., Kingsford, C., and Singh M. A semidefinite programming approach to side-chain positioning with new rounding strategies, INFORMS J. on Computing, 16:380–392, 2004.

[Cliften *et al.* 2003] Cliften, P., Sundarsanam P., Desikan, A., Fulton, L., Fulton, B., Majors, J. *et al.*. Finding functional features in Saccharomyces genomes by phylogenetic footprinting. 301: 71–76.

[CPLEX 7.1] ILOG CPLEX 7.1 http://www.cplex.com.

[Cook *et al.* 1997] Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A. 1997. Combinatorial Optimization. Wiley-Interscience, New York.

[Day and McMorris 1992] Day, W. H. and McMorris, F. 1992. Critical comparison of consensus methods for molecular sequences. Nucl. Acids Res. 20: 1093–1099.

[Dayhoff *et al.* 1978] Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C. 1978. A model of evolutionary change in proteins. In "Atlas of Protein Sequence and Structure" 5(3) M.O. Dayhoff (ed.), 345–352.

[Desmet *et al.* 1992] Desmet, J., De Maeyer, M., Hazes, B., Lasters, I. 1992. The dead-end elimination theorem and its use in protein side-chain positioning. Nature 356: 539–542.

[Eskin and Pevzner 2002] Eskin, E. and Pevzner, P. Finding composite regulatory patterns in DNA sequences. 2002. Bioinformatics (Supplement 1), 18: S354–S363.

[Feng and Doolittle 1987] Feng, D. and Doolittle, R. F. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol., 60: 351–360.

[Fourer *et al.* 2002] Fourer, R., Gay, D.M., and Kernighan, B.W. 2002. AMPL: A Modeling Language for Mathematical Programming. Brooks/Cole Publishing Company, Pacific Grove, CA.

[Frith *et al.* 2004] Frith, M.C., Hansen, U., Spouge, J.L. and Weng Z. 2004. Finding functional sequence elements by multiple local alignment. Nucleic Acids Res. 32(1): 189–200.

[Gelfand 1995] Gelfand, M. 1995. Prediction of function in DNA sequence analysis. J. Comput. Biol. 2: 87–115.

[Gelfand *et al.* 2000] Gelfand, M., Koonin, S. and Mironov, A. 2000. Prediction of transcription regulatory sites in Archaea by a comparative genomic approach. Nucl. Acids Res. 28: 695–705.

[Grötschel*et al.* 1993] Grötschel, M., Lovász, L., and Schrijver, A. 1993. Geometric Algorithms and Combinatorial Optimization. Springer-Verlag, Berlin, Germany, 2nd edition.

[Gusfield 1993] Gusfield, D. 1993. Efficient methods for multiple sequence alignment with guaranteed error bounds. Bull. Math. Biol. 55(1): 141–154.

[Henikoff and Henikoff 1992] Henikoff, S. and Henikoff, J. 1992. Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. USA. 89(biochemistry): 10915–10919.

[Hertz and Stormo 1999] Hertz, G. and Stormo, G. 1999. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. Bioinformatics 15: 563–577.

[Holm 1979] Holm, S. 1979. A Simple Sequentially Rejective Multiple Test Procedure. Scandinavian Journal of Statistics 6: 65–70.

[Hu *et al.* 2005] Hu, J., Li, B. and Kihara, D. 2005. Limitations and potentials of current motif discovery algorithms. Nucleic Acids Research 33(15): 4899-4913.

[Hughes *et al.* 2000] Hughes, J., Estep, P., Tavazoie, S. and Church, G. 2000. Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. J. Mol. Biol. 296: 1205–1214.

[Keich and Pevzner 2002] Keich, U. and Pevzner, P. 2002. Finding motifs in the twilight zone. Bioinformatics, 18: 1374–1381.

[Kellis *et al.* 2003] Kellis, M. Patterson, N., Endrizzi, M., Birren, B. and Lander E. Sequencing and comparison of yeast species to identify genes and regulatory elements. Nature, 423: 241–254.

[Kingsford *et al.* 2005] Kingsford, C.L., Chazelle, B. and Singh, M. 2005. Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics 21(7): 1028–1039.

[Lawrence *et al.* 1993] Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A., and Wootton, J. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. Science 262: 208–214.

[Lawrence and Reilly 1990] Lawrence, C. and Reilly, A. 1990. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. Proteins: Structure, Fuction, and Genetics, 7: 41–51.

[Lee *et al.* 2002] Lee, T.I., Rinaldi, N.J., Robert, F., Odom, D.T., Bar-Joseph, Z., Gerber, G.K., Hannett, N.M., Harbison, C.T., Thompson, C.M., Simon, I., Zeitlinger, J., Jennings, E.G., Murray, H.L., Gordon, D.B., Ren, B., Wyrick, J.J., Tagne, J.B., Volkert, T.L., Fraenkel, E., Gifford, D.K., Young, R.A. 2002. Science 298(5594): 799–804.

