8-2012

Alignment, Clustering and Extraction of Structured Motifs in DNA Promoter Sequences

Faisal Abdulmalek Alobaid
Syracuse University

ABSTRACT

A simple motif is a short DNA sequence found in the promoter region and believed to act as a binding site for a transcription factor protein. A structured motif is a sequence of simple motifs (boxes) separated by short sequences (gaps). Biologists theorize that the presence of these motifs play a key role in gene expression regulation. Discovering these patterns is an important step towards understanding protein-gene and gene-gene interaction thus facilitates the building of accurate gene regulatory network models. DNA sequence motif extraction is an important problem in bioinformatics. Many studies have proposed algorithms to solve the problem instance of *simple motif* extraction. Only in the past decade has the more complex *structured motif extraction* problem been examined by researchers.

The problem is inherently challenging as structured motif patterns are segmented into several boxes separated by variable size gaps for each instance. These boxes may not be exact copies, but may have multiple mismatched positions. The challenge is extenuated by the lack of resources for real datasets covering a wide range of possible cases. Also, incomplete annotation of real data leads to the discovery of unknown motifs that may be regarded as false positives. Furthermore, current algorithms demand unreasonable amount of prior knowledge to successfully extract the target pattern.

The contributions of this research are four new algorithms. First, *SMGenerate* generates simulated datasets of implanted motifs that covers a wide range of biologically possible cases. Second, *SMAlign* aligns a pair of structured motifs optimally and efficiently given their gap constraints. Third, *SMCluster* produces multiple alignment of structured motifs through hierarchical clustering using SMAlign's affinity score. Finally, *SMExtract* extracts structured motifs from a set of sequences by using SMCluster to construct the target pattern from the top reported two-box patterns (fragments), extracted using an existing algorithm (Exmotif) and a two-box template. The main advantage of SMExtract is its efficiency to extract longer degenerate patterns while requiring less prior knowledge, about the pattern to be extracted, than current algorithms.

ALIGNMENT, CLUSTERING AND EXTRACTION OF STRUCTURED MOTIFS IN DNA PROMOTER SEQUENCES

By

Faisal Abdulmalek Alobaid
M.S. in CS, University of Southern California, 2005
B.S. in CE, University of Massachusetts, Dartmouth, 2000

Dissertation
Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Syracuse University
August 2012

Advised by:
Prof. Kishan Mehrotra
Prof. Chilukuri Mohan
Prof. Ramesh Raina

ACKNOWLEDGMENTS

The journey towards earning my PhD degree was difficult but very rewarding experience, and it was not possible without, first and foremost, God's blessings and then my family's support and encouragement. This work is especially dedicated to my beloved parents whose unconditional love, sacrifices, and prayers have always been my driving force in pursuing higher achievements, and making them proud of their son's accomplishments. Since my early age, they have played a major role in supporting my interest in science and technology, especially computer related technology. Also, this achievement was made possible by my beloved wife's continuing love, support, and devoted patience in raising our son while I was away, which she did an admirable job. My siblings have always been caring and supportive through out my journey as well.

Great amount of gratitude goes to my respected advisors. Prof. Mehrotra always provided great guidance and encouragement to challenge my ideas to gain deeper understanding and achieve greater results. Prof. Mohan's keen observations and advise were valuable in enhancing the research's outcome. Prof. Raina's valuable role in easing the understanding of the biological problem and his valuable feedback have enhanced the research's results and made it relevant to the ultimate beneficiary, the biologists community. I will always be very grateful for their mentoring, support, and guidance.

I'm also grateful to my defense committee chair, Prof. Borer, and members, Prof. J. Oh and Prof. Fardad, for taking the time out of their busy schedule to give their valuable feedback, further enhancing the final version of this dissertation.

Finally, many thanks go to my scholarship sponsor PAAET, my PAAET advisor Ms. Sahakyan for her continuing help and support, and my friends in the US and back home, in Kuwait.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background

The field of bioinformatics aspires to solve challenging problems in molecular biology by applying algorithms from the field of computational intelligence. In recent decades, the focus of the field was on structural genomics. Scientists and researchers were racing to sequence as much DNA of living beings (bacteria, plants, animals and human) as possible where they tried to define the structural components of DNA by identifying the genes' subsequences within. These tremendously successful efforts helped to advance the attention of the field from structural genomics to functional genomics. The goal has shifted to not only study the components of DNA, but also to understand the functionality of these components (genes) as collective entities rather than as individuals to ultimately produce gene regulatory networks as models for genome wide behavior [1].

Conceiving gene regulation models allows us to better diagnose and treat diseases, and use genomic engineering to inject one species of plants with capabilities from other plants, such as drought resistance. Thus, in order to step closer to achieving that goal, understanding gene expression regulation is crucial. Biologists theorize that the presence of short DNA sequences called motifs (cis-regulatory elements) in the non-coding (intergenic or promoter) region of DNA plays a key role in gene expression regulation [2]. Motifs play a crucial role in regulating nearby

(downstream) genes' expression levels by acting as binding sites for proteins called transcription factors, which control genes' transcription, hence their expression levels (Figure 1.1).

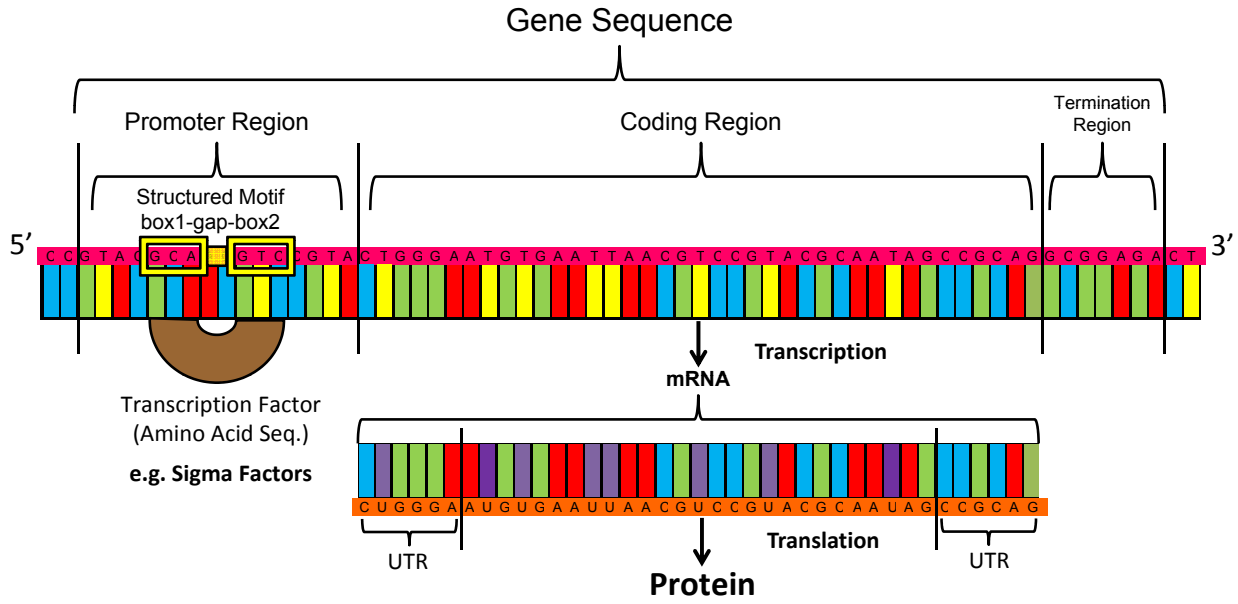


Fig. 1.1: A basic view of the gene expression process. A transcription factor protein (e.g., a sigma factor) binds to a site in the promoter region (showing a two-box site, i.e., a structured motif) that affects the transcription level of the down stream gene, which in turn affects its expression level (protein production).

Solving motif discovery problem in DNA sequences is an important step towards constructing gene regulatory networks. Many algorithms have been proposed, utilizing a variety of problem formulations and solution approaches [3, 4, 5]. Most of them were focused on the simple motif extraction problem, where motifs are considered as short single subsequences of DNA. However, many important cell mechanisms rely on proteins that bind to structured motifs. A structured motif is a set of short DNA subsequences (boxes) separated by gaps that have variable, yet bounded, lengths. For example, *sigma factors* are transcription factor proteins, in the bacteria *Streptomyces coelicolor* [6] and *Bacillus subtilis* [7], that bind to two-box structured motifs.

1.2 Problem Definition

In this section, we formally define the problem of structured motif extraction based on the biological hypothesis: *if a set of genes are co-regulated (co-expressed) together under certain conditions,*

then there exist functionally significant overrepresented DNA patterns in their promoter regions [2, 8]. Co-regulated genes are a set of genes that have their expression level changed, from normal, under the same condition. Their promoter regions are hypothesized to contain DNA patterns that are present more than statistical chance (e.g., present in 70% of the given promoter regions). Hence, the goal is to find an overrepresented DNA pattern (structured motif) in a given set of sequences $\mathcal{S} = \{S_i\}_{i=1}^{t}$. Each sequence $S_i = s_{i1}s_{i2}\cdots s_{in_i}$ is of length n_i, and its elements $s_{ij} \in \Sigma_{\text{DNA}} = \{A, C, G, T\}$. Suppose $\Sigma_{\text{IUPAC}} = \{A, C, G, T, R, Y, K, M, S, W, B, D, H, V, N\}$ is the extended DNA alphabet, where the degenerate symbols are as shown in Table 1.1. Then the

Table 1.1: IUPAC Extended DNA Alphabet

Symbol	Description	Bases
A	Adenine	A
C	Cytosine	C
G	Guanine	G
T	Thymine	T
R	puRine	A or G
Y	pYrimidine	C or T
K	Keto	G or T
M	aMino	A or C
S	Strong	C or G
W	Weak	A or T
B	not A	C or G or T
D	not C	A or G or T
H	not G	A or C or T
V	not T	A or C or G
N	aNy base	A or C or G or T

consensus sequence of a structured motif M with k boxes and gap ranges can be expressed as:

$$M = m_1 G_1 m_2 G_2 m_3 \cdots m_{k-1} G_{k-1} m_k,$$

where $m_i \in (\Sigma_{\text{IUPAC}})^{|m_i|}$, and $G_i = [g_i^{\min}, g_i^{\max}]$ is an integer range that represents the possible lengths for the gap between m_i and m_{i+1}. The minimum length of M is defined as:

$$|M|_{\min} = |m_k| + \sum_{i=1}^{k-1} (|m_i| + g_i^{\min}). \tag{1.1}$$

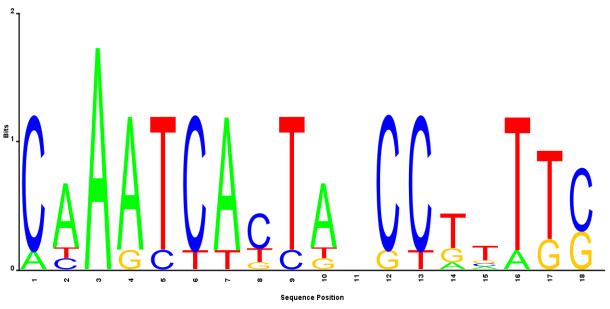
$|M|_{\max}$ is defined similarly, replacing $g_i^{\min}$ with $g_i^{\max}$. An instance of M (a transcription factor binding site) is expressed as:

$$M^x = m_1^x g_1^x m_2^x \cdots m_{(k-1)}^x g_{(k-1)}^x m_k^x,$$

where $x \in \mathbb{Z}^+$ is the instance index, $g_i^x \in G_i$ is the actual gap size between m_i^x and m_{i+1}^x, and $m_i^x \in \Sigma_{\text{DNA}}$ denotes the i^{th} box in instance M^x. M^x and M^y are valid instances of M provided the Hamming distance between boxes m_i^x and m_i^y is $\leq 2d_i^M$, where d_i^M is the number of allowed mismatches in box m_i of M. These mismatches can be position specific or arbitrarily positioned among instances (details in Section 3.2.2). We also define a *quorum* parameter $0 \leq q_M \leq 1$ as the fraction of sequences in $\mathcal{S}$ that have at least one instance of M. Finally, non-overlapping repeats are allowed, and the possibilities for the instances to be present in the coding strands, the complementary strands or both are also allowed. Figure 1.2 shows a sample set of five DNA sequences containing an overrepresented pattern of a two-box structured motif.

```
>seq1 size:70bp
aattcagttttaacCAAATCTCTActaaaccattctCCTATGCggccggct
aatgaaattggacgttctc
>seq2 size:70bp
ttcaaacagctatatgcccCAAACCACTAatataggtcaattctGCATTTC
tgattccagatttttcct
>seq3 size:70bp
atGATGAGGcgttAAGTGATTGGttcttattaCTAATCATTAtagagttCC
GTTTGaaaaaaaaattagc
>seq4 size:70bp
taaAAAATCAGTAttCCGTTTCcCAAAAAGgcgggtcacttcaatCAGTGA
TTTGtaagacactataggg
>seq5 size:70bp
tttGCACAGGtggactaacTGGTGACTTGcCAAAAGGattggcgtatacct
cTAATAATTTGgttttttt
```

(a) Showing a two-box structured motif instances implanted on both strands with repeats.



(b) The structured motif logo. The 11^{th} position is a variable gap of size $\in [2, 15]$.

Fig. 1.2: A simulated set of five DNA sequence promoter regions of co-regulated genes having instances of a two-box structured motif.

Example Problem: Consider the following conditions in a simulated problem. Suppose $\mathcal{S}$ is a set of $t = 100$ randomly generated sequences. Each sequence is generated from a uniform distribution over Σ_{DNA}, each of size $n = 1000$bp (base pairs). In this set of sequences, we implanted 136 instances of the 3-box structured motif:

$$M = \texttt{ACGACGGCCT[4,18]TGCTCCTAGTGG[4,18]AAAGAGAGCCGGTA}$$

such that each sequence has one instance of M and the remaining 36 instances are distributed randomly among the sequences. Also, 62 of the 136 instances are on the complementary strand, and M has one arbitrarily positioned mismatch per box ($d_i^M = 1, \forall i \in [1, 3]$). The goal is to extract M from the given set $\mathcal{S}$ using limited or no *a priori* information about M.

The structured motif extraction problem is inherently challenging for the following reasons:

1. Structured motif patterns are segmented into several simple motifs (boxes) separated by gaps of variable sizes in each instance.
2. These boxes are non-exact copies having a number of mismatches (position specific or arbitrary) among the instances.
3. Some instances may be found in the coding strand and some on the complementary strand.
4. Multiple (repeated) non-overlapping instances may exist in the same promoter sequence.
5. Instances may not exist in all given promoter sequences.
6. Multiple distinct overrepresented structured motif patterns may exist in the same set of sequences.

In addition, researchers are hindered in their effort to discover efficient and testable algorithms due to the following challenges:

1. Lack of resources for real datasets covering a wide range of biologically possible cases.
2. Incomplete annotation of real datasets can lead to considering the discovery of previously unknown patterns as false positives.

The focus of this dissertation is to provide a novel algorithm to solve effectively this challenging problem, as outlined in the next section.

1.3 Dissertation Outline

In this dissertation, we address the challenges of the structured motif extraction problem posed in the previous section. First, in Chapter 2, we briefly review the state of the art for the simple motif extraction problem and then survey six structured motif extraction algorithms, focusing on their novel ideas, strengths and weaknesses. Next, in Chapter 3, we address the problem of providing a variety of controlled datasets by introducing *SMGenerate*, a comprehensive, flexible, and biologically relevant, synthetic dataset generator. Then, in Chapter 4, we introduce *SMAlign*, an efficient and optimal structured motif pairwise alignment algorithm that is at the core of this dissertation. In Chapter 5, we introduce *SMCluster*, a progressive structured motif multiple alignment algorithm based on hierarchical clustering, that uses the pairwise *affinity* score produced by SMAlign. In Chapter 6, we propose *SMExtract*, a novel approach for solving the structured motif extraction problem by fragment assembly. That is, we use an existing algorithm (Exmotif [9]) to extract and rank two-box patterns (fragments), from the input sequences, and then use SMCluster to construct the target unknown pattern by assembling together the top found fragments. Compared to current methods, we will show that our method is efficient, especially for longer degenerate patterns, and flexible in finding a wide range of patterns having various characteristics, while requiring little to no prior knowledge about the target pattern. Concluding remarks and future research directions are discussed in Chapter 7. All algorithms introduced in this dissertation can be accessed at *http://bioproject.syr.edu/smtools*.

CHAPTER 2
RELATED WORK

In this chapter, we review the current state of the field starting with a brief review of simple motif extraction, followed by detail discussion of six structured motif extraction algorithms, namely, Exmotif [9], Structured Motif Finder (SMF) [10], BioProspector [11], a Multi-Objective Genetic algorithm based extractor [12], Reversible Jump MCMC based on Hidden semi-Markov Model (HSMM) [13] and Hybrid 2^{nd} order Hidden Markov Model (HMM2) with a combinatorial method [14]. The first two are exact algorithms while the rest are stochastic.

2.1 State of the Field

It is useful to begin by examining the state of the simple motif problem, where much more research has been conducted and the findings help us understand and carry out some concepts to the structured motif extraction problem. The simple motif extraction problem has been studied extensively in the past decade. The volume and variety of approaches warranted at least four survey papers in recent years [3, 5, 8, 15].

Despite the number of papers on this topic, only Tompa *et al.* [8] have provided a comprehensive quantitative analysis of available algorithms at the time. They assessed the performance of 13 algorithms on extracting simple motifs from a variety of datasets. In their work, they have attempted

to present a collection of standard datasets and quantitative performance measures for objective comparison. Their performance measures focused on the quality of the results without considering the time or space complexity of the examined algorithms. Valuable insights were presented in their work, such as the complementary behavior of many algorithms tested, where in one algorithm provides a set of motifs that complement another. Nonetheless, they had faced many difficulties that prevented true objective assessment to produce clear results. The lack of absolute standard by which to measure correctness of the algorithms' results, and the lack of complete understanding of the regulatory mechanism, led them to conclude that their chosen datasets were a poor approximations of the biological truth. Thus, they have provided a concise list of suggestions for future studies that assess the performance of motif extraction algorithms:

1. Eliminate real datasets due to their incompleteness.
2. Eliminate the negative control datasets that contain no implanted motifs.
3. Require each tested algorithm to extract a fixed number of motifs (i.e., three motifs).
4. For each algorithm and dataset, choose the motifs with the greatest *nCC* (nucleotide correlation coefficient) score to represent that algorithm.

Li and Tompa [16] later provided a followup study to analyze further their earlier results. Specifically, they looked at the objective functions that the examined algorithms attempted to optimize, to assess their competence to solve the problem. Further, they reexamined the datasets they used, looking for some features that explain the level of difficulty facing current algorithms. Lastly, they have proposed an objective function based on the conservation of the motif's position in the promoter sequence, which no algorithm has taken into account. Other survey papers focused on qualitative description and categorization of examined algorithms.

Wei and Yu [5] proposed a categorization mainly based on algorithmic type, i.e., combinatorial vs. probabilistic. They have also placed some algorithms in a category of phylogenetic footprinting approach, which is a qualitatively different issue. Nevertheless, they have provided detailed qualitative descriptions of algorithms in each category, with an abundance of web resources for finding tools and datasets.

Das and Dai [3] provided categorization of the literature based on biological data employed for motif extraction. Three main categories were presented. First, algorithms that use promoter sequences from co-regulated genes from a single genome; second, those that use orthologous promoter sequences of a single gene; third, those that employ both data types. One interesting approach discussed was by Kaplan *et al.* [17], which combined DNA sequence data with transcription factor protein structure information, to infer context-specific amino acid binding site recognition preference (that applies to the same protein structural family).

Another point of view was provided by Sandve and Drablos [15]. They have argued that the availability of many distinct varieties of algorithmic approaches contributed to the difficulty of obtaining a good understanding of the field. They proposed an integrated framework that integrates all relevant elements for describing motif extraction problem formulation, be it simple or structured. A qualitative discussion of current algorithms was provided, based on their proposed integrated framework with four levels of abstraction. In Level 1, the framework describes motifs as single short contiguous segments of DNA in non-coding regions. Level 2 represents a module (structured motif) of simple motifs in proximity to each other, with flexible inter-distance constraints. Level 3 represents multiple interacting structured motifs that regulate a single gene. Level 4 represents multiple sets of structured motifs regulating multiple genes in the genome. Details for each level were presented, and it was concluded that no single algorithm takes into account all relevant elements of the proposed framework, and these algorithms treat those elements separately.

There are some common conclusions from the studies discussed above. First, quantitative analysis of available algorithms is difficult for variety of reasons including the lack of clean standardized datasets, the lack of universal problem formulations and the need for more concise performance and quality measures. Second, structured motifs are considered as collections of simple motifs with inter-motif distance (gap) constraints. However, at least a few studies [9, 12] speculate that applying single motif algorithms to discover each component separately may leave some structured motifs undiscovered, though no proof was given. Third, available algorithms perform well in relatively low complexity DNA, such as bacteria (prokaryotes). On the other hand,

as DNA complexity increases, as in human DNA (eukaryotes), current algorithms perform poorly.

Most of the above discussion addressed simple motif extraction, with some consideration of structured motifs in [15]. However, the concepts discussed above are of great importance and relevance to the structured motif problem, which we will examine next.

2.2 Exact Structured Motif Extraction

In this section we present two exact structured motif extraction algorithms, Exmotif [9] and Structured Motif Finder (SMF) [10]. The first relies on specifying a template of the target structured motif to be extracted. The second constructs the target structured motif from a list of simple motifs found using some existing simple motif algorithm.

2.2.1 Exmotif

Zhang and Zaki [9] proposed an exact enumeration type algorithm to solve the structured motif extraction problem. Given a set of sequences, Exmotif extracts the common overrepresented patterns that conform to a user defined template. The user has to define up to $5k - 1$ template parameters, where k is the number of boxes, each requiring minimum, maximum sizes and number of allowed mismatches d_i; $(k - 1)$ number of gaps, each requiring minimum and maximum sizes; finally, the user must specify the quorum q (the least percentage of sequences that contain a pattern).

Exmotif solves two different extraction problems. The first is finding all common structured motifs in a set of sequences such that the support of each is at least q (the focus of this dissertation). The second is finding all repeated structured motifs in a single sequence such that the weighted support level of each pattern is at least q. The algorithm takes a set of sequences $\mathcal{S}$ as input, each having an arbitrary length that represent promoters of co-expressed genes, orthologous promoters, or other types of DNA data (no prior assumption). Also, it takes a user defined template of the form $\mathcal{T} = [b_1^{\min}, b_1^{\max}] - [g_1^{\min}, g_1^{\max}] - \ldots - [b_k^{\min}, b_k^{\max}]$ where the ranges define the upper and lower bounds for each box length as well as each gap length, the quorum q and the number of allowed

mismatches per box d_i. Depending on which of the two problems (mentioned earlier) the user choose to solve, Exmotif outputs the list of found structured motifs that matches the template and have support (or weighted support) of at least q.

The key concept behind Exmotif is the use of an inverted index of symbol positions. For each symbol in an input sequence, Exmotif constructs a position list indicating where the symbol occurs in each input sequence. Then, using the template for guidance, it combines these lists incrementally to build position lists for power-of-2 length strings, until it reaches the maximum permitted box length. These new position lists are then combined further to construct all frequent structured motifs present in the sequences. The algorithm avoids redundant processing as it constructs only structured motifs that have the potential to exceed the quorum, and prunes other possible constructions on the fly. However, this pruning is most effective when no mismatches are specified. Also, Exmotif only stores minimal information in each position list, and reuses intermediate results, to save space and time. The time complexity of the algorithm is $O(kN|\Sigma_{\text{DNA}}|^{kb})$, where k is the total number of boxes in the template, N is the total length of all input sequences, $|\Sigma_{\text{DNA}}|$ is the number of symbols in the DNA alphabet and b is the maximum box length in the template.

The authors tested Exmotif against the latest version, at the time, of an algorithm called RISOTTO [18] (earlier version called RISO). In the first experiment, they used 1,062 non-coding sequences of *B. subtilis* [7] genome having a total of 196,736 nucleotides. They tested the two algorithms for exact matching (without mismatches), as well as approximate matching (allowing bounded mismatches). Also, for each case, they varied the gap ranges, the number of template boxes, and the quorum. Their results compare the average running time of hundreds of generated templates. This study showed that Exmotif dominated the performance of RISOTTO in all cases except one case. RISOTTO outperformed Exmotif only when the latter dealt with high quorum and high mismatch bound (allowing more mismatch tolerance). The authors also tested Exmotif's performance to extract actual (Zinc factors) structured motifs from yeast genome against known motifs. They created several templates, based on their knowledge of the known motifs, and tested them on the upstream region of 68 genes. Then the resulting structured motifs were ranked according

to a z-score to pick the top ranked results. Their algorithm was able to predict all eight structured motifs in contrast with RISOTTO, which was only able to predict three of the eight.

Based on this study, it is reasonable to conclude that Exmotif is an exact and efficient (only for patterns with little to no mismatches) algorithm to solve the structured motif extraction problem. However, despite its advantages, the major drawbacks in effectively using Exmotif are the exponential drop in efficiency for patterns with many number of mismatches and the difficulty in specifying the $5k - 1$ template parameters for an unknown pattern. This leads to a high dimensional search space for all possible templates that makes it difficult to formulate a suitable template with minimal prior knowledge about an unknown pattern. Another drawback is that Exmotif's complexity grows exponentially with k, b, and $\sum d_i$.

2.2.2 Structured Motif Finder (SMF)

The *Structured Motif Finder* (SMF) proposed by Federico *et al.* [10] took a different modular approach to the problem. The authors formulated the structured motif extraction problem in a manner similar to that described in Section 1.2, and their algorithm solves the problem utilizing two phases. First, it uses a black box approach using an existing single motif finding algorithm to find all occurrences of simple motifs in the input sequences. Second, they store each found motif together with their list of occurrences in an associative array called a *simple motif list*. Then they incrementally build a list of structured motifs having i boxes ($1 \leq i \leq k$ where k is the maximum number of boxes) by combining the list of structured motifs that contain $i - 1$ boxes (the previous iteration) with the *simple motif list*. Rejection of infrequent structured motifs is done on the fly, pruning paths with less frequent simple motifs based on the quorum. This is motivated by the fact that the components of a frequent structured motif are themselves frequent. This process is repeated until the final structured motifs, with k boxes, are available in the list of lists called *structured motif lists*.

This study focused on solving the problem of extracting an implanted structured motif in a set of randomly generated sequences. They implanted simple motifs having lengths l and max

mismatches d to represent the components of an implanted structured motif. The user inputs the max number of boxes k, (l, d) for each box, and the maximum quorum q. Then the algorithm outputs all occurrences of structured motifs having from two to k number of boxes. For their simple motif extraction algorithm, they selected SMILE [19] because it has the property of being an exact algorithm that finds all occurrences of simple motifs. Based on this choice, the time complexity of SMF is $O(\sum_{(l,d)} nt^2 v(l,d) + oM_1 \sum_{i=1}^{k-1} oM_i)$ where n is the maximum sequence length in the input set, t is the total number of input sequences, (l, d) pair represents the length and max mismatches for each box, $v(l, d)$ is the maximum number of l-*mers* at Hamming distance at most d from another l-mer (total number of neighbors), k is the total number of boxes specified by the user, and oM_i is the total number of structured motifs having i boxes.

SMF was tested against RISOTTO [18] as well. The test was conducted on synthetically generated sequences to solve the implanted structured motif problem. The dataset consisted of $t = 20$ randomly generated sequences each having a length of $n = 600$bp. They conducted four experiments with (l, d) pairs as $(9, 2), (10, 2), (11, 3)$ and $(12, 3)$ varying the number of boxes in the range $k \in [2, 10]$. They recorded the average running time of each experiment performed 30 times. In all experiments, SMF dominated RISOTTO's performance in finding the same structured motifs.

Federico *et al.* [10] claim that a better algorithm used for simple motif extraction (phase one) will only further improve the results. However, this approach suffers from the same problems as the previous algorithm, in requiring the user to specify the sizes of the simple motifs and the number of allowed mismatches, which increases the difficulty of using the algorithm when no such information is available. Further, this approach does not address the problem when there are repeated occurrences of structured motifs or motifs implanted on both strands.

2.3 Stochastic Structured Motif Extraction

In this section, we present four stochastic algorithms. The first is based on Gibbs sampling, the second is based on an evolutionary algorithm approach and the remaining are based on Hidden Markov Models.

2.3.1 BioProspector

Proposed by Liu *et al.* [11], BioProspector uses a variation of the Gibbs sampling approach to solve the structured motif extraction problem. It assumes a Markovian background model of up to 3^{rd} order, and has the ability to account for multiple motif occurrences per sequence by introducing a *threshold sampler* method. The algorithm starts by initializing a motif probability matrix, of a user defined size (box size), by randomly picking motif sites in the input sequences. Then the motif matrix is sampled iteratively to pick new sites proportional to the probability $A = Q/P$, where Q is the probability that the new site is generated by the motif matrix and P is the probability that the new site is generated by the Markovian background model. Then the motif matrix is updated using all of the new sampled sites that have $A \geq T_H$ (where T_H is the high threshold) and a subset of the new sites that has $T_L \leq A \leq T_H$ picked in proportion to $A - T_L$ (where T_L is the low threshold that is fixed at 0 for the first 10 iterations and then linearly increased to $T_H/8$). This iterative process is repeated until convergence. After that, BioProspector scores the motif matrix based on a weighted Kullback-Leibler divergence measure against the background model. BioProspector reports motifs that have scores 5 standard deviations (default) above the null distribution mean of scores of the same size motifs found in sequences generated based on the background model. For a two-box structured motif, two matrices are simultaneously used, with a user defined gap range.

As with earlier algorithms, BioProspector also takes as input a set of sequences, exact box sizes (not ranges), and a gap range, for a two-box structured motif. However, BioProspector neither uses a quorum parameter nor per-box mismatches, as they are inherently handled by the algorithm. The major drawback of BioProspector is its limitation of only finding dyads or palindromes of

user defined sizes. It requires the user to specify the exact sizes of the two-boxes (each box size should be in the range of 4bp to 50bp) and a range for the gap in-between, where (maximum gap - minimum gap) ≤ 20bp. When the pattern is unknown, the user cannot be expected to supply such precise information. Also, based on its stochastic nature, multiple runs are required to obtain a set of candidate results.

2.3.2 Multi-Objective Genetic Algorithm

Kaya [12] proposed a multi-objective optimization approach to solve the structured motif extraction problem using a genetic algorithm. The proposed approach requires some user knowledge of the parameters, e.g., the maximum number of boxes k, the lengths of these boxes (as a range), the range of the gap lengths and a *limit* $\in [0, 1]$ value used as a threshold. Also, no specific biological problem or datasets assumed during construction of the approach. This algorithm outputs what is called the dominant set of structured motifs as strings of IUPAC codes with gaps (similar to the output of algorithms discussed earlier).

Gene 1										Gene t	
$\|M_1\|$	$\|M_2\|$	$\|M_3\|$	w_1	s_1	max_{11}-min_{11}	max_{12}-min_{12}	...	w_t	s_t	max_{t1}-min_{t1}	max_{t2}-min_{t2}

Fig. 2.1: An individual representation in the genetic algorithm population [12].

The key idea in this approach is to use a genetic algorithm to simultaneously optimize three competing objective functions, namely, maximizing similarity, minimizing total gap range and maximizing support. These objectives are used to measure the fitness of an individual in the population of the genetic algorithm. Each individual in the population has the structure as in Figure 2.1. Each individual starts with k number of fields, which is the user defined max number of boxes in a structured motif, and each field contains the length of the corresponding box. Then the individual is divided into t genes, where t is the total number of input sequences, such that each gene corresponds to an input sequence. Furthermore, each gene is subdivided into a weight field $w_i \in [0, 1]$ (if $w_i <$ *limit*, then this instance of the structured motif for sequence i is not considered), first gap starting position s_i and the gap ranges (two gap ranges per gene in Figure 2.1 for the

maximum number of three boxes, $k = 3$).

For the first objective measure, the author used the following similarity measure to be maximized, *similarity*$(M) = \sum_{i=1}^{l} dv(i)/l$, where M contains the boxes of the structured motif being examined (an individual in the population), l is the total length of these boxes, and $dv(i)$ is the dominance value of the dominant nucleotide in position i that have maximum frequency (in the input sequences). The second measure is the total gap range in a structured motif, to be minimized. The third measure is the support for a structured motif, to be maximized, that is the number of sequences that contain at least one such structured motif. To solve the problem, the author used a well-known multi-objective genetic algorithm called *Non-dominated Sorting Genetic Algorithm (NSGA-II)* [20]. This algorithm provides a sorted set of non-dominated individuals (structured motifs) in the population based on optimizing the above objective measures. We say a solution x_i dominates another solution x_j if both of the following are true:

1. The solution x_i is not worse than x_j in all objectives.
2. The solution x_i is strictly better than x_j in at least one objective.

The set of all non-dominated solutions is called the *Pareto-optimal* set. The algorithm starts by initializing the population having a user defined size N and max number of generations. Pairs of individuals are selected at random from the population to form the parents set. Then a one-point crossover operator is applied to parents, based on a user defined crossover probability, and three mutation operators are also applied based on user defined mutation probability. The first mutation operator is shifting the starting location s_i of a randomly chosen gene to the right. The second is shifting the starting location s_j of another randomly chosen gene to the left. The third is adding or subtracting a small random integer to any of the lengths (for the boxes or gaps), weights w_i, or starting locations s_i (all changes are checked not to exceed specified bounds). After that, based on the Pareto-optimal relation and another measure (called the crowding distance), good solutions are chosen from the combined set of parents and offsprings. This process is repeated until the termination condition is reached. The crowding distance is used when two individuals have the same Pareto-optimal rank and the individual with the larger crowding distance is chosen, otherwise,

the better Pareto-optimal rank individual is chosen. The crowding distance ensures the choice of individuals is spread along the Pareto-optimal front which in turn ensures maximum coverage avoiding local minima.

The author tested his approach against RISOTTO [18] and Exmotif [9] using $t = 70$ randomly generated sequences each of length $n = 1000$bp implanting one structured motif in the dataset, varying the number of boxes, the gap ranges and the boxes' lengths for each experiment. The results, first, confirm the dominance of Exmotif over RISOTTO (as discussed previously), second, the author's approach dominate Exmotif's results in all experiments. A real dataset was used for another experiment using the promoters of co-regulated genes regulated by seven transcription factor (TF) proteins to extract structured motif, and the multi-objective approach produced more accurate results. For example, for TF GAL4 the structured motif `CGGRnRCYnYnCnCCG` is known, Exmotif predicted `CGG[11,11]CCG`, whereas the multi-objective approach predicted `CGG[3,4]YY[5,5]CCG`, a more accurate representation of the actual motif.

This approach shows a lot of potential as it outperforms other tested algorithms in terms of time and quality of results. Nevertheless, this algorithm is basically a stochastic search in the space of all possible structured motifs subject to user provided parameters. As a stochastic algorithm, optimal or complete solutions are not guaranteed. Furthermore, it requires multiple runs with good choices for the cross-over and mutation rates as well as choosing a large enough number of generations to reach convergence. The algorithm also requires specifying the upper bound for the usual parameters (number of boxes k, the box size range, and the gap ranges). It is shown by the author that the time complexity grows with the number of boxes as well as the size of these boxes and the size of the gaps. However, it is not clear if the growth is exponential or otherwise, as the presented plots have few data points (three or four), and no formal complexity analysis was presented. Furthermore, the choice of the objective measures to be optimized may need further exploration; especially the minimization of gap ranges. This measure leads to a bias towards extracting structured motifs with smaller gap ranges, which might not be desirable. For instance, in the case of composite motifs (structured motifs with boxes having larger gaps in-between) [15], this algorithm may rank

them much lower than structured motif patterns with smaller gaps in the same dataset. Thus, this algorithm prefers structured motifs with smaller gaps over ones with larger gaps.

2.3.3 Hidden Markov Model Based Algorithms

In this section we examine two algorithms that use some form of hidden Markov model (HMM) approach to model structured motifs. Deviating from the classical HMM (Figure 2.2a), the first algorithm's model is based on the hidden semi-Markov model (HSMM) where a predefined variable-length observation sequence is emitted per state (Figure 2.2b) rather than a single observation per state as in (HMM). In the second algorithm, a 2^{nd} order HMM is used where each state emits a fixed predefined length observation sequence (Figure 2.2c) (see [21] for more information on HMM in bioinformatics).

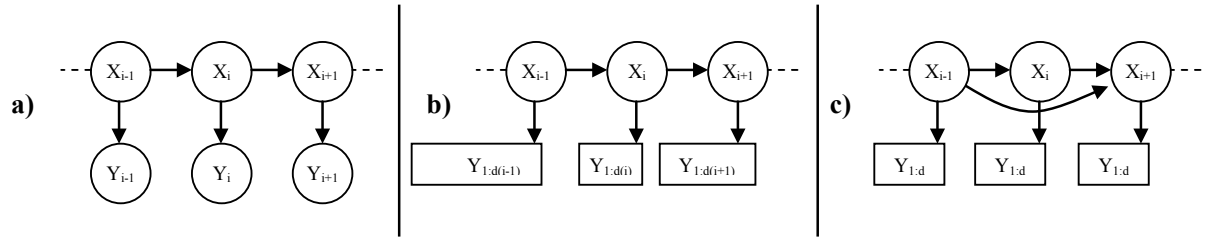


Fig. 2.2: General representation of various HMMs. (Xs) are hidden states and (Ys) are observations. a) HMM, b) HSMM with variable length observation sequence per state, c) 2^{nd} order HMM with fixed length observation sequence per state.

2.3.3.1 Reversible Jump MCMC

Nicolas *et al.* [13] proposed a stochastic HSMM based approach to extract structured motifs specifically from bacterial genome *B. subtilis* [7]. This algorithm tries to extract two-box structured motif of the type (box1-gap-box2). The biological process of binding sigma factors to a two-box structured motif is shown in Figure 2.3a, while the proposed HSMM model is shown in Figure 2.3b. There are two paths in the model, an upper path that detects structured motifs and a lower path that is taken if no structured motif is detected. The shapes in the model indicate hidden states. The gray colored shapes emit nucleotides based on a background Markovian model. The square shape has a

shifted negative binomial distribution to model its length (duration) simulating the distance from the transcription site. The rectangular shapes have fixed, but unknown lengths. The diamond shape has a multinomial distribution to model the gap's length. Finally, the circle shape is the absorbing state to simulate the end of a structured motif site.

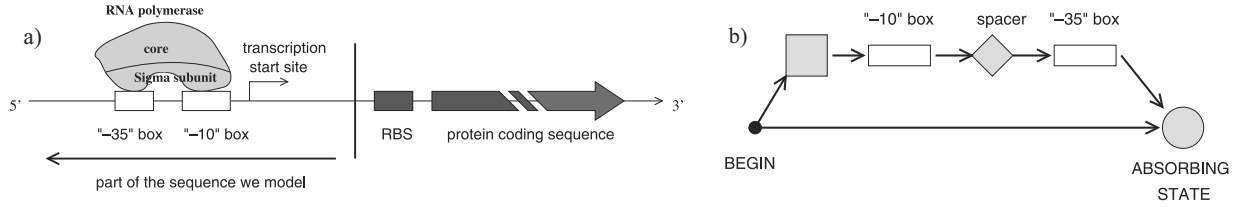


Fig. 2.3: a) Sigma factor biological process to be modeled [13]. b) HSMM model proposed by [13].

The basic idea of this approach is as follows. Suppose Y_n is a set of observed sequences (upstream sequences of genes that are regulated by the same sigma factor), and suppose $\theta = (a, b, d, r)$ are the model's parameters, where a is the transition probabilities matrix, b is the probability of emitting nucleotides from the hidden states, d is the multinomial (also, the binomial) distribution for some states (as stated earlier) and r is the order of the background Markovian model. Then, using the reversible jump MCMC, simultaneously estimate θ and predict the motif position in each sequence by sampling from the joint posterior distribution $\pi(\theta, S|Y_n)$ where S is the associated hidden path (one of the two paths in Figure 2.3b). The reversible jump part of the algorithm allows for a change in the dimensions of the model during estimation. Here the dimensions of the model are the widths of the boxes, the size of the support of the multinomial distribution for the gap's length, and the order of the background Markovian model. Here are the basic steps of the algorithm:

1. Update the state transition matrix a.
2. Update the emission probabilities b.
3. Update the parameters for the negative binomial distribution.
4. Update the parameters of the multinomial distribution.
5. Update the hidden state paths S.
6. Update the width of the box states.
7. Update the support of the multinomial distribution.

8. Update the order of the background Markovian model.

The steps 1) through 5) are Gibbs transitions leaving the model's dimension unchanged, while steps 6) through 8) are reversible jump moves as they do change the model's dimension.

The algorithm's output is a position weighted matrix (PWM) for the two-box structured motif found in the dataset. The authors did not provide any time or space complexity analysis of the approach, but they conducted three experiments on three datasets. No comparison to other algorithms was provided, only analysis of the quality of the result compared to known motifs. The authors state that the computational time grows linearly with the number of sequences. However, for an 87 sequence set with a maximum of 150 nucleotides each (Sigma B dataset), the running time of the algorithm was 70 minutes performing 5000 sweeps on the dataset. Lastly, as an example, one of the authors' result plots shows that for Sigma B dataset, the algorithm predicted 70 structured motifs in 87 sequences 10 of which are confirmed known structured motifs out of 12 total known motifs for this set. The remaining unknown predicted ones may be false positives, novel structured motifs or a mixture.

The model shows potential and has interesting properties in its ability to build an HSMM model that resembles the pattern to be extracted. On the other hand, this approach was constructed to solve specifically the two-box structured motif extraction problem. The algorithm does not scale well for longer patterns in terms of box lengths or number of boxes. Also, the approach suffers from slow performance compared to other previously discussed algorithms.

2.3.3.2 Hybrid 2^{nd} order Hidden Markov Model

Eng *et al.* [14] prospered this hybrid approach to the structured motif extraction problem by combining a stochastic 2^{nd} order HMM (HMM2) and a combinatorial method. As with the previous algorithm, their approach is focused on solving the specific two-box structured motif extraction from the genome of *S. coelicolor* [6] bacterium.

The novelty of their approach is three folds. First, they use *l-mers* strings as observations rather than single nucleotides. Second, they use expectation maximization (EM) to estimate the

parameters of the HMM2 model using the entire genome without prior knowledge of its structure, while other approaches use only promoter sequences. Third, they created an original automatic peak extraction algorithm to capture short motifs detected by the peaks of the posterior probability. The authors' motivation to use 2^{nd} order HMM (as oppose to 1^{st} order) is from the area of speech recognition where 2^{nd} order Markov chain has shown better capability in representing short speech segments. Also, using l-*mers* provides contextual information defined by neighboring observations that produced better results in areas like data mining and ecology.

For $l = 3$ and shifting by one, the 3-*mers* observations of a sequence `TAGGCTA` are `##T-#TA-TAG -AGG-GGC-GCT-CTA`, where `#` is an empty context symbol. Using the l-*mers* observations of the entire genome (model training), the authors' use EM training to estimate the parameters of the model, that is, the 2^{nd} order hidden states' transitional probabilities and the hidden states' emission probabilities. After that, they test the following posterior $P(X_t = s_i, X_{t-1} = s_i|O)$, where O is an observed l-*mer* from the entire genome. The desired outcome is when this posterior probability distribution produces peaks over some short DNA sequences over the entire genome for a specific state s_i in the model. To achieve that, the authors have designed a training strategy to successively train an ergodic (fully connected states) HMM2 with increasing number of states until the posterior probability of a state s_i appears to have far enough distance away from other states measured by Kullback-Leibler divergence measure. This state is then called the best decoding state s_i and its posterior $P(X_t = s_i, X_{t-1} = s_i|O)$ is used to find local peaks (Figure 2.4). Then, they extract the short sequences under those peaks located in the intergenic regions (called iPeaks). Next, they

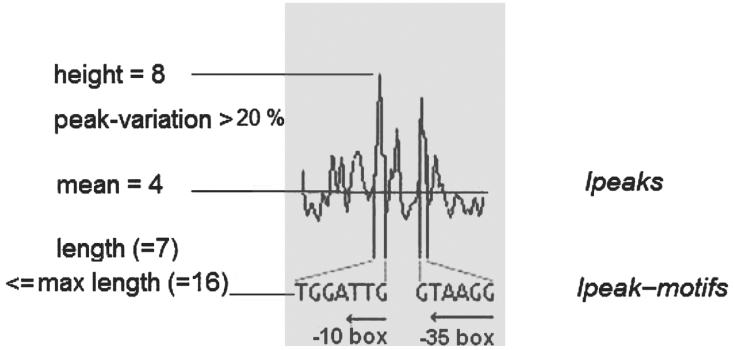


Fig. 2.4: Extracting short sequences from under iPeaks that meet specific criteria based on the input sequences [14].

use hierarchical clustering to group together the short sequences. The authors used the reciprocal of Smith and Waterman's local similarity score [22] as a distance measure to build their clusters, which produced homogeneous clusters. Then they used R'MES tool [23] to identