CONSTRAINT BASED METHOD FOR FINDING MOTIFS IN DNA SEQUENCES

DONG XIAOAN

(Bachelor of Management, Wuhan University, China)

A THESIS SUBMITTED FOR

THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgements

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. My primary thanks go to my supervisor, Prof. Sung Sam Yuan, for his invaluable guidance and advice throughout my research. His priceless support has helped me all the time in the research.

I deeply appreciate Dr. Sung Wing Kin for his constructive guidance in my research. He shared with me his knowledge and tips in writing research paper, and provided me friendly encouragement all the way.

I sincerely appreciate my good friends Fa Yuan, Tang Jiajun, Yang Xia, Chen Yabing, Zhou Yongluan, Li Jianer, Zhang Xi. They have helped me in one way or other and made my study and research experience unforgettable.

Last but not least, I am grateful to my parents for their patience and love. Without them this work would never have come into existence.

Contents

Summary

Pattern discovery in unaligned DNA sequences is a fundamental problem in both computer science and molecular biology. It has important applications in locating regulatory sites and drug target identification. This thesis introduces two novel motif discovery algorithms based on the use of constraint mechanism and constraint rules respectively. The key idea is to convert sets of similar substrings on the DNA sequences into patterns, as early as possible, using constraint mechanism or constraint rules. The advantages are two folds. Firstly, the approach generates limited number of patterns while still guaranteeing that the actual motifs are contained in the pattern set. Secondly, the procedure for deriving patterns is very cost-effective since it can be considered as that we use many "look ahead" to speed up the procedure. Therefore, the algorithms have the advantages of the high sensitivity of pattern-driven algorithms as well as the efficiency of sample-driven algorithms.

Chapter 1

Background

The history behind motif discovery in unaligned DNA sequences dates back to 1970, when Hamilton Smith [18] discovered the Hind II restriction enzyme. It may have been the first DNA pattern. This discovery provided biological scientists with a new technological tool to study DNA sequences in a more efficient manner.

Since the dawn of the 21st century, there has been a dramatic increase in the number of completely sequenced genomes due to the efforts of both public genome agencies and the pharmaceutical industries. Large-scale genomics have become a fundamental tool for understanding an organism's biology. Access to multiple complete genomic sequences helps biologists to formulate and test hypotheses about how genomes are organized and evolved, as well as how a genome encodes the observed properties of a living organism. Key questions being pursued include: what parts of our genome encode the mechanisms for major cellular functions like metabolism, differentiation, proliferation, and programmed death? How do multiple genes act together to perform specialized functions? How is our non-protein-coding DNA organized, and which parts of it are func-

tionally important? How do selective pressures act on the random processes of gene duplication and mutation to give rise to complex constructs like eyes, wings, and brains? Why do humans appear so different from worms and flies, despite sharing so many of the same genes?

Until the 1990's, molecular biologists could pursue questions about the content and function of genomes only indirectly, or else at great cost. Indirect techniques such as Giemsa staining and CoT-based measurement of repetitive content [45] provided limited information about a genome. Full sequence was available for only a few short regions found to be functionally significant, usually after a long and expensive process of localization by (e.g.) linkage mapping, followed by cloning out and finally sequencing a minimal region of interest. The cost and time required to sequence DNA made sequencing a tool to be applied only at particular points, and only once a region was shown to be important by other means.

More recently, high-throughput DNA sequencing has enabled a direct approach to studying genomes. Using this new technology, biologists have obtained progressively larger complete genomic sequences, from viruses [11] to prokaryotes [36] to single-celled [19] and multicellular [1] eukaryotes. Available genomes today include those of several higher metazoans, including the fruit fly Drosophila melanogaster [31], the flowering plant Arabidopsis thaliana [2], and, of course, Homo sapiens [3]. Armed with substantially complete euchromatic sequences from these organisms, we can now directly interrogate global properties like base frequencies and repetitive content, obtain immediately the sequence of any potentially interesting region, and perhaps most exciting compare corresponding long stretches of genomic DNA in two or more organisms. Such analysis encompass massive amounts of sequence, on a scale requiring computa-

tion that defies manual analysis. The need to automate analysis of long or numerous genomic sequences gives rise to the field of computational genomics.

In this work, we address a particular problem of computational genomics: how to discover which parts of a long DNA sequence encode particular biological features, such as genes. Even when the whole sequence is available for inspection, finding these features reliably can be surprisingly difficult. If we know little about the features being sought, or their presence leaves only a weak imprint on the underlying sequence, finding them may be theoretically intractable or practically beyond our limited budget of computing time and space. This work focuses specifically on new techniques to find features that are difficult to find in theory or simply intractable to existing search algorithms.

The algorithms that we introduce in this thesis are founded on two novel techniques, constraint mechanism and constraint rules, which extract patterns from sets of similar strings. We show how to exploit the power of them to find motifs efficiently. As a result, we can more readily identify more interesting features and ultimately provide more knowledge to biologists.

1.1 Road Map to the thesis

We begin by providing the reader with a brief guide to the content of the thesis. Some readers may find the biological terms used in subsequent sections and chapters unfamiliar; hereafter, we will both define such terms at their point of first use and provide a glossary (see Appendix A) of terms to collect the important definitions in one place.

Chapter 1 is devoted to background and significance. We first review the nature of genomic DNA. Then we introduce interesting features which our algorithms focus on. Finally we introduce the basic approach, sequence similarity comparison, to identify sequence features.

Chapter 2 is devoted to review the existing research work on motif finding. We first present the formal definition of planted motif finding problem, then we analyze the critical techniques - motif models - used for pattern extraction. Based on the analysis, we review the existing motif finding algorithms. Finally we revisit the significance of our algorithms.

Chapter 3 introduces two novel algorithms, namely constraint mechanism-based motif finding algorithm (CMMF) and constraint rules-based motif finding algorithm (CRMF). We then show how to implement the algorithms in practice.

Chapter 4 presents the experimental results on both synthetic data and biological data. Based on the results, we compare CMMF with CRMF, and we also compare our algorithms with other leading motif finding algorithms.

Chapter 5 summarizes the merits as well as limitations of our work. We propose the ways to extend the algorithms to achieve better performance and pose the open problems as well.

1.2 Biological Background: DNA and Sequence Features

The first prerequisite to developing algorithms for finding features in genomic sequences is to understand what we are looking for and why. We

therefore begin with a brief review of genomic DNA and its major features. Readers seeking more background on genomic DNA or on molecular biology in general may wish to consult the standard text by Lewin [29] or the gentler introduction by Joao Setubal and Joao Meidanis [40].

1.2.1 DNA and Genomic Sequence

The information encoded in genetic material, DeoxyriboNucleic Acid (DNA), is responsible for establishing and maintaining the cellular and biochemical functions of an organism. In most organisms, the DNA (see Figure 1.1) is an extended double-stranded polymer composed of a sequence of nucleotides, also called bases. Four such bases - A(Adenine), C(Cytosine), G(Guanine), and T(Thymine) - form the alphabet from which all natural DNA is constructed. Abstractly, a DNA sequence is simply a string over the alphabet {A,C,G,T}. We will use the terms "string" and "sequence" interchangeably.

The sequence of bases of one DNA strand is complementary to the bases of the other strand. This complementarity enables new DNA molecules to be synthesized with the same linear array of bases in each strand as an original DNA molecule. The process of DNA synthesis is called replication, which plays a critical role in passing on genetic information from one generation to the next. Complementary bases forms base pairs. The pairing is deterministic: A always pairs with T, while C pairs with G. Thus, the sequence of one strand determines the sequence of its complement, and we can describe a DNA sequence uniquely by only one of its strands. Because of this pairing, bases are sometimes classified as "weak" (A/T, joined by two hydrogen bonds) or "strong" (C/G, joined by three hydrogen bonds). Another common classification of bases, this

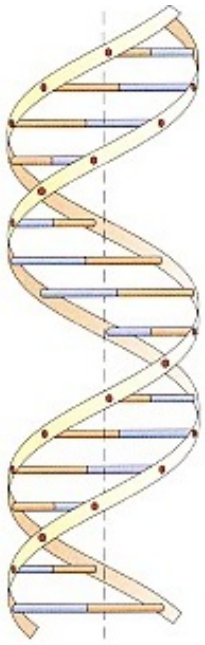

Figure 1.1: Double Stranded DNA Model

time by chemical structure, is as purine (A/G) or pyrimidine (C/T). An unspecified purine or pyrimidine is denoted by the characters R and Y respectively.

DNA either swims within the cytoplasm of prokaryotic cells (e.g. bacteria and E.coli) or locates within the nucleus of eukaryotic cells (e.g. plant and animal). An organism's complete set of DNA sequence is its genome. The differences in genomic sequence from one organism to another within a species are quite small compared to the differences between species, so it makes sense to talk about an entire species' genome. For example, the human genome, which is 3×10^9 base pairs in length, is 99.9% similar between individuals, while the genome of our closest relative, the chimpanzee, is only 98% - 99% similar to ours [8].

An organism's genome is organized into a small number of discrete DNA molecules, called chromosomes. Bacteria typically have a single,

circular chromosome a few million bases in length, while eukaryotic species have anywhere from three to over 100 linear chromosomes of total length ranging from tens of millions up to billions of bases.

An essential feature of DNA is that it is not static over time. Chemicals, radiation, and copying errors can all cause a DNA sequence to mutate. Biologically common types of mutation include substitutions, in which one base is replaced by another, and indels (insertions and deletions), in which bases are added to or removed from a sequence. Different types of mutation happen at different rates; for example, transition substitutions - those that replace A with G or C with T and vice versa - are roughly twice as common [9] as other substitutions, which are called transversion.

1.2.2 Regulatory Sites - a Feature of Genomic Sequence

Most sequence features fall broadly into three categories: genes, which encode the active molecules that carry out the cell's business; regulatory sites, which control the behavior of genes; and repetitive elements. Our algorithms focus on finding regulatory sites, which will be introduced in detail at follows.

Regulatory sites control the behavior of genes. Precisely, regulatory sites control when and where genes are expressed to produce their products. It is necessary to know genes before we illustrate regulatory sites.

Genes are the basic physical and functional units of heredity. A gene is a specific sequence of bases, which encode instructions for building other polymeric molecular species. A gene's basic function is to have its DNA

sequence transcribed into a corresponding (single-stranded) polymer of RNA, or RiboNucleic Acid. The sequence of an RNA molecule is identical to that of its originating gene, except that T bases are mapped not to T but rather to a different base, U (Uracil).

Cells have regulatory mechanisms for controlling when and where genes are expressed to produce their products. Sets of short stretches of base pairs (signal regions) within the DNA are required to ensure that gene expression is initiated at the correct nucleotide and that it terminates at a specific nucleotide. The sequences that control the initiation of gene expression usually precede the coding sequence, and termination signal sequences follow it. Figure 1.2 illustrate how a structural gene in prokaryotes is transcribed into mRNA [16], which then is translated into protein. In prokaryotes, a contiguous DNA segment forms a structural gene. Prokaryotic transcription entails the binding of RNA polymerase to a promoter region, the initiation of transcription at the first nucleotide of the gene, and the cessation of transcription at a termination sequence that lies downstream from the coding region.

In this work, we focus on one particular form of regulation: control of gene transcription by a class of proteins called transcription factors. These proteins adhere to genomic DNA at binding sites, regions up to a few tens of bases in length that contain factor-specific signal sequences. Transcription factors often bind at sites within a few hundred bases at the start of a gene, where they influence how frequently the RNA polymerase complex initiates transcription of that gene. These sites are called enhancer/repressor regions. If a transcription factor causes the gene to be expressed at a higher level, it is said to be an enhancer; if it causes a lower level of expression, it is a repressor. Figure 1.3 illustrates how a repressor

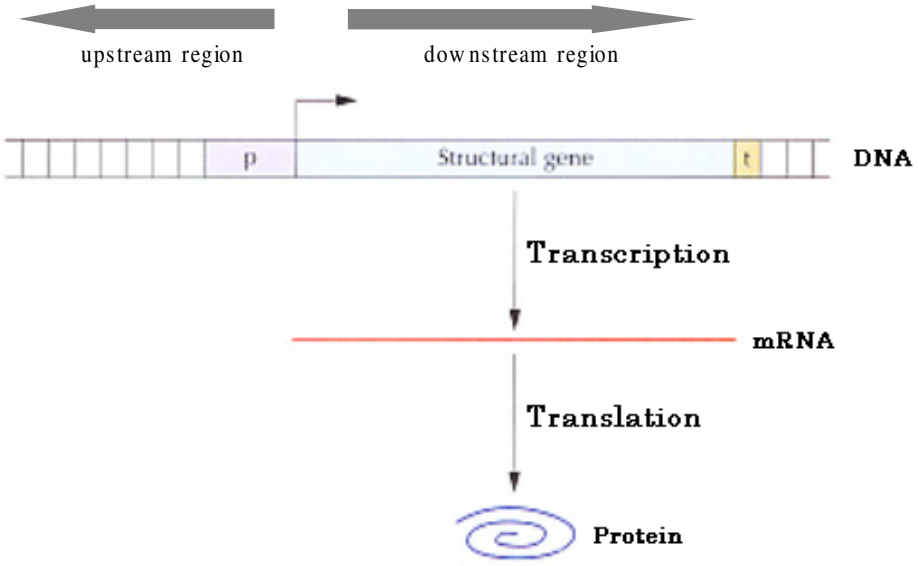

Figure 1.2: Prokaryotic Transcription. *Schematic representation of a prokaryotic structural gene. The promoter region (p), the site of initiation and direction of transcription (the right-angled arrow), and the termination sequence for RNA polymerase (t) are depicted. A prokaryotic structural gene is transcribed into mRNA and then directly into protein.*

protein binds to a regular binding site to block the transcription.

Transcription factors are often activated in response to changes in the cell's environment, especially changes in the amounts of various chemicals (including other gene products). These proteins can therefore orchestrate the cell's transcriptional response to changing external conditions as well as carrying out "programs" such as cell division, differentiation, or death in response to particular chemical signals. The exact mechanism by which transcription factors transduce these changes varies. Many factors form (or block formation of) protein complexes that contact the RNA polymerase directly, increasing or decreasing its affinity for binding to a gene's promoter and initiating transcription [29]. Factors may also alter the conformation of the DNA to which they bind, again changing the binding affinity of the polymerase [38, 39].

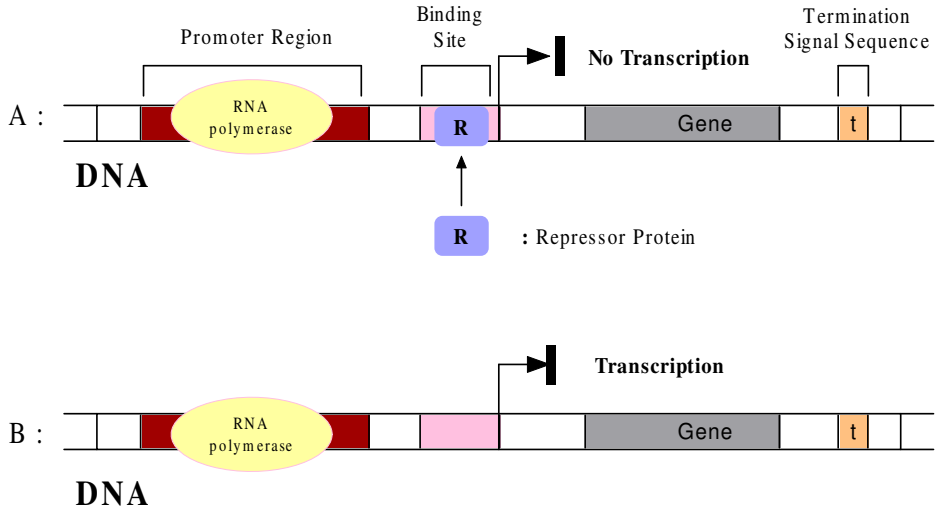

Figure 1.3: Schematic representation of a bacterial transcription unit. *Transcription is catalyzed by RNA polymerase. In Figure A, the repressor protein (R) binds to the regular binding site and blocks transcription. In Figure B, the repressor protein can not bind to the binding site due to some chemical changes, thus RNA polymerase can transcribe the gene.*

Multiple transcription factors can act on a single gene, in which case several different binding sites may cluster near that gene. The factors' actions are not necessarily independent; in general, they may form a complex cis-regulatory logic that permits fine control over when and how strongly a gene is expressed. At this time, few examples of cis-regulatory logic have been worked out in detail; the work of Yuh et al. in sea urchin development [53] illustrates the complexity possible in such logic.

Transcription factor binding sites, while clearly are important sequence features. Unfortunately, they are difficult to identify in raw genomic sequence. We know that sites are likely to occur in clusters in the promoter regions of genes, typically within a few hundred to a few thousand bases of the transcription start site. However, significant sites may be found elsewhere, including the introns of genes [23] and locus control regions that may be ten kilobases (ten thousand bases) or more away

from the genes they regulate [14]. In general, we cannot assume much a priori about what binding sites look like - their sequence patterns are too dependent on the particular factor that they bind. Certain types of transcription factor may require binding sites with known structure, such as a DNA palindrome for some homodimeric factors, but such structures are far from universal.

Finally, we note that even if all the sites for a given transcription factor had identical sequence (which is not the case), the sequence pattern is usually short enough that it may occur purely by chance in the background sequence, at a place where no protein actually binds. Programs to find new transcription factor binding sites in genomic sequences are therefore challenged not only by a lack of identifying characteristics for these sites but also by confusions between true binding sites and chance occurrences of their sequence patterns.

1.3 Finding Sequence Features based on Sequence Similarity

We now come to the vital problem of identifying features in raw DNA sequence. There is well-known conjecture that in the industry of biology that, if two DNA sequence are highly similar, we can infer that they share similar function. Consequently, researchers of bioinformatics can find interesting sequence features through comparing the similarity between two or more biological sequences.

The similarity between the occurrences of a feature is due to its conservation, or lack of change, over evolutionary time. Although all DNA

sequences are subject to mutation, natural selection ensures that we observe today only those individuals whose ancestors' reproductive fitness was not limited by strongly deleterious mutations. Many mutations to genes or regulatory elements can render them dysfunctional, causing the organism carrying these mutations to die or to have fewer viable offspring. In contrast, mutations in nonfunctional sequence can accumulate freely with no effect on reproductive fitness. We therefore expect that the organisms we see today exhibit fewer mutations, or equivalently more conservation, in their functional sequences than in their background sequence.

Sequence alignment is a quantitative measure of similarity. Suppose that some ancestral DNA sequence s_0 evolves by mutation along two separate lineages, creating present-day sequences s_1 and s_2. If we knew the entire mutation history of s_1 and s_2, we could match up those bases in each sequence that derive from the same ancestral base of s_0. Figure 1.4 shows such a matching, or alignment, of two sequences, written as a series of columns in which bases deriving from the same ancestor appear in the same column. If, as in this example, the sequences are subject to indels, the alignment contains gaps, represented in the figure by columns containing dashes "−", where bases in one sequence do not correspond to any part of the other sequence.

The goodness of alignment is defined by $\sum_i \delta(s_1[i], s_2[i])$, where $\delta(x, y)$ is a similarity function between x and y, each is a single base or a single space. e.g., $\delta(x, y) = 2, -1, -1, -1$ for match, dismatch, delete and insert respectively. In the example illustrated in Figure 1.4, We can check that the optimal alignment has the maximal score. An optimal alignment between two sequences can be computed using global alignment, in

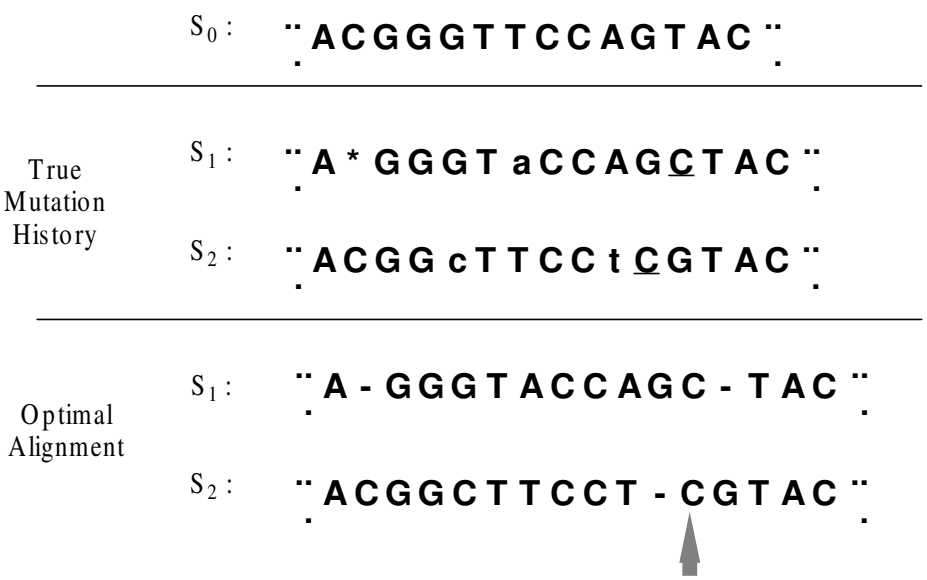

Figure 1.4: Example of a optimal alignment between two DNA sequences s_1 and s_2 with a common ancestor s_0. *In the true mutation history, lowercase letters indicate substitutions, while underlined bases and "$*$" indicate insertions and deletions. In the optimal alignment, some spaces, indicated by "$-$", are introduced to match as much as letters in the two sequences. Note that the best alignment of the sequences is historically incorrect. The two bases, indicated by arrow, do not derive from the same ancestral base.*

particular the Needleman- Wunsch dynamic programming algorithm [33].

Features are always embedded in long genomic sequences. Compared with features, background sequences are either wholly unrelated or so ill-conserved as to be unalignable. To find short and well-conserved features in long background sequences, we can use local alignment, in particular the Smith-Waterman dynamic programming algorithm [47], which ignore the background sequence and measuring only the similarity between features.

As shown in the Figure 1.4, even the optimal alignment may not reflect the true history of two sequences. The fact is that, the history of modern genomic sequence is unknown, and what we can do is to plausibly guess at the true matching of bases by finding an optimal alignment.

Sequence similarity forms the basis to find interesting features in long genomic sequences. Similar substrings between sequences are considered as possible occurrences of a feature. Based on such substrings, we derive the possible feature and verify it globally against all background sequences.

Chapter 2

A Survey of Motif Finding Algorithms

In this chapter, we first formalize the motif finding problem. Then we analyze the critical techniques - motif models - used for pattern extraction, and discuss their strengths and limitations respectively. Based on the analysis, we review the existing motif finding algorithms. Finally we revisit the significance of our algorithms. Note that we focus specifically on the widely studied problem of finding regulatory motifs in genomic sequence by ungapped multiple local alignment.

2.1 Problem Definition

A motif is a conserved DNA sequence pattern recognized by a transcription factor or by other cellular machinery. The conservation of a regulatory motif across organisms or across genes allows us to identify it through similarity search. However, since regulatory motifs are so short and are imperfectly conserved, limited occurrences of a motif by themselves may

not provide significant evidence of conservation. For example, consider the problem of finding two occurrences of a conserved 20-mer motif that differ by only five substitutions, in a pair of 1-kb background sequences that are randomly generated with equal base frequencies. The expected number of 20-mer matches with at most five substitutions appearing by chance in the background is about 3.67, so two occurrences of the motif would be indistinguishable from the background. Unless we can localize the motif to a very much smaller region, the only way to demonstrate its significance is to find additional occurrences in other sequences.

Following Buhler & Tompa [7], the formal definition of the motif discovery problem can be as follows.

Planted (l, d) - **Motif Problem**: Consider a set E of t nucleotide sequences each of length n. Suppose there is a fixed but unknown nucleotide sequence M (motif) of length l which is implanted in every sequence of E. The motif discovery problem is to determine M given E. More precisely, the problem is to compute M such that every sequence in E contains a length-l substring which has at most d mismatches when compared with M.

Note that there are two widely used consensus based motif models, where the motif consists of instances which are mutated occurrences of the motif skeleton. One is FM model [35] where each of the t sequences contains one instance of an (l,d)-motif. The other one is VM model [35] where again each sequence contains exactly one instance, only now each position of the instance is mutated, independently of all other positions, with probability ρ. Due to our work concentrate on the first model, it is used in the above formulation of motif problem.

2.2 Motif Models: Strengths and Limitations

It is always difficult to identify all the occurrences of a conserved motif without any information of the motif, especially in the case of substantive background sequences. Most existing algorithms capture the motif skeleton, an estimated motif, through collecting partial occurrences as a start, then we try to find additional occurrences against the whole background to restore the motif. Obviously the procedure of extracting out the motif skeleton from partial occurrences plays an critical role in deciding the accuracy of these algorithms. And this procedure is more often called as pattern extraction. Many different pattern extraction methods exist for multiple sequences [17]. However what we focus on are not these methods themselves, but several underlying motif models commonly used in these methods. They are consensus model, the profile or weight matrix model (WMM) and multiprofile model. We also introduce constraint based model used in our algorithms.

It is assumed that the occurrences of a motif may differ only by substitutions, not by indels (insertions or deletions) in the above four models. This assumption reflects (1) the limitations of many computational technologies for finding motifs and (2) the fact that biologically interesting motifs are frequently ungapped. Some known motifs consist of a small number of ungapped segments with intervening variable-length spacers [26, 41]; such motifs can be modelled as a collection of ungapped consensus whose occurrences always appear near each other with gaps of varying length.

2.2.1 Consensus Model

The consensus model is a simple combinatorial description of a motif. In this model, the motif is considered as a consensus sequence. Each occurrence of the motif is a copy of the consensus sequence, perhaps with a few substitutions. Given multiple occurrences of a motif, the consensus sequence can be formed as follows. The consensus at each position of multiple sequences is defined as the base which occurs most often at the position. In the case that two or more bases have equal highest occurrences at a position, the consensus can be chosen randomly from these bases. And the consensus sequence consists of the consensuses at each position as illustrated in Figure 2.1.

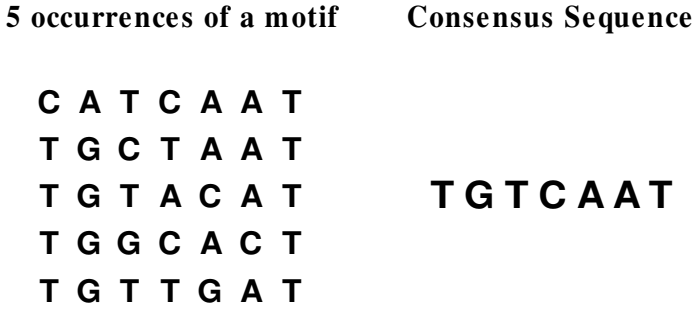

Figure 2.1: A consensus model inferred from five occurrences of a motif. *The most frequent base in each position of the occurrences becomes the base of the consensus at the position. If two or more bases appear equally often in a given position, as with T and C in the fourth position, the choice of the consensus base at that position is arbitrary.*

One could measure the conservation of a motif by the number of substitutions between each occurrence and the consensus sequence.

Strength. Consensus model is the simplest model. Given multiple occurrences, it extract a single pattern - consensus sequence. In most cases, it is effective in the sense that the base that appears most frequently in each

position has the highest likelihood to be the original base of the motif.

Limitation. Consensus model risks missing the actual motif. This happens in the situation that the base at any position of the motif is badly conserved in its occurrences.

2.2.2 Weight Matrix Model

The consensus model is uninformative due to that it does not reveal either how strongly the consensus base in each position is conserved or the distribution of non-consensus bases. However, all these information are described in the weight matrix model (WMM), also called profile model. WMM is a probabilistic model, which models a motif of length l as a $4 \times l$ matrix M, where the entry at position $M[p, q]$ gives the probability that an occurrence of the motif contains a base q (q = A,C,G,T) in its pth position. Each column of the matrix therefore sums to one as illustrated in Figure 2.2. The distribution of bases in different positions are independent of each other. Given a length-l sequence s, let $s[i]$ denotes the base at its ith position. Based on the weight matrix, the probability that M produces a particular length-l motif instance m is : $Pr[\ m \mid W\ ] = \prod_{i=1}^{l} W[\ m[i],\ i\ ]$. Given a set of motif occurrences M, the weight matrix $W[\ M\ ]$ can be easily computed by calculating the frequency of each base in each position. The weight matrix of the five motif occurrences in Figure 2.1 is shown in Figure 2.2.

The matrix $W[\ M\ ]$ is the best description of M in the sense of maximum likelihood. It is the WMM W that maximizes the likelihood $L[\ W[M] \mid M\ ] = \prod_{m \in M} Pr[\ m \mid W\ ]$. And the likelihood $L[\ W[M] \mid M\ ]$ is also a useful score by which to measure the extent of conservation of the

5 occurrences of a motif

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

Weight Matrix

	1	2	3	4	5	6	7
A	0	0.2	0	0.2	**0.6**	**0.8**	0
C	0.2	0	0.2	**0.4**	0.2	0.2	0
G	0	**0.8**	0.2	0	0.2	0	0
T	**0.8**	0	**0.6**	0.4	0	0	**1**

Figure 2.2: A weight matrix model (WMM). It is inferred from the five motif occurrences in Figure 2.1. *Entries corresponding to the consensus base at each position are identified in bold face. Unlike the consensus model, the WMM captures the frequencies of both consensus base and non-consensus bases, and it remains well-defined even when the consensus base is ambiguous, as in the fourth position.*

motif.

If the motif occurs in random background sequences with a base distribution P, a better scoring function for the set M of motif occurrences is the likelihood ratio $LR(M)$, defined as

$$LR(M) = \frac{L[\, W(M) \mid M \,]}{L[\, P \mid M \,]}$$

where

$$L[\, P \mid M \,] = \prod_{m \in M} Pr[\, m \mid P \,]$$

The likelihood ratio, while is not strictly a measure of conservation, is a principled way to account for the background base distribution when scoring a motif. The ratio adjusts for the background distribution by recognizing that, if base i appears frequently in the background, then a collection of strings with a high frequency of i's is more likely to occur purely by chance, and is therefore less significant as a putative motif, than one with few i's.

Strength. WMM is a probabilistic model, which captures the frequencies

of each base in each position. It is the best description of M in the sense of maximum likelihood. In addition, the impact of the background distribution can be taken into account for measuring the conservation.

Limitation. Instead of extracting a specific motif, WMM provides the information to infer the likelihood that any length-l string is the actual motif. However, it is possible that the model is biased on wrong bases in some positions in the situation that mutations occur preferentially on a small subset of positions of its occurrences. To get the model best reflect the actual motif, the initial model need be refined using the technique - expectation maximization (EM, [4, 28]). Unfortunately, the refinement procedure involves huge computational cost.

2.2.3 Multi-positional Profile Model

Multi-positional profile model utilize a "corrective" system to modify a motif occurrence to the actual motif. This model is introduced by Keich and Pevzner [24], and it is successfully deployed in the algorithm MULTIPROFILER to find motif effciently. Multi-positional profile model is different from consensus model as well as weight matrix model in the sense that it is applied to a set of strings which include both motif occurrences and background strings, instead of a pure set of motif occurrences. Given a motif occurrence m, a set S of strings which have hamming distance no greater than $2d$ are identified from the background sequences to aid in modifying m. Usually the random substrings, also called noises, of background sequences dominate the set S. However the use of multi-positional profiles is able to make the noises widely distributed while the motif occurrences stay centralized. Thinking chemically, this measure is

like that we make the purities obvious through diluting the impurities. An example are shown at Figure 2.3.

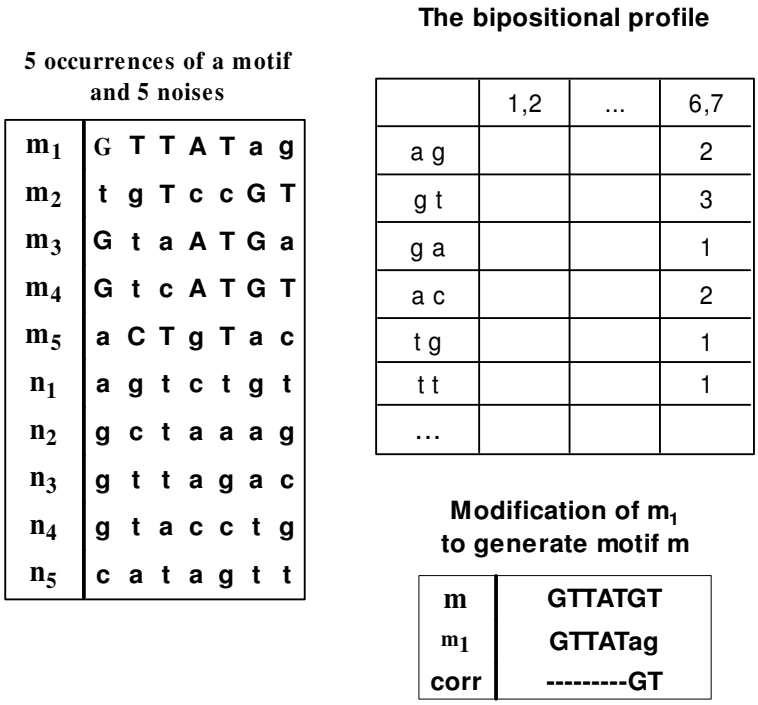

Figure 2.3: Multi-positional profile model inferred from sample sequences which consist of the five motif occurrences in the Figure 2.1 and additional five random sequences. *The motif skeleton is* $m = GTTATGT$. *There are five motif occurrences in the sample sequences:* $m_1,\cdots,m_5$. *And five random sequences are also included:* $n_1,\cdots,n_5$. *Note that pairs consisting of the two rightmost bases of random sequences are widely distributed in 5 areas:ac,ag,gt,tg,tt. While those pairs consisting of the two rightmost bases of motif occurrences are concentrated on only three areas:ag,ga,gt, and mainly on the last area.*

The application of multi-positional profile model is based on the assumption that a motif occurrence(reference sequence) has been found. The choice of sample sequences to which multi-positional profile model is applied is based on a simple principle, that is any one in the sample sequences should agree with the motif occurrence except on at most α positions. In the case of an (l,d)-motif, $\alpha = 2d$ is the choice to strike the balance between allowing as many as motif occurrences into sample

sequences and decreasing the noise.

The subsequence of a string, which is typically nonconsecutive, is denoted as *stringlet*. A k-stringlet is defined in terms of its k positions in a string and their content. For example, the string $ATGTAT$ contains the 3-stringlet $-T--AT$. The stringlet which is used to correct a motif occurrence m_i should be disjoint from m_i in the sense of that it differs from m_i in all its positions. For details, you may wish to consult the paper [24].

Strength. Multi-positional profile model allows one to detect subtle consensus sequences that escape detection by the standard profiles.

Limitation. Since its application involves all the occurrences of the motif and a large amount of noises, the computational cost of deriving a motif is huge. In addition, the hope for deriving the actual motif rely on the distribution of noises. More specifically, its success rely on the uniform distribution of stringlets of sample sequences which correspond to the mutated positions of the reference sequence. In the situation that those stringlets are centralized, the model will fail.

2.2.4 Constraint based Model

Constraint based model generate possible motif skeletons which satisfy the pre-defined constraints. As its name show, the key element of this model is constraints. And constraints are formed based on the features of various motif finding problems. In the case of (l,d)-motif problem, the constraint which qualify a motif is that its every occurrence has at most d substitutions relative to the motif. Precisely, the hamming distance between a motif m and every motif occurrence m_i should be no greater

than d: $dist(m, m_i) \leq d$. Figure 2.4 show the example to apply this model to the 5 motif occurrences in Figure 2.1. Compared with consensus model, constraint based model try to find all possible motif skeletons instead of the most likely one indicated by a set of motif occurrences.

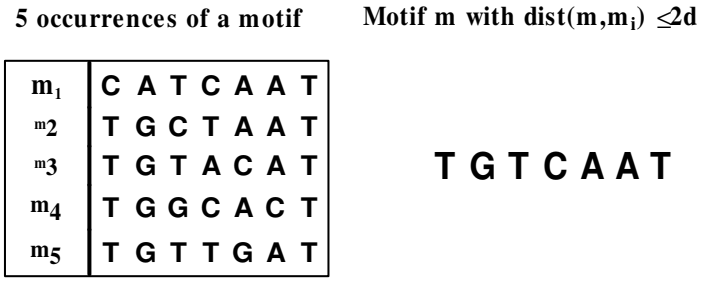

Figure 2.4: Constraint based model inferred from the five motif occurrences in Figure 2.1. *There exists only one motif in this case. Unlike the consensus model where the base in the fourth position is arbitrarily selected due to C and T has the same frequency, the only choice for the position is C so that the constraint* $dist(c, m_i), 1 \leq i \leq 5$ *is met.*

With the help of constraint mechanism and constraint rules, which is to be introduced in Chapter 3, this model can be both reliable and economic in the sense that actual motifs are never missed and all patterns are generated in a cost-effective way.

Strength. Given a set of motif occurrences, constraint based model never fails to include the actual motif in its derived pattern set. The complexity is low even we apply an exhaustive search to find all the centers of limited number of motif occurrences. Furthermore, with the help of constraint rules, the complexity can be reduced to the theoretical limit in that motifs are enumerated straightforwardly.

Limitation. This model may generate too many patterns. It involves much computational cost to filter out "noises" among the huge pattern

set. However, there exist efficient filtering technique to overcome this flaw, which is addressed in Chapter 3.

2.3 Motif Finding Algorithms

A number of algorithms have been proposed to find motifs in DNA sequences. These algorithms can be classified into two categories: enumeration and local search.

Enumerative algorithms, also called pattern driven algorithms, usually test all 4^l length-l patterns to find the high-scoring patterns according to some metrics. Enumerative algorithms include methods by Brzama et al. [6], Staden [42], Pesole et al. [34], Wolfertstetter et al. [52], van Helden et al. [49] and Tompa [48].

While enumerative algorithms are guaranteed to find the highest-scoring motif in the input, searching through all 4^l length-l patterns exhaustively becomes impractical for large l. One way to lower these methods' high cost is to enumerate partial motifs much smaller than the desired length, then try to assemble them into full-length motifs. This strategy is implemented by the TEIRESIAS algorithm by Rigoutsos and Floratos [37]. However the drawback is that the running time is exponential in the motif length l. Thus its implementation is almost impractical, especially for the currently fast-growing DNA database.

In order to come up with some practical solution to the motif finding problem, motif finders resort to the heuristic approach of local search. Local search methods guess an initial model of the motif, then iteratively make small changes to the model that improve its score with respect to the input sequences. The model eventually converges to a local maximum

whose score cannot be improved by further iteration but which is not guaranteed to be the globally highest-scoring motif. Local search methods increase their chances of finding the globally best motif by guessing many different initial models, iteratively improving each one, and finally reporting the highest-scoring motif resulting from any guess. Iterative improvement of the likelihood score can be performed numerically by expectation maximization (EM) or seminumerically by greedy search over models or by Gibbs sampling.

Local search is the technique of choice for sample-driven algorithms. Local search is used to limit the search based on the patterns appearing in the sequences from the sample. Sampe-driven algorithms include methods by Bailey and Elkan [5], Fraenkel et al. [13], Li et al. [30], Gelfand et al. [15], Buhler and Tompa [7], Hertz and Stormo [21], Lawrence et al. [27], Lawrence & Reilly [28] and Pevzner & Sze [35].

Although sample-driven algorithms has relatively low computational cost, local search needs to be taken with caution in the case of subtle signal. The problem is that the approach may eventually find a local optimal motif rather than the best motif in the situation that it is difficult to distinguish the motif instances from noises that are similar to the motif just by chance.

PROJECTION (Buhler and Tompa, [7]) and MULTIPROFILER (Keich and Pevzner, [24]) may be the best currently available algorithms on motif finding. The former one first uses the weight matrix model introduced in Section 2 to derive initial motif model from sets of substrings, then it use expectation maximization [4] to change the initial model to the one that has a locally maximum-likelihood. The later one uses the multi-positional profile model introduced in Section 2. Both algorithms

are able to find subtle motifs more reliably than previous algorithms. The following is a brief introduction of their performance. Details are given in Chapter 4.

PROJECTION succeeds in 16 out of 20 times in finding the same (15,4)-motif implanted in twenty 2000 bp sequences while all previous algorithms failed to find. However, MULTIPROFILER not only successfully finds the same motifs in more than 99% of the time, but also finds motifs implanted in twenty 3000 bp sequences in more than 98% of the time. The performance level has been pushed forward greatly by these two algorithms.

2.4 Significance of the Thesis Revisited

Armed with some knowledge on existing motif finding algorithms , we revisit the significance of this work. Most motif finding algorithms either pursue high sensitivity at the price of high computational cost (pattern-driven algorithms), or reduce search cost at the price of limiting the search's sensitivity (sample-driven algorithms). In this work, we develop two constraint based algorithms which have the best of both worlds. Precisely, the algorithms have the advantages of high sensitivity of pattern-driven algorithms as well as the efficiency of sample-driven algorithms.

The high sensitivity of the algorithms is realized through the use of constraint based model introduced in Section 2. Given a set of motif occurrences, the model guarantees the actual motif is included in its derived patterns. The efficiency of the algorithms is realized through the cost-effective pattern extraction methods and the advanced pattern filtering techniques.

Experimental results on synthetic data have shown that our algorithms outperform those leading motif finding algorithms.

Chapter 3

Finding Motif using Constrain Based Method

In this chapter, we present two novel algorithms for the planted (l,d)-motif problem, namely CMMF (constraint mechanism-based motif finding algorithm) and CRMF (constraint rules-based motif finding algorithm). Both algorithms are based on the use of constraint based motif model introduced in Chapter 2.2. What distinguish CMMF and CRMF is that they implement the constraint based motif model using two different techniques, namely constraint mechanism and constraint rules. Intuitively, constraint mechanism is a general mechanism that is able to convert any set of strings into corresponding patterns. In contrast, each constraint rule is a refined constraint mechanism, whose capability is limited to convert some specific sets of strings, however with enhanced efficiency.

This chapter is organized as follows. Section 1 gives some preliminary definitions to be used throughout this chapter. Then, Section 2 introduces the constraint mechanism, including both the naive version and the improved one with heuristics. Section 3 introduces the algorithm CMMF

that exploits the constraint mechanism to discover motifs. Section 4 and 5 are devoted to introduce constraint rules and constraint rules-based algorithm CRMF.

3.1 Preliminaries

This section gives definitions and some simple results that will be useful later.

Both constraint mechanism and constraint rules we will develop later take three length-l strings as input, and the output is a set of strings which have hamming distance at most d to every input string. Based on this principle, we have the following definitions.

Let $S = \{s_1, s_2, s_3\}$ be a set of three length-l strings s_1, s_2, s_3. For any two sequences s_i and s_j of length l, $dist(s_i, s_j)$ is defined to be the hamming distance between s_i and s_j, that is, the number of mismatches between s_i and s_j. Let $dist(s, S) = \sum_{i=1,2,3} dist(s, s_i)$ be the distance from a length-l string s to a set of strings S. Given a set S, then a string s is a *center string* (also called a *center* for simplicity) of S iff $dist(s, s_i) \leq d$ for $i = 1, 2, 3$. By way of contrast, s_m is a *median string* of S iff there is no string $s^{'}$ with $dist(s^{'}, S) < dist(s_m, S)$.

With the above definitions, we can clarify the purpose of constraint mechanism more clearly. Given any set S, the constraint mechanism derives all possible centers of S. Let $C(S)$ be the complete set of centers of S, that is, $C(S) = \{c \mid dist(c, s_i) \leq d,\ 1 \leq i \leq 3\}$. i.e., consider a set $S = \{s_1 = ccccaaaaaaaaaaa, s_2 = aaaaggggaaaaaaa, s_3 = aaaaaaaaattttaaa\}$. When $d = 4$, the only center of S is $aaaaaaaaaaaaaaa$. Therefore $C(S) = \{aaaaaaaaaaaaaaa\}$.

We can also think of a set S as a $3 \times L$ base matrix. Then we refer to the columns of this matrix as columns of the set of strings. For any string s of length l, we use $s[p]$, $1 \leq p \leq l$, to denote the base at position p in s. Note that, given a set S, a median string can be easily computed by choosing, in every column, a base occurring most often. If a base is chosen in this way, we call it the majority vote; it is, however, not necessarily unique.

Any column o can be put in one of the following 3 types: (A) three bases differ from each other, e.g., $s_1[o] \neq s_2[o]$, $s_1[o] \neq s_3[o]$ and $s_2[o] \neq s_3[o]$; (B) two of them are the same while the other is different, e.g. $s_1[o] = s_2[o]$,$s_1[o] \neq s_3[o]$; (C) three bases are the same, e.g. $s_1[o] = s_2[o] = s_3[o]$. For type B, we further define B_i for $i = 1, 2, 3$ as the column type where $s_i[o]$ differs from the other two bases in column o.

3.2 Constraint Mechanism

This section introduces the basic algorithm to implement constraint based model (introduced in Chapter 2.2), constraint mechanism. We also present heuristic improvement for the basic algorithm.

Constraint mechanism is the engine to extract patterns. Given any set of three strings, it is able to derive all possible local patterns (centers). It has two features: 1. It is efficient in the sense that most strings it generates in the course of pattern extraction are centers. 2. It is accurate in the sense that, it guarantees that the actual motif is included in the derived patterns.

3.2.1 The Basic Algorithm

The idea of our strategy for deriving centers of any given set S is to start with its median string, which has the minimum distance with S. Then we recursively try all the ways to mutate the median string to develop all possible centers. Constraints serve to restrict the way that the candidate center is mutated.

In the mutating procedure, a mutation is defined as that the current base at a particular position is replaced by the other base. Thus the mutating procedure can be considered as a combination of mutations without any two mutations happening at the same position. In what follows, it is implicit that mutations never happen at ever-mutated positions.

Algorithm 1 outlines a recursive procedure for deriving centers of any given set S. It is based on the bounded search tree paradigm that is frequently successfully applied in the development of fixed-parameter algorithms [22, 10, 12]. A parameter s is initialized to a median string s_m and a parameter p is initialized to 0. In each recursive call, we mutate the string s using different ways and in each way at most one mutation is permitted which happen only at the pth position or a latter position. In this way, we can avoid either running into the situation that two or more mutations happening at the same position or finding the same center multiple times. The mutating procedure is realized through the recursive call of the algorithm. For the correctness of the algorithm we need the following simple constraint.

CONSTRAINT 1. *Given a set of strings S, assuming no or a few mutations have happened on its median string s_m. If the resulting string $s_m^{'}$ has distance greater than $3d$ to the set S, then it is impossible to*

Algorithm 1 Algorithm D, recursive procedure CM (s, p)

Global variables: a set of 3 strings $S = \{ s_1, s_2, s_3 \}$ and a set of centers C.
Input: center seed s and position p.
(D0) If $dist(s, S) > 3d$, then stop ;
(D1) If $dist(s, s_i) \leq d$, $\forall i = 1, 2, 3$, then insert s into C ;
(D2) For every position $i \in \{p, \ldots, l\}$ do
    $B := \{b \mid b \neq s[i]\}$;
    For every base $b \in B$ do
        $s^{'} := s$;
        $s^{'}[i] := b$;
        CM($s^{'}$, $i + 1$)

generate centers by further mutating $s^{'}_m$.

PROOF. We find that it is sufficient to concentrate on unchanged positions of the string $s^{'}_m$ in that mutations never happen at ever-mutated positions. Any mutation will either maintain or increase the distance between $s^{'}_m$ and S. The reason is as follows. The distance $dist(s_m, S)$ can be measured columnwisely. In each unchanged position, $s^{'}_m$ inherits from s_m the base that causes the minimum number of mismatches with S. Therefore, if $dist(s^{'}_m, S) > 3d$, the further mutated string $s^{''}_m$ will also have distance greater than $3d$. It follows that $s^{''}_m$ cannot be a center of S. $\triangle$

Correctness. We have to show that Algorithm D can find all possible centers of any given set of strings S.

Starting from a median string s_m, which has the minimum distance dis to the set S, Algorithm 1 recursively tests if the string is a center or not, then it tries all the ways to move around to strings which have distance dis or $dis+1$ to the set S. It stops until it moves "too far away" from the set S. In this way, all the strings that have distance no greater than $3d$ are scanned and tested. Therefore the resulting center set should

Algorithm 2 The refined instruction of D2 in Algorithm 1.

For every position $i \in \{p, \ldots, l\}$ do
  If ith column of the set S is of type A do
    If $dist(s, S) \leq 3d$ do
      $B := \{b \mid b \neq s[i]\}$;
      For every base $b \in B$ do
        $s^{'} := s$;
        $s^{'}[i] := b$;
        CM($s^{'}$, M)
  If ith column of the set S is of type B do
    If $dist(s, S) < 3d$ do
      $B := \{b \mid b \neq s[i]\}$;
      For every base $b \in B$ do
        $s^{'} := s$;
        $s^{'}[i] := b$;
        CM($s^{'}$, M)
  If ith column of the set S is of type C do
    If $dist(s, S) < 3d - 2$ do
      $B := \{b \mid b \neq s[i]\}$;
      For every base $b \in B$ do
        $s^{'} := s$;
        $s^{'}[i] := b$;
        CM($s^{'}$, M)

consist of all possible center strings.

3.2.2 Heuristic Improvement

The algorithm's performance depends on the efficiency of instruction D2. The goal is to refine this instruction to achieve better performance. The refined instruction D2 is shown in Algorithm 2.

In each recursive call, instruction D2 will be performed upon the common condition that the string s has distance at most $3d$ to the set S. In instruction D2, the string s is mutated in $(l - p + 1)$ ways, and each of them is in the form that a mutation happen at or between positions from p to l. However, some of these ways can be avoided through the use of the following three constraints. In addition, with the new instruction D2,

we can avoid the execution of D0. These constraints are developed based on the observation of the features of different column type introduced in Section 1. We refer to the type of a position of a string s as the type of the corresponding column of the set S for the convenience of explanation.

CONSTRAINT 2. *Given a set of strings S, assuming no or a few mutations have happened on its median string s_m. If the resulting string $s_m^{'}$ has distance greater than $3d$ to the set S, then it is impossible to generate centers by further mutating $s_m^{'}$'s positions of type A.*

Proof. Constraint 2 can be simply induced from Constraint 1.

CONSTRAINT 3: *Given a set S, assuming no or a few mutations have happened on its median string s_m. If the resulting string $s_m^{'}$ has distance greater than $(3d-1)$ from the set S, then it is impossible to generate centers by further mutating $s_m^{'}$'s positions of type B.*

Proof. The underlying reason is that a mutation happening at a type-B position will increase $s_m^{'}$'s distance to the set S by at least 1. It is proved in what follows. Without loss of generality, in each column of the set of strings S, the 4 bases can be categorized according to the number of their occurrences. In a type-B column, there exist one base with two occurrences, one base with one occurrences and other two bases with no occurrences. Assuming a mutation happen in a type-B position, it means that the current base (with two occurrences) is replaced either by the one with one occurrence or by one of the two bases with no occurrences. This causes the number of mismatches between $s_m^{'}$ and S increased by either 1 or 2.

If $dist(s_m^{'}, S) > 3d-1$, the further mutated median string $s_m^{''}$ will have distance $dist(s_m^{''}, S) > 3d$. It follows that $s_m^{''}$ cannot be a center of

S. △

CONSTRAINT 4: *Given a set S, assuming no or a few mutations have happened on its median string s_m. If the resulting string $s_m^{'}$ has distance greater than $(3d-3)$ from the set S, then it is impossible to generate centers by further mutating $s_m^{'}$'s positions of type C.*

Proof. The underlying reason is that a mutation happening at a type-C position will increase $s_m^{'}$'s distance to the set S by at least 3. Similar to the proof of Constraint 2, the 4 bases in a type-C column can be categorized into two categories: one base with three occurrences, the other three bases with no occurrences. Assuming a mutation happen in a type-C position, it means that the current base (with three occurrences) is replaced by one of the three bases with no occurrences. This causes the number of mismatches between $s_m^{'}$ and S increased by 3.

If $dist(s_m^{'}, S) > 3d-3$, the further mutated median string $s_m^{''}$ will have distance $dist(s_m^{''}, S) > 3d$. It follows that $s_m^{''}$ can not be a center of S. △

With the use of the above constraints, we can avoid some of the calls of the recursive procedure which are destined to generate no centers.

In addition, most of computational cost of the algorithm are related to checking constraints. We use a few hamming distance tables during the recursive procedure to enhance its performance. Details are follows. Before starting the recursion, we build a table containing the distances of the media string s_m to all the three given strings. During each recursion, the table of distances can be easily kept updated since at most one mutation is allowed, and the copy of the updated table will be passed to the next recursion for further use. In this way, all the constraints can be

efficiently checked.

3.3 CMMF - Constraint Mechanism-based Motif Finding Algorithm

The basic idea of CMMF is to find a set of three motif instances implanted in three sequences of E so that actual motifs can be derived as local patterns, then all derived patterns are checked to identify those actual motifs. Note that unlike most of the existing motif discovery algorithms, we are able to infer the centers (motif candidates) using the constraint mechanism without knowing the complete set of motif instances. Below, we describe the algorithm in detail.

Let $I = \{I_1, \ldots, I_t\}$ be the set of implanted motif instances. The motif discovery problem can be simplified as finding three elements of the set I. This can be accomplished by the use of techniques related to finding cliques (a clique is a set of vertices such that there exists an edge between every pair of the vertices) in multipartite graph. These techniques have been applied by Pevzner and Sze [35] to develop the WINNOWER algorithm. To facilitate the application of the clique finding techniques, we need to represent the motif finding problem in a simple geometric framework first. Recall that $E = \{E_1, \ldots, E_t\}$ is the set of t sequences which are implanted with the motif M. Given the parameter l (the length of the motif) and the parameter d (the maximum number of mismatches), we can construct a graph $G(E, l, d)$ as follows. For every position p in the sequence E_i, we construct a vertex E_{ip} representing the length-l substring starting at position p in E_i. Connect vertex E_{ip} with vertex E_{jq} by an edge if $i \neq j$ and the hamming distance between E_{ip}

Algorithm 3 CMMF

```
1: Q ← ∅;
2: Choose 3 sequences E_1, E_2, E_3;
3: for each E_1i ,E_2j and E_3k (1 ≤ i, j, k ≤ n − l) do
4:    if the distance between any two edges is no greater than 2d then
5:       Build clique cli = { E_1i, E_2j, E_3k } ;
6:       CLIQUE_CONVERSION( cli );
7:    end if
8: end for
```

and E_{jq} does not exceed 2d.

Since every sequence in E contains a motif instance of M, let $p = \{p_1, \ldots, p_t\}$ be the set of positions where p_i is the position of the motif instance in sequence E_i. Let $V = \{E_{1p_1}, E_{2p_2}, \ldots, E_{tp_t}\}$ be the subset of vertices in $G(E, l, d)$ representing the t motif instances. Every pair of vertices in V should have an edge in the $G(E, l, d)$ graph, therefore, the set V corresponds to a clique of size t in $G(E, l, d)$. In most existing algorithms, the (l, d)-motif problem is formulated as finding large cliques in a graph. There are two approaches to find cliques of size t. One approach is to explore the forest of edges (Hertz and Stormo [20]). The other approach is to remove edges that surely are not contained in large cliques (Vingron and Argos [50]; Vingron and Pevzner [51]; Pevzner and Sze [35]). No matter which approach to be used, the cost of finding cliques of size t is inevitably huge. However, in our algorithm, finding large cliques of size t is no longer the goal. Instead, finding a 3-clique (size-3 clique) which consist purely of motif instances is the target. The algorithm is illustrated in Algorithm 3 through finding all the cliques in the first three sequences.

After extracting the centers from a 3-clique through the use of constraint mechanism, we need to test if each center is an actual motif. The

details of center testing will be given in Section 6. In summary, the CMMF starts from identifying the complete set of 3-cliques in the three sequences, which can be randomly chosen from E. Then constraint mechanism is used to convert every clique into a set of centers and test if any of them is an actual motif. To do it in a memory-effective way, once a clique is identified, the constraint mechanism can be applied, followed by the verification of the resulted centers. In this way, nothing except the motifs will be kept in the memory.

3.4 Constraint Rules

In this section, we present another way to implement constraint based motif model. It is based on the use of constraint rules.

Compared with the constraint mechanism, a constraint rule targets at handling special sets of strings with enhanced efficiency, while still guaranteeing to find all the centers. The underlying reason for the difference is that the constraint mechanism need to test quite a few strings to identify centers among them, however constraint rules enumerate centers straightforwardly.

To some extent, each constraint rule can be considered as a "refined" constraint mechanism for special sets of strings. Section 2 demonstrated the constraint mechanism, that is, Algorithm D recursively tests if the string is a center or not, then it moves around to its neighborhood string. In this way, all the feasible centers are generated. In contrast, constraint rules can straightforwardly generate all centers. To do this, all it need is the apriori knowledge that, for each center, which combination of mutations transform the median string to the center.

Table 3.1: Base selection rules for computing a median string

Column Type	Chosen Base
A	b_1
B_1	b_2
B_2	b_1
B_3	b_1
C	b_1

Each constraint rule can only deal with a special type of string sets. In our implementation, we distinguish different types based on the two criteria: First, three pairwise hamming distances $dist(s_i, s_j)$,$1 \leq i, j \leq 3$; Second, number of columns of type A. The reason is that, the other information, namely number of columns of type C and $B_i(1 \leq i \leq 3$), can be determined given the above information.

At what follows, we use an example to illustrate how the constraint mechanism is refined to a constraint rule for handling a specific type of string sets. The refinement process will be general enough to be applicable to any case.

Refinement. Consider $S = \{s_1, s_2, s_3\}$ be a set of three length-l sequences such that $dist(s_1, s_2) = 2d - 1$,$dist(s_1, s_3) = 2d - 1$,$dist(s_2, s_3) = 2d - 2$ and number of type-A columns of the set S is 2. Such a string set has $(2d + 5)$ corresponding centers. At what follows, we will identify those combinations of mutations leading to these centers.

Without loss of generality, let b_i denote a base that occurs in the string s_i at a column, and b_0 denote a base with no occurrences at a column. A median string can be computed according to the rules listed in Table 3.1.

With a determinate median string, the next step is to identify concrete combinations of mutations that change the median string to centers. Let

Table 3.2: Combinations of mutations leading to centers. *Every cell in the first column presents a combination of mutations. And the blank cell means no mutations, namely, the media string s_m itself is a center. The second column presents the hamming distance between mutated median string $s_m^{'}$ and s_i, in the form of $\{dist(s_m^{'}, s_1), dist(s_m^{'}, s_2), dist(s_m^{'}, s_3)\}$. The third column shows the number of unique centers a combination of mutations can yield.*

Mutation	*dist*	n_c
	$\{d-1, d, d\}$	1
$A[b_1 \Longleftarrow b_2]$	$\{d, d-1, d\}$	2
$A[b_1 \Longleftarrow b_3]$	$\{d, d, d-1\}$	2
$A[b_1 \Longleftarrow b_0]$	$\{d, d, d\}$	2
$A[b_1 \Longleftarrow b_2]$ $A[b_1 \Longleftarrow b_3]$ $B_1[b_2 \Longleftarrow b_1]$	$\{d, d, d\}$	$2(d-1)$

$b_i \Longleftarrow b_j$, denote a mutation in the form that the base b_i is replaced by the base b_j. Let $Y[mutation]$ specify the location of a particular mutation is at the type-Y column of the median string. e.g., a mutation, in the form that the current base b_1 is substituted by the base b_2 in a type-B_2 column, can be denoted as $B_2[b_1 \Longleftarrow b_2]$. The resulted changes of the mutation to the hamming distances between the median string s_m and the string s_i is that, both $dist(s_m, s_1)$ and $dist(s_m, s_3)$ increase by 1, while $dist(s_m, s_2)$ decrease by 1. Note that, given a mutation form $Y[b_i \Longleftarrow b_j]$, the number of possible mutations of this form can be decided, that is, it equals to the number of non-mutated columns of type Y. This forms the basis to calculate the number of centers which a combination of mutation forms can possibly yields.

Table 3.2 lists those combinations leading to centers. The hamming distance between a string s and every string in S can be easily computed if the base of s in each position is known as b_i, $i \in \{0, 1, 2, 3\}$. Thus, the hamming distance from the median string s_m or the mutated median

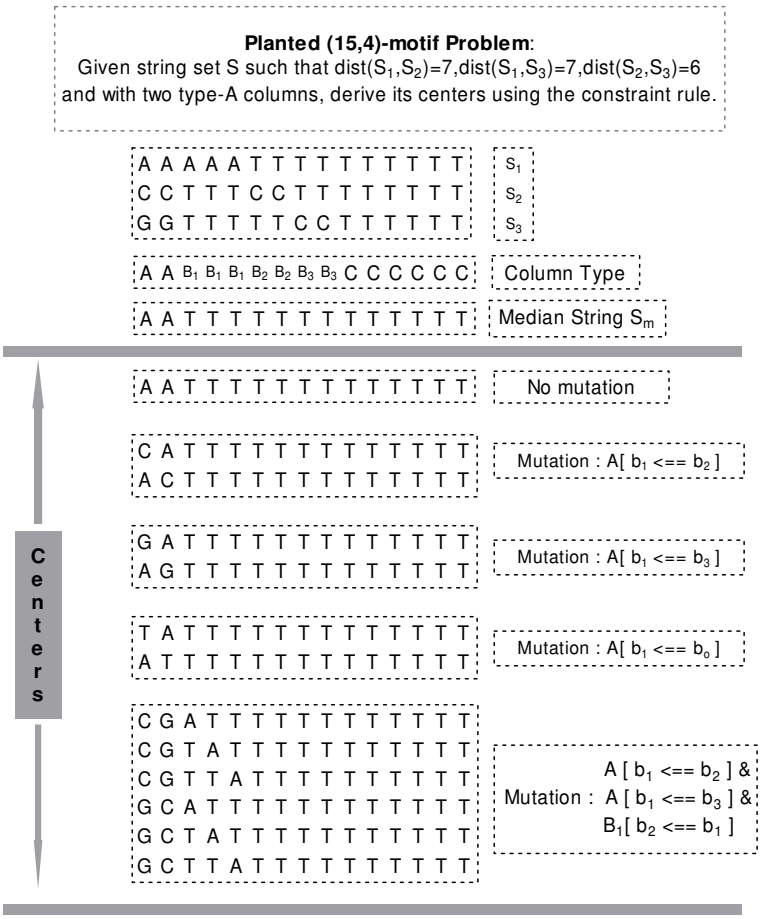

Figure 3.1: Center Collection

string s'_m to the three strings in the set S can be computed. In Figure 3.1, we provide a concrete example, and show how to use the constraint rule provided in Table 3.2 to derive all the centers in this case.

We can use the above method to identify the combinations of mutations that lead to centers for any type of string sets. A constraint rule incorporates these combinations as a package so that it is able to convert a particular type of string sets into centers straightforwardly. Compared with constraint mechanism, the efficiency of constraint rules has been improved greatly due to that those combination of mutations which do not lead to centers have been eliminated.

Table 3.3: Number of centers for those types of string sets which we choose to refine the constraint mechanism to constraint rules.

Type of S: $D(S)$, n_A		number of centers
$D(S)$	n_A	
$2d, 2d, 2d$	0	1
$2d, 2d, 2d-1$	1	1
$2d, 2d, 2d-2$	0	$d+1$
	2	1
$2d-1, 2d-1, 2d-1$	1	4
	3	6
$2d-1, 2d-1, 2d$	0	1
	2	2
$2d-1, 2d-1, 2d-2$	0	$3d+1$
	2	$2d+5$
	4	12
$2d-2, 2d-2, 2d-2$	Too many centers	
$2d-2, 2d-2, 2d$	0	d^2+1
	2	$2d$
	4	6
$2d-2, 2d-2, 2d-1$	Too many centers	
$2d, 2d-1, 2d-2$	1	$d+1$
	3	3

Note that there exist many cases where the output may contain too many centers. Refining the constraint mechanism for such cases costs too much. To strike a balance between the heavy effort on refining the constraint mechanism and the enhanced efficiency of pattern extraction procedure, we choose to refine the constraint mechanism for limited types of string sets. The choice of these types is based on the following two criteria: First, these types can cover most string sets on which constraint mechanism is applied; Second, the number of derived centers for such a type of string sets is tractable, e.g. less than certain threshold. Accordingly, for finding motifs in randomly distributed background sequence, we consider those types of string sets in which pairwise hamming distances range from $(2d-2)$ to $2d$ and number of the corresponding centers does

not exceed 20. The list of these types as well as corresponding number of centers are given in Table 3.3.

3.5 CRMF - Constraint Rules-based Motif Finding Algorithm

This section describes the algorithm CRMF which fully exploit the power of constraint rules.

Not as general as constraint mechanism, every constraint rule focuses on a particular type of string sets. And we just generate constraint rules for limited cases. Thus we can no longer expect these constraint rules can handle any set of three strings. The natural solution is to apply constraint mechanism to those cases where constraint rules are not applicable. However, there exists a better way to fully exploit the power of constraint rules given the assumption that every sequence contains at least one motif instance.

The idea is to find a set of three motif instances which can be converted into centers using constraint rules. Let a *planted* clique denotes a clique comprised of only motif instances. Assuming a planted 3-clique have been found, but unfortunately, centers can not be generated by existing constraint rules. Then, we can extend this clique to a planted 4-clique by incorporating one more motif instance. If still unlucky, no constraint rules are applicable to any three motif instances in the clique. We keep on extending the planted clique in this way until we find at least one set of three motif instances in the clique falls to any case listed in Table 3.3. Then the clique can be converted into centers using Algorithm 4: Apply the constraint rule to convert the three motif instances into a set

of centers, test every center locally to see if it satisfies the constraint that the hamming distance between the center and every other motif instance in the clique should be at most d. Those centers passing the testing process successfully are the centers of the clique. And they will be further tested globally to identify the actual motifs.

As a clique expands, the chance of conversion will increase linearly. Assuming a n-clique expands to $(n+1)$-clique, the chance of conversion will increase by $(\frac{3}{n-2})$ due to that the number of size-3 sets of strings in the clique increase from C_n^3 to C_{n+1}^3. It indicates that most cliques are converted into centers when their sizes are small. This is the main reason why the algorithm is faster than all existing solutions.

By a **convertible clique**, we mean 3 vertices of the clique meet a application requirement of a constraint rule so that it can be converted into centers through using constraint rules. In the luckiest case, even the size of the clique is only 3, we already can find a convertible planted clique and extract the actual motif.

Intuitively, CRMF algorithm begins with finding a set Q consisting of all the 2-cliques(cliques of size 2) in the first two sequences E_1 and E_2. Then, we iteratively enlarge the cliques in Q with the remaining sequences in E. For the enlarged cliques which are convertible, we imme-

Algorithm 4 Clique Conversion Algorithm

```
1: CLIQUE_CONVERSION ( cli )
2: Generate the set C of centers through applying a constraint rule to 3
   strings in cli;
3: for each center c in C do
4:    if c has distance greater than d to any string in cli then
5:       Remove c as from C ;
6:    end if
7: end for
8: Report C as the center set of the clique cli ;
```

diately convert them into centers and test if they are actual motifs. The remaining enlarged cliques are inserted into Q. The process is iterated until Q is empty or we finished processing all t sequences in E. If some t-cliques are still remained in Q after the whole process, we can force the conversion through using constraint mechanism, and the centers of the t-clique are the actual motifs directly. However, this case rarely happens. The whole procedure is illustrated in Algorithm 5. Experiments on synthetic data have shown that those constraint rules are very powerful in converting cliques. Even in the difficult case of (15,4)-motifs implanted in twenty length-3000 sequences, all cliques can be converted into centers before their size reach 11. While in cases, such as (15,4)- and (9,2)-motifs implanted in twenty length-600 sequences, all cliques can be converted before their size reach 7.

The cliques of size 3 can be considered as seed cliques. The whole procedure can be considered as the iterative extending cliques by incorporating vertices and then filtering cliques through using constraint rules. And the actual motif is ensured to be preserved either in the form of a center or in the form of a clique which consists of a subset of motif instances.

3.6 Implementation Issues

A naive implementation of the rules-based algorithm is not efficient and uses much memory. This section demonstrates a number of techniques implemented to reduce the computational cost and to employ memory efficiently.

Algorithm 5 CRMF Algorithm

```
1: Q ← ∅;
2: for each E_1i and E_2j (1 ≤ i, j ≤ n − l) do
3:    if dist(E_1i, E_2j) ≤ 2d then
4:       {E_1i, E_2j} is a 2-clique in G(E, l, d) and
         Insert {E_1i, E_2j} to Q;
5:    end if
6: end for
7: for each m = 3 . . . t do
8:    if Q = ∅ then
9:       Break;
10:   end if
11:   for each (m − 1)-clique cli in Q and each vertex E_mj do
12:      Remove cli from Q;
13:      if cli = cli ∪ {E_mj} is a m-clique in G(E, l, d) then
14:         if cli is a convertible clique then
15:            CLIQUE_CONVERSION( cli );
16:         else
17:            Insert cli into Q;
18:         end if
19:      end if
20:   end for
21: end for
22: for each remaining t-clique cli in Q do
23:   Compute the motif of cli by Constraint Mechanism
24: end for
```

3.6.1 Hamming Distance Matrix

Similar to the dot-matrices used by Vingron and Pevzner [51], the hamming distance matrix between sequences E_i and E_j can be considered simply as a matrix, where the entry at position (p, q) denotes the hamming distance between the vertices E_{ip} and E_{jq} (definition for vertex is given in Section 3). For simplicity, the hamming distance matrix is denoted as the HD matrix below. Due to the fact that an edge may appear in many cliques, hamming distances between two vertices may be computed many times in the course of finding cliques. To avoid the redundant computation of hamming distances, we store those hamming distances in

HD matrices.

Apart from reducing the computational cost, HD matrices also save the memory resource. In the case where the sequences in E are long, the number of cliques may be a lot. For instance, suppose the sequences in E are of length n. Then, the number of possible cliques in the first m sequences E_1,E_2, ..., E_m can be as big as O($(n-l+1)^m$). Instead of storing all cliques, we can just store the HD matrices for every pair of these m sequences. Then, the cliques in the m sequences can be recovered one by one by scanning the HD matrices. By this method, we only need to store $\binom{m}{2}$ HD matrices, which takes O($\binom{m}{2}(n-l+1)^2$)space.

3.6.2 Clique Conversion Threshold

In some cases, a clique may generate so many centers that the cost for validating these centers exceeds the total cost for extending it and then testing less centers of the extended ones. Therefore, we add the clique conversion threshold T to prevent such cases from happening. Precisely, if the number of centers converted by a clique is greater than T times the size of the clique, we will extend the clique instead of converting it into too many centers. Experiments show that an appropriately chosen value for T can reduce the running time effectively.

3.6.3 Duplicated Centers Elimination

Statistics show that above 10 percent of derived centers are duplicates in case of randomly distributed background sequences. To avoid redundant center testing, hashing is used to eliminate these duplicates. The hash function we choose is related to integer encoding (Myers [32]). Let

$s = s_1 s_2 \cdots s_l$ be a DNA sequence of length l, $s_i \in \{a, c, g, t\}$. Let $\Phi(a)$,$\Phi(c)$,$\Phi(g)$ and $\Phi(t)$ be numbers 0, 1, 2 and 3 respectively. The hash function is $\sum_{i=1}^{l} \Phi(s_i)4^{i-1}$. The number of buckets required is 4^l. Observe that the size of this hash table increases exponentially in the sequence length l. To control the memory usage of the hash table, in our implementation, we choose to hash the first 9 bases of a center to generate a hash key K. The details are below. For every center, we hash its left-most 9 bases to generate a hash key K and hash right-most $(l-9)$ bases to generate a value V. If V exists in the Bucket K, the center is a duplicate; otherwise, we store the value V in the Bucket K and state that this is a new center.

3.6.4 Center Testing

In our implementation, we use two center testing methods.

[Method 1]. The first method is to find one similar substring of a center in each sequence. However, if it fails in any sequence, it will deny that the center is an actual motif and the testing procedure will stop at once. In this way, most of centers can be filtered out without testing them against all sequences. However, this method is based on the assumption that every sequence contain at least one motif instance. This method is suitable when the motif is well conserved in every nucleotide sequence.

[Method 2]. The second method is to find all similar substrings of a center in all the sequences. According to a predefined threshold, we determine the center is actual or not. The threshold can simply be the sum of the estimated number of random similar substrings and the expected number

of motif instances. This method is applicable to most cases although it is a little slower than the first one.

Both methods make use of the Pigeon Hole principle to speed up the searching of similar substrings. The principle has been used by Sung and Lee [46] to develop fast probe selection algorithm. Below is the key lemma.

Lemma 1: If there exist length-l strings p and q such that $dist(p, q) \leq d$, then we can find length-k substrings $p' = p[i \ldots i + k - 1]$ and $q' = q[i \ldots i + k - 1]$ such that $dist(p', q') \leq v$ where $v = \lfloor \frac{dk}{l} \rfloor$.

Proof. Given p and q where $dist(p, q) \leq d$, there are at most d mismatches between p and q. These mismatches are distributed across $\frac{l}{k}$ pairs of length-k substrings $(p[ik \ldots (i+1)k-1], q[ik \ldots (i+1)k-1])$ in p and q, for $i = 0, 1, \ldots, \frac{l}{k}$. By the Pigeon Hole principle, at least one pair of the length-k substrings has at most $v = \lfloor \frac{dk}{l} \rfloor$ mismatches. The lemma follows. ∘

A length-l center c can be divided into $\lceil \frac{l}{k} \rceil$ length-k substrings c_i where $c_i = c[ik \ldots (i+1)k-1]$ for all $i = 0, 1, \ldots, \lfloor \frac{l}{k} \rfloor - 1$ and $c_{\lfloor \frac{l}{k} \rfloor} = c[l - k \ldots l-1]$. Similarly, any length-$l$ substring s can be divided into length-k substrings s_i for all $i = 0, \ldots, \lceil \frac{l}{k} \rceil - 1$. Below, to simplify the discussion, we assume l is divisible by k. By Lemma 1, only if $dist(c_i, s_i) < v$ for some $i \in \{0, \ldots, \frac{l}{k}\}$, it is possible that $dist(c, s) \leq d$. Based on this idea, those length-l substrings, which are for sure not similar to the center c, can be filtered out. The center testing algorithm based on the above idea is shown in Algorithm 6. For any two length-k substrings s_1 and s_2, we say they form a *hit* if $dist(s_1, s_2) \leq v$. The testing procedure can be speeded up greatly by building a hashing table which stores positions of

Algorithm 6 Center Testing Algorithm

1: **CENTER_TEST**(center c)
2: **for** each sequence $E_i \in E$ **do**
3: Compute $dist(c, s)$ for those s in E_i whose $dist(c_i, s_i) \leq v$
4: **if** $disc(c, s) \leq d$ **then**
5: Report center c successfully finds a "similar substring" in sequence E_i;
6: **end if**
7: **end for**
8: Report those centers who successfully finds a "similar substring" in every sequence $E_i \in E$ as motifs;

hits in sample sequences for all 4^k length-k substrings.

The probability p that $dist(c_i, s_i) \leq v$ equals $\sum_{i=0}^{v} C_k^i (\frac{3}{4})^v (\frac{1}{4})^{l-v}$. Thus, in every sequence of length n, the expected number of similar substrings indicated by c_i is $(n-k+1)p$. Every center c has $\frac{l}{k}$ length-k substrings. Therefore the number of substrings whose hamming distances with center c need to be computed is $\frac{l}{k}(n-l+1)p$ per sequence. Compared with the brute force approach through which we need to compute hamming distances between every substring ($n-l+1$ in total) and the center c, we get the complexity reduced by $100(1-\frac{lp}{k})\%$.

Chapter 4

Experimental Results

In this chapter, we investigate CRMF and CMMF's capability of finding motifs in both synthetic and real-life data. All the experiments were taken on a 2.4GHz P4 CPU. The generation of synthetic data follows the rationale of (l,d)-motif problem with randomly distributed background sequences. First, we chosen a length-l motif consensus m by picking l bases at random. Second, we generate a motif instance by randomly choosing d positions of m and mutate the base at each chosen position to a different, randomly chosen base. Through repeating the second step t times, we can get t motif instances. Third, we construct t background sequences of length n using $n \times t$ randomly chosen base. Finally, we assign each motif instance to a random position in a background sequence, one motif instance per sequence.

In Section 1, the synthetic (15,4)-motifs which are implanted in twenty sequences of length-n, where n is a parameter to determine the difficulty of motif finding, are designed to test the performance of CMMF and CRMF. And the comparison are also presented. In Section 2, we compare the performance of CRMF with other leading motif finding algorithms, such as

PROJECTION and MULTIPROFILER over some widely accepted challenging motif finding problems. In Section 3, we present the statistical limits of (l,d)-motif finding to prove that CRMF already reach the performance limit. In Section 4, we test the algorithms using some biological samples.

4.1 Performance of CMMF and CRMF on Synthetic Data

The comparison is to show the superiority of constraint rules relative to the constraint mechanism. It is assumed that every sequence contains at least one motif instance. Therefore we choose Method 1 (see Chapter 3.6.4) to test centers.

To compare the performances of both CMMF and CRMF, we apply them to find (l,d)-motifs, with l and d are fixed, implanted in background sequences of increasing length. Precisely, we apply the two algorithms to find the (15,4)-motif implanted in twenty length-n sequences, where n range from 500 to 3000. Due to that both CMMF and CRMF can find all implanted motifs with success rate 100%, we do not present the experimental results in terms of commonly used performance coefficient, which mainly measures algorithm accuracy. Instead, the performance is measured using running time. Their performances are shown in Figure 4.1 in terms of running time. The running time of both algorithms increase exponentially with the increasing sequence length. It is explicit that the running time of CRMF is always as almost half as that of CMMF. It indicates that the derived constraint rules from constraint mechanism

has improved the performance almost by a half, while without sacrificing accuracy.

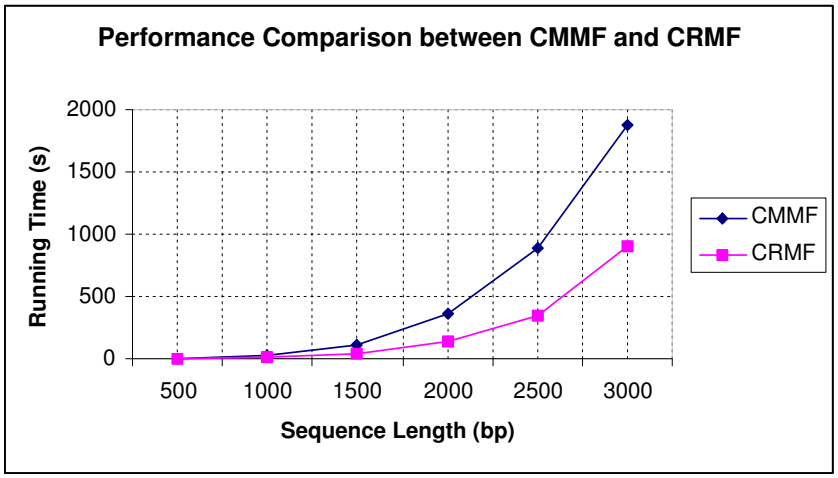

Figure 4.1: The Performance Comparison between CMMF and CRMF. The performance of two algorithms are shown when exploring the (15,4)-motifs implanted in twenty n-bp sequences. Average running time are taken over 5 independent experiments.

Given a specific motif problem, the determinant parameter of CRMF's performance is the clique conversion threshold T, introduced in Chapter 3.6.2. Experiments show that a careful chosen T have positive impact on the performance.

Note that the maximum number of centers a constraint rule can generate is 17 based on Table 3.3. So when T reaches 4, it is implied most of constraint rule are allowed to be applied in the phrase of small clique size. We therefore chose value 1,2,3,4 for T to observe the variance of algorithm performance. The results are shown in Figure 4.2. The performance difference with different T is not distinct since their running time always differ by only seconds or tens of seconds. However, it is still explicit that the performance of CRMF is maximized when conversion threshold T equals 3.

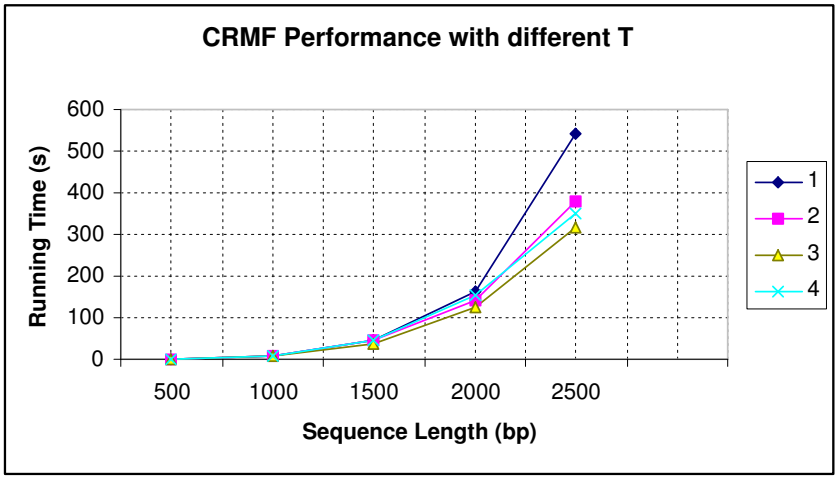

Figure 4.2: The Performance of CRMF with varying clique conversion threshold T. *CRMF is tested against (15,4)-motifs implanted in twenty n-bp sequences. Average running time in the case of a specific sequence length are taken over 5 independent experiments. CRMF achieved the best performance when T is set to 3.*

4.2 Challengeing Problems on Simulated Data

Keich and Pevzner [25] has systematically studied the limits of motif finding algorithms. "Subtle motif" is used to describe a motif that is almost indistinguishable from random motifs in the statistical sense. As for (l,d)-motif problem, "subtle motif" is the consequence of too long background sequences. When the length n of background sequences reach certain point, the motif becomes too subtle and one is then likely to encounter random motifs which are at least as significant statistically as the implanted motif itself. In this section, we explored the underlying cause of motif subtleness statistically. Then we show the robust performance of our algorithm on handling those extremely challenging problems.

Let $p_d = \sum_{i=0}^{d} C_l^i (\frac{3}{4})^i (\frac{1}{4})^{l-i}$ be the probability that a given length-l string s occurs with up to d substitutions at a given position of a random sequence. Then the expected number of length-l motifs that occur with up

Table 4.1: Average performance of CRMF on synthetic data. *Each row in the table shows the average performance of CRMF over five independent experiments for finding a subtle (l, d)-motif implanted in twenty length-n sequences. T is the clique conversion threshold used in each five experiments. $E^{'}(l, d, n)$ ([7]) is the expected number of random (l, d)-motifs generated in the sequences. And n_M is the average number of detected motifs. All the experiments were taken on a 2.4GHz P4 CPU.*

l	d	n	$E^{'}(l, d, n)$	T	n_M	$Time$(s)
9	2	600	2.6	5	2.2	3
11	3	600	5.7	4	6.2	9
15	4	3000	1.02	3	1	992
15	4	4000	3.96	3	4.4	3271

to d substitutions at least once in each of t random length-n sequences is approximately $E(l, d, n) = 4^l(1 - (1 - p_d)^{n-l+1})^t$. This expectation is only an estimate due to that overlapping occurrences of a given consensus string do not occur independently in the background.

Using the above formula, we can identify some challenging problems. For instance, twenty random length-600 sequences, with no planted motif, are expected to contain more than one (9,2)-motif and more than four (11,3)-motifs by chance.

We ran CRMA on sets of challenging planted (l,d)-motif problems. The algorithm's performance on these sets is also reported in Table 4.1, including the expected number of motifs $E^{'}(l, d, n)$, the average running time $TIME$ and the chosen clique conversion threshold $Threshold$. Due to that one motif has implanted into the background sequence, the expected motifs $E^{'}(l, d, n)$, including the planted one and random ones, should be $(E(l, d, n) + 1)$. Table 4.1 presents only the performance results of CRMA associated with the appropriate choice of $Threshold$ which minimizes the running time in each case. The results confirmed that our algorithm never failed to include the actual motif in the set of detected

motifs. Moreover, the number of detected motifs by our algorithm is very close to $E'(l, d, n)+1$. This means that the algorithm can find the planted motif together with all the random motifs.

4.3 Benchmarking

CONSENSUS (Hertz and Stormo [20]), GibbsDNA (Lawrence et al. [27]) and MEME (Bailey and Elkan [5]) successfully detected (15, 3)-motifs implanted in twenty 600 bp (abbreviation for base pair) sequences, but they all failed to find the (15, 4)-motifs implanted in twenty 600 bp sequences. Both WINNOWER and SP-STAR (Pevzner and Sze [35]) succeeded in solving the above motif problem, but SP-STAR failed to solve the (15, 4)-motifs implanted in 1000 bp sequences and WINNOWER failed to work on the (15, 4)-motifs implanted in twenty 1300 bp sequences. PROJECTION (Buhler and Tompa [7]) succeeded in 16 out of 20 times in detecting the same (15, 4)-motif implanted in twenty 2000 bp sequences. And PROJECTION also solved the difficult planted (14, 4)-, (16, 5)- and (18, 6)-motifs implanted in twenty 600 bp sequences. However it failed to solve more difficult planted (9, 2)-, (11, 3)-, (13, 4)-, (15, 5)-, or (17, 6)-motifs.

MULTIPROFILER (Keich and Pevzner, [24]) can successfully detect in more than 99% of the time the (15, 4)-motifs implanted in 2000 bp sequences. It also can detect more than 98% of such motifs implanted in twenty 3000 bp sequences. In the case of a (9,2)-motif implanted in twenty 600 bp sequences, it included the implanted motif among the patterns with success rate 100% which are extracted according to a score function, while PROJECTION succeeded in only 16 out of 20 cases using

a same score function.

The above data shows that MULTIPROFILER is the best in terms of the sensitivity and reliability among existing motif discovery algorithms. We compared our algorithm with MULTIPROFILER. As shown in the results we present, it is implicit that our algorithm always has success rates of 100%. Since MULTIPROFILER has success rates close to 100% in most cases, we focus on the comparisons of running time between them. For (15,4)-motifs implanted in twenty 2000 bp sequences, MULTIPROFILER found them in about 75 minutes(on a 500 MHz G4), while we are able to detect them in 3 minutes(on a 2.4GHz P4). For (15,4)-motifs implanted in twenty 3000 bp sequences, MULTIRPOFILER found them in about 3 hours(on a 500 MHz G4), however we are able to detect them in 13 minutes(on a 2.4GHz P4). For (9,2)-motifs implanted in twenty 600 bp sequences, MULTIPROFILER included them among the patterns in less than a minute(on a 500 MHz G4), whereas we are able to do it in 3 seconds(on a 2.4GHz P4). Moreover, we can detect the (15, 4)-motif implanted in twenty 4000-long sequences in less than one hour. The detail is given in Table 4.1.

4.4 Finding Motifs in Realistic Biological Data

In algorithm CRMF, both techniques, namely clique extension and Method 1 for center testing, are based on the assumption that each sequence contains at least one motif instance. So the existence of more than one motif instance(invaded samples) in each sequence is tolerated by CRMF. However, the existence of none motif instances (corrupted samples) in

any sequence is not tolerated by CRMF, which is exactly the algorithm's weak point.

In algorithm CMMF, if Method 2 for center testing is deployed, invaded samples as well as corrupted samples can be tolerated. The only prerequisite for CMMF's success is that each of the three sequences used to build 3-cliques contains at least one motif instance. However, this prerequisite cannot be guaranteed when it handling corrupted samples. One approach to overcome this shortcoming is to iterate CMMF so that the chance of selecting 3 sequences each of which contains motif instances is increased. Assume $T^{'}$ out of T sequences contain at least one motif instance in a sample, if we iterate CMMF m times, with randomly chosen 3 sequence to build 3-cliques every time, the probability that none of iterations succeed is : $1-(1-(\frac{T^{'}}{T})^3)^m$. Therefore, given $T^{'}$ and T, we can set an appropriate value for m so that the success rate can be kept at a high level. These modifications have been made for CMMF so that it can find biological motifs effectively.

To test the performance of our algorithm, we examined orthologous sequences from a variety of organisms taken from regions upstream of four types of genes: preproinsulin, dihydrofolate reductase(DHFR), gametic lethal(GAL) and yes-associated protein(YAP). Experiments on finding exact known motifs have been successfully conducted in the above samples. However, in Table 4.2, we only present length-15 motifs found by our algorithm to demonstrate its flexibility. The details of the above biological examples can be found in [44] and [49].

Table 4.2: Performance on biological data. *We only look for (15,2)-motifs. Input Size is the total number of base pairs in samples. Reference motifs are either established biological motifs or those found by alternative algorithms. (1) was taken from Stormo and Hartzell ([44]), (2) from van Helden and Andre [49].*

Sequence	Input Size	Found motif of length 15	Published motif	Ref.
Preproinsulin	7689	tgcAGACCCAGCAcc	AGACCCAGCA	(1)
		CCTCAGCCCCctgcc	CCTCAGCCCC	(1)
DHFR	800	taaATTTCacGCCAa	ATTTCnnGCCA	(1)
		cgtggGGGCGGGGCC	GGGCGGGGCC	(1)
		gaTTCGCGCCAAACT	TTCGCGCCAAACT	(1)
GAL	5600	CGGtagCCGcaacaa	CGGnnnCCG	(2)
YAP	12800	attgtgaTTACTAAt	TTACTAA	(2)

Chapter 5

Conclusions and Open Problems

In this thesis, we have described two algorithms CMMA and CRMA for finding motifs using constraint based methods. They were designed to efficiently solve problems from the planted (l,d)-motif model. Experiments on synthetic data show our algorithms outperform the existing motif finding algorithms in terms of both reliability (how well does it find motifs) and complexity (at what cost).

The complexity of both algorithms mainly depends on the number of cliques generated in the whole procedure. So its performance can be further improved by generating fewer cliques, however, at the risk of losing the cliques which consists of motif instances. We are looking for appropriate approaches to apply constraint rules which generate fewer cliques with success rates kept high, although not 100%. In addition, our algorithms are developed based on FM mode (Stormo [43]) where each sequence contains one instance of an (l, d)-motif. Further work is needed to make it work on synthetic data in VM mode (Pevzner and Sze [35]) where each

sequence contains one instance, each position of which is mutated with probability p.

Although the constraint based method have reached the theoretical limit for solving extremely difficult synthetic problems, the major questions about the methods focus on how to extend it to accommodate more features of real biological motif-finding problems. In particular, we pose the following three questions. Firstly, is it possible to build cliques whose vertices may be from the same partition(sequence) so that the multiple motif occurrences in single sequences in invaded samples can be better utilized? Secondly, how to extend CRMF to handle corrupted biological samples? Finally, can we any extend it to find motifs whose instances contain insertions and deletions, or contain spacers (sequences of N's)?

Bibliography

[1] The c. elegans sequencing consortium. genome sequence of the nematode c. elegans: a platform for investigating biology. *Science*, 282:2012–2018, 1998.

[2] The arabidipsis genome initiative. analysis of the genome sequence of the owering plant arabidopsis thaliana. *Nature*, 408:796–815, 2000.

[3] Genome international sequencing consortium. initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[4] N. M. L. A. P. Dempster and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

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

[6] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *Journal of Computational Biology*, 5:279–305, 1998.

[7] J. Buhler and M. Tompa. Finding motifs using random projections. In *Proceedings of the Fifth Annual International Conference on Research in Computational Molecular Biology*, RECOMB-01, pages 69–76, Montreal, Canada, April 2001. ACM Press.

[8] F. C. Chen and W. H. Li. Genomic divergences between humans and other hominoids and the effective population size of the common ances-

tor of humans and chimpanzees. *American Journal of Human Genetics*, 68(444):56, 2001.

[9] W. G. D. J. States and S. F. Altschul. Improved sensitivity of nucleic acid database searches using application-specific scoring matrices. *Methods: a Companion to Methods in Enzymology*, 3(1):66–70, 1991.

[10] R. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, New York, 1999.

[11] G. F. H. D. F. H. F. Sanger, A. R. Coulson and G. B. Petersen. Nucleotide sequence of bacteriophage lambda dna. *Journal of Molecular Biology*, 162(4):729–773, 1982.

[12] M. R. Fellows. Parameterized complexity: the main ideas and connections to practical computing. *LNCS*, 2285:262–273, 2002.

[13] Y. Fraenkel, Y. Mandel, D. Friedberg, and H. Margalit. Identification of common motifs in unaligned dna sequences: application to escherichia coli lrp regulon. *Comp. Appl. Biosci.*, 11:379–387, 1995.

[14] P. Fraser and F. Grosveld. Locus control regions, chromatin activation, and transcription. *Current Opinion in Cell Biology*, 10(3):361–365, 1998.

[15] M. Gelfand, E. Koonin, and A. Mironov. Prediction of transcription regulatory sites in archaea by a comparative genomic approach. *Nucleic Acids Res.*, 28:695–705, 2000.

[16] B. R. Glick and J. J. Pasternak. *Modecular biotechnology: principles and applications of Recombinant DNA*, chapter 3, pages 19–43. American Society for Microbiology, 1998.

[17] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. PhD thesis, Cambridge University, New York, NY, 1997.

[18] S. H. O and K. W. Wilcox. A restriction enzyme from hemophilus influenzae. i. purification and general properties. *Journal of Molecular Biology*, 51(2):379–391, 1970.

[19] M. B. e. a. H. W. Mewes, K. Albermann. Overview of the yeast genome. *Nature*, 387(6632 suppl.):7–65, 1994.

[20] G. Hertz and G. Stormo. Identifying dna and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, 15:563–577, Jul-Aug 1999.

[21] G. Z. Hertz and G. D. Stormo. Identification of consensus patterns in unaligned dna and protein sequences: a large-deviation statistical basis for penalizing gaps. In *Proceedings of the Third International Conference on Bioinformatics and Genome Research*, pages 201–216, Singapore, 1995. World Scientific Publishing Co.

[22] J. G. J. Alber and R. Niedermeier. Faster exact solutions for hard problems: a parameterized point of view. *Discrete Mathematics*, 229:3–27, 2001.

[23] D. M. M. W. M. R. A. G. J. C. Oeltjen, T. M. Malley and J. W. Belmont. Large-scale comparative sequence analysis of the human and murine bruton's tyrosine kinase loci reveals conserved regulatory domains. *Genome Research*, 7:315–329, 1997.

[24] U. Keich and P. Pevzner. Finding motifs in the twilight zone. In *Proceedings of the Sixth Annual International Conference on Research in Computational Molecular Biology*, RECOMB, pages 195–204, 2002.

[25] U. Keich and P. Pevzner. Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics*, 18(10):1382–1390, 2002.

[26] C. C. M. P. R. J. S. L. V. D. L. McCue, W. Thompson and C. E. Lawrence. Phylogenetic footprinting of transcription factor binding sites in proteobacterial genomes. *Nucleic Acids Research*, 29:774–782, 2001.

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

[28] C. E. Lawrence and A. A. Reilly. An expectation maximization (em) algorithm for the identification and characterization of common sites in

unaligned biopolyer sequences. *Proteins: Structure, Function and Genetics*, 7:41–51, 1990.

[29] B. Lewin. *Genes VI.* Oxford University Press, 1997.

[30] M. Li, B. Ma, and L. Wang. Finding similar regions in many strings. In *Proceedings of the 31st ACM Annual Symposium on Theory of Computing*, pages 195–204, Atlanta, Georgia, May 1999.

[31] e. a. M. D. Adams. The genome sequence of drosophila melanogaster. *Science*, 287:2185–2195, 2000.

[32] E. W. Myers. A sublinear algorithm fro approximate keywork search. *Algorithmica*, 12:345–374, 1994.

[33] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[34] G. Pesole, N. Prunella, S. Liuni, M. Attimonelli, and C. Saccone. Wordup: an efficient algorithm for discovering statistically significant patterns in dna sequences. *Nucleic Acids Research*, 20(11):2871–2875, 1992.

[35] P. Pevzner and S. H. Sze. 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, San Diego, CA, Aug 2000. AAAI Press.

[36] O. W. R. A. C. e. a. R. D. Fleischmann, M. D. Adams. Whole-genome random sequencing and assembly of haemophilus in uenzae rd. *Science*, 269:496–512, 1995.

[37] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences. *Bioinformatics*, 14:55–67, 1998.

[38] C. J. B. S. D. Sheridan and G. W. Hatfield. Activation of gene expression by a novel dna structural transmission mechanism that requires supercoiling-induced dna duplex destabilization in an upstream activating sequence. *Journal of Biological Chemistry*, 273(21):298–308, 1998.

[39] C. J. B. S. D. Sheridan and G. W. Hatfield. Inhibition of dna supercoilingdependent transcriptional activation by a distant b-dna to z-dna transition. *Journal of Biological Chemistry*, 274(81):69–74, 1999.

[40] J. Setubal and J. Meidanis. *Introduction to Computation Molecular Biology*. PWS Publishing Company, 1997.

[41] S. Sinha and M. Tompa. A statistical method for finding transcription factor binding sites. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 344–354, San Diego, CA, 2000.

[42] R. Staden. Methods for discovering novel motifs in nucleic acid sequences. *Computer Applications in Biosciences*, 5(4), 1989.

[43] G. Stormo. Dna binding sites: representation and discovery. *Bioinformatics*, 16:16–23, 2000.

[44] G. Stormo and G. Hartzell. Identifying protein-binding sites from unaligned dna fragments. In *Proceedings of the National Academy of Sciences*, volume 86, pages 1183–1187, USA, 1989.

[45] T. Strachan and A. P. Read. *Human Molecular Genetics*. BIOS Scientific Publishers, 2001.

[46] W. K. Sung and W. H. Lee. Fast and accurate probe selection algorithm for large genomes. In *Proceedings of IEEE Computer Society Bioinformatics Conference (CSB)*, Stanford, CA, 2003.

[47] S. T. F and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[48] M. Tompa. 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*, Heidelberg, Germany, 1999. AAAI Press.

[49] J. van Helden, B. Andre, and J. Collado-Vides. Extracting regulatory sites from the upstream region of yeast genes by computational analysis

of oligonucleotide frequencies. *Journal of Molecular Biology*, 281(5):827–842, 1998.

[50] M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *Journal of Molecular Biology*, 218:33–43, 1991.

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

[52] F. Wolfertsteeter, K. Frech, G. Herrmann, and T. Wernet. Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithm. *Computer Applications in Biosciences*, 12(1):71–80, 1996.

[53] C. H. Yuh, H. Bolouri, and E. H. Davidson. Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene. *Science*, 279:1871–1872, 1998.

Appendix A

Glossary

- ALIGNMENT: a correspondence between two or more sequences. Each column of an alignment matches up corresponding bases from each participating sequence. If a base in one sequence does not match any base in the other sequence, that base is matched to a gap character "–".

- AMINO ACIDS: the building blocks of proteins. A group of 20 different kinds of small molecules that link together in long chains to form proteins.

- BACKGROUND SEQUENCE: genomic sequence lacking any biologically meaningful feature, usually assumed to be under no selective pressure and, in sufficiently diverged organisms, to be completely uncorrelated.

- BASES: "letters" that spell out the genetic code. In DNA, the code letters are A, T, G, and C, which stand for the chemicals adenine, thymine, guanine, and cytosine, respectively. In RNA, thymine is replaced by uracil.

- BASE PAIR: two bases which form a "rung of the DNA ladder". In base pairing, adenine always pairs with thymine, and guanine always

pairs with cytosine.

- BINDING SITE: a short DNA or RNA sequence to which a molecule specifically binds. In DNA, the molecule is often a transcription factor.

- CHROMOSOME: a structure of compact intertwined molecules of DNA found in the nucleus of cells. Chromosome contain the cell's genetic information. Humans normally have 46 chromosomes.

- CIS-REGULATORY LOGIC: a set of regulatory sites in genomic sequence that collectively determine a gene's level of expression. A complete functional description of a gene's cis-regulatory logic is rare in the literature.

- COMPLEMENTARY STRANDS: the two polymeric strands of the DNA double helix, held together by hydrogen bonds between complementary A-T and C-G base pairs. Because of base pair complementarity, the sequence of one strand predicts the sequence of the other.

- COMPUTATIONAL GENOMICS: an academic discipline at the interface of computer science and molecular biology, devoted to automated analysis and annotation of large amounts of genomic sequence.

- CONSENSUS SEQUENCE: in multiple alignments and motifs, a description of aligned sequences by a single sequence whose jth position contains that base occurring most frequently in the jth column of the alignment or motif.

- CONSERVATION: the maintenance of a sequence with few or no changes over time because of evolutionary pressure, particularly selective pressure against mutations deleterious to its function. Conserved sequences in two organisms or two parts of the same organism can often be identified by their similarity.

- DNA: DeoxyriboNucleic Acid, the chemical inside the nucleus of a cell that carries the genetic instructions for making living organisms.

- ENHANCER: One of the necessary regulatory elements of a gene. An enhancer is a site on DNA to which a complex of transcription factors bind to affect the availability of the promoter to RNA polymerase. A gene may have multiple enhancers; contrast repressor.

- ENHANCER REGION: the DNA sequence roughly 200 to 1000 bases upstream of a gene's transcription start site; a common location for transcriptional enhancer and repressor elements other than the core promoter.

- EUKARYOTIC: an organism that uses a membrane to enclose its nucleus and organelles in its cells. e.g. plant and animal; contrast prokaryotic.

- EXPRESSED: of a gene, transcribed (and possibly translated) into its active end product.

- EXON: The region of a gene that contains the code for producing the gene's protein. Each exon codes for a specific portion of the complete protein. In some species (including humans), a gene's exons are separated by long regions of DNA (called introns or sometimes "junk DNA") that have no apparent function; contrast intron.

- GAP: in an alignment, a run of one or more columns in which bases of one sequence are not aligned to any base of another.

- GENE: the functional unit of the genome; the functional and physical unit of heredity passed from parent to offspring. Genes are pieces of DNA, and most genes contain the information for making a specific protein.

- GENE EXPRESSION: The process by which proteins are made from the instructions encoded in DNA.
- GENOME: the complete collection of an organism's DNA; all the sequence that must be copied when a cell replicates.
- HOMOLOGOUS: having the same evolutionary origin but serving different functions. See orthologous and paralogous.
- INDEL: an mutation in which a base is inserted or deleted from a sequence.
- INTRON: A noncoding sequence of DNA that is initially copied into RNA but is cut out of the final RNA transcript; contrast exon.
- LOCAL ALIGNMENT: alignment in which only a substring of each sequence, rather than the entire sequence, participates in the alignment.
- LOCUS CONTROL REGION: a regulatory region, not in the promoter or enhancer region of a particular gene, that may control the expression of several genes up to tens of kilobases away.
- MOTIF: a pattern appearing (perhaps with small differences) in multiple sequences. Motifs in genomic sequence often derive from conserved transcription factor binding sites.
- MOTIF FINDING: the process of locating the occurrences of a hidden motif in one or more genomic sequences.
- MUTATION: a change in a sequence, usually caused on a small scale by insertion, deletion, or replacement of a base. In most cases, DNA changes either have no effect or cause harm, but occasionally a mutation can improve an organism's chance of surviving and passing the beneficial change on to its descendants.

- NUCLEOTIDE: often used synonymously with base; A nucleotide consists of a base plus a molecule of sugar and one of phosphoric acid.
- ORTHOLOGOUS: of sequences in two or more organisms, deriving from the same sequence in the organisms' evolutionary common ancestor. Contrast paralogous.
- PARALOGOUS: of sequences in a single organism, deriving from a single ancestral sequence by duplication, as with families of duplicated genes. Contrast orthologous.
- PROKARYOTIC: of cells, lacking a nuclear envelope. Bacteria and archaea are prokaryotic cells; contrast eukaryotic.
- PROMOTER: the part of a gene that contains the information to turn the gene on or off. The process of transcription is initiated at the promoter.
- PROMOTER REGION: the sequence between 0 and about 200 bases upstream of a gene's transcription start site, containing the basal promoter and possibly one or more regulatory sites.
- PROTEIN: a long polymer of covalently linked amino acids. Proteins perform almost all enzymatic and most structural and regulatory functions in living cells; their function is determined by their three-dimensional folded shape.
- PURINE: an A or G base, denoted R in sequences.
- PYRIMIDINE: a C or T base denoted Y in sequences.
- REGULATORY MECHANISM: any means by which a cell controls the expression of its genes. Common regulatory mechanisms include control of transcription, selective degradation of RNA and protein, and RNA structures such as hairpins that affect the rate of translation.

- REGULATORY SITE: any site in genomic sequence that affects the expression of a nearby gene. Regulatory sites are typically binding sites for transcription factors or other proteins.

- REPETITIVE ELEMENT: any sequence that occurs multiple times in a genome. Types include duplications and interspersed repeats.

- REPRESSOR: a genomic sequence element that acts to decrease transcription of a gene, usually by attracting a transcription factor that directly or indirectly hinders recruitment of the polymerase complex; contrast enhancer.

- RNA: RiboNucleic Acid, a chemical similar to a single strand of DNA. RNA delivers DNA's genetic message to the cytoplasm of a cell where proteins are made.

- RNA POLYMERASE: a macromolecular complex that transcribes DNA into RNA.

- SEQUENCE FEATURE: a substring of a genomic sequence with an identifiable present or past function.

- SUBSTITUTION: a mutation in which one base changes into another.

- TRANSCRIPTION: The synthesis of an RNA copy from a sequence of DNA; the first step in gene expression. Contrast translation.

- TRANSCRIPTION FACTOR: a protein that modifies the rate at which a gene is transcribed. Transcription factors often work by directly or indirectly contacting the RNA polymerase complex, but they may also work in other ways, e.g. by temporarily bending the DNA near a gene.

- TRANSITION: a substitution mutation in genomic sequence that changes one purine to another or one pyrimidine to another; contrast transversion.

- TRANSLATION: the process by which the information coded in a sequence of m-RNA is translated into a sequence of amino acids in a protein.

- TRANSVERSION: a substitution mutation in genomic sequence that changes a purine to a pyrimidine or vice versa; contrast transition.