[Liu *et al.* 2001] Liu, X., Brutlag, D.L., Liu, J.S. 2001. BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. Proceedings of the Pacific Symposium on Biocomputing, pages 127–138.

[Lukashin and Rosa 1999] Lukashin, A. and Rosa, J. 1999. Local multiple sequence alignment using dead-end elimination. Bioinformatics 15: 947–953.

[Man and Stormo 2001] Man, T. K. and Stormo, G. D. 2001. Non-independence of Mnt repressor-operator interaction determined by a new quantitative multiple fluorescence relative affinity (QuMFRA) assay. Nucl. Acids Res. 29: 2471–2478.

[Marsan and Sagot 2000] Marsan, L., and Sagot, M. F. 2000. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. J. Comput Biol. 7(3-4): 3450–62.

[McCue *et al.* 2001] McCue, L., Thompson, W., Ryan, M., Liu, J., Derbyshire, V. and Lawrence, C. 2001. Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes. Nucl. Acids Res. 29: 774–782.

[McGuire *et al.* 2000] McGuire, A., Hughes, J. and Church, G. 2000. Conservation of DNA regulatory motifs and discovery of new motifs in microbial genomes. Genome Res. 10: 744–757.

[Mukherjee *et al.* 2004] Mukherjee, S., Berger, M.F., Jona, G., Wang, X.S., Muzzey, D., Snyder, M., Young, R.A., Bulyk, M.L. 2004. Rapid analysis of the DNA-binding specificities of transcription factors with DNA microarrays. Nature Genetics 36(12): 1331–1339.

[Neuwald *et al.* 1995] Neuwald, A., Liu, J., Lawrence C. 1995. Gibbs motif sampling: detection of bacterial outer membrane protein repeats. Protein Sci. 4(8): 1618–32.

[Osada *et al.* 2004] Osada, R., Zaslavsky, E., and Singh, M. 2004. Comparative analysis of methods for representing and searching for transcription factor binding sites. Bioinformatics 20(18): 3516–3525.

[Pavesi *et al.* 2004] Pavesi, G., Mereghetti, P., Mauri, G. and Pesole. G. 2004. Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. Nucleic Acids Res. 32: W199–W203.

[Pevzner and Sze 2000] Pevzner, P. and Sze, S. 2000. Combinatorial approaches to finding subtle signals in DNA sequences. In Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology, pages 269–278. AAAI Press.

[Prakash and Tompa 2005] Prakash, A., and Tompa, M. 2005. Statistics of local multiple alignments. Bioinformatics 21 (Suppl 1): i344–i350.

[Price *et al.* 2003] Price, A., Ramabhadran, S. and Pevzner, P. 2003. Finding subtle motifs by branching from sample strings. Bioinformatics 19: 149–155.

[Reinert *et al.* 1997] Reinert, K., Lenhof, H.P., Mutzel, P., Mehlhorn, K., and Kececioglu, J. A branch-and-cut algorithm for multiple sequence alignment. In Proceedings of the First Annual International Conference on Computational Molecular Biology, pages 241–249. ACM Press.

[Rigoutsos and Floratos 1998] Rigoutsos, I., and Floratos, A. 1998. Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. Bioinformatics 14(1): 55–67.

[Robison *et al.* 1998] Robison, K. and McGuire, A. M. and Church, G. M. 1998. A comprehensive library of DNA-binding site matrices for 55 proteins applied to the complete *Escherichia coli* K-12 Genome. J. Mol. Biol. 284: 241–254.

[Salgado *et al.* 2004] Salgado, H., Gama-Castro, S., Martínez-Antonio, A., Díaz-Peredo, E., Sánchez-Solano, F., Peralta-Gil, M., Garcia-Alonso, D., Jiménez-Jacinto, V., Santos-Zavaleta, A., Bonavides-Martínez, C. and Collado-Vides, J. 2004. RegulonDB (version 4.0): Transcriptional Regulation, Operon Organization and Growth Conditions in Escherichia Coli K-12. Nucleic Acids Res. 32: 303–306.

[Schneider *et al.* 1986] Schneider, T. D., Stormo, G. D., Gold, L. and Ehrenfeucht, A. 1986. Information content of binding sites on nucleotide sequences. J. Mol. Biol. 188: 415–431.

[Schneider and Stephens 1990] Schneider, T. D. and Stephens, R. M. 1990. Sequence logos: a new way to display consensus sequences. Nucleic Acids Res. 18:6097–6100.

[Schuler *et al.* 1991] Schuler, G., Altschul, S., and Lipman, D. 1991. A workbench for multiple alignment construction and analysis. Proteins 9(3): 180–190.

[Shannon 1948] Shannon, C. E. (1948) Bell System Tech. J., 27, 379-423, 623-656

[Sinha and Tompa 2000] Sinha, S. and Tompa, M. A statistical method for finding transcription factor binding sites. In: Eighth International Conference on Intelligent Systems for Molecular Biology, San Diego, CA, August 2000, 344–355.

[Sinha and Tompa 2003] Sinha, S. and Tompa, M. 2003. YMF: A program for discovery of novel transcription factor binding sites by statistical overrepresentation. Nucleic Acids Res. 31(13): 3586-3588.

[Spellman *et al.* 1998] Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., Futcher, B. 1998. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Molecular Biology of the Cell 9(12): 3273–3297.

[Staden 1984] Staden, R. 1984. Computer methods to locate signals in nucleic acid sequences. Nucleic Acids Res. 12: 505–519.

[Stormo 2000] Stormo, G. D. 2000. DNA binding sites: representation and discovery. Bioinformatics 16: 16–23.

[Sze *et al.* 2004] Sze, S.-H., Lu, S. and Chen, J. 2004. Integrating Sample-driven and Pattern-driven Approaches in Motif Finding. Proceedings of the Fourth Workshop on Algorithms in Bioinformatics (WABI), pages 438–449.

[Tan *et al.* 2001] Tan, K., Moreno-Hagelsieb, G., Collado-Vides, J. and Stormo, G. D. 2001. A comparative genomics approach to prediction of new members of regulons. Genome Res. 11: 566–584.

[Tatusov *et al.* 1994] Tatusov, R.L., Altschul, S.F., and Koonin, E.V. 1994. Detection of Conserved Segments in Proteins: Iterative Scanning of Sequence Databases With Alignment Blocks. PNAS 91: 12091-12095.

[Tavazoie *et al.* 1999] Tavazoie, S., Hughes, J. D, Campbell, M. J., Cho, R. J., Church, G.M. 1999. Systematic determination of genetic network architecture. Nature Genetics 22(3): 281–285.

[Thieffry *et al.* 1998] Thieffry, D., Salgado, H. Huerta, A. M. and Collado-Vides, J. 1998. Prediction of transcriptional regulatory sites in the complete genome sequence of Escherichia coli K-12. Bioinformatics 14: 391–400.

[Thompson *et al.* 2003] Thompson, W., Rouchka, E. C. and Lawrence, C. E. 2003. Gibbs Recursive Sampler: finding transcription factor binding sites, Nucleic Acids Research, 31(13): 3580-3585.

[Tompa 1999] Tompa, M. An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. In Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology, pages 262–71. AAAI Press, 1999.

[Tompa *et al.* 2005] Tompa, M., Li, N., Bailey, T. L., Church, G.M., De Moor, B., Eskin, E., Favorov, A. V., Frith, M.C., Fu, Y., Kent, W. J., Makeev, V. J., Mironov, A. A., Noble, W. S., Pavesi, G., Pesole, G., Regnier, M., Simonis, N., Sinha, S., Thijs, G., van Helden, J., Vandenbogaert, M., Weng, Z., Workman, C., Ye, C., and Zhu, Z. 2005. Assessing computational tools for the discovery of transcription factor binding sites. Nature Biotechnology 23(1): 137–44.

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

[Vingron and Pevzner 1995] Vingron, M., and Pevzner, P. 1995. Multiple sequence comparison and consistency on multipartite graphs. Advances in Applied Mathematics 16: 1–22.

[Yamauchi 1991] Yamauchi, K. 1991. The sequence flanking translation initiation site in protozoa. Nucleic Acids Res. 19: 2715–2720.

[Workman and Stormo 2000] Workman, C.T. and Stormo, G.D. ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. In Proceedings of the Fifth Pacific Symposium on Biocomputing, pages 467–478, 2000.

[Wang and Jiang 1994] Wang, L. and Jiang, T. 1994. On the complexity of multiple sequence alignment. Journal of Computational Biology, 1: 337–348.

[Wingender *et al.* 1996] Wingender, E., Dietze, P., Karas, H., and Knüppel, R. 1996. TRANSFAC: A database on transcription factors and their DNA binding sites. Nucleic Acids Res. 24: 238241.

[Witten and Frank 2000] Witten, I. H. and Frank, E. 2000. Data mining: Practical machine learning tools and techniques with Java implementations. 119–150. Morgan Kaufmann Publishers, San Francisco, CA.

[Zaslavsky and Singh 2005] Zaslavsky, E. and Singh, M. Combinatorial Optimization Approaches to Motif Finding. Manuscript, submitted for publication, 2005.

[Zhang and Marr 1993] Zhang, M. Q. and Marr, T. G. 1993. A weight array method for splicing signal analysis. CABIOS 9: 499–509.

[Alberts *et al.* 2002] Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. 2002. Molecular Biology of the Cell. Garland Science, New York, NY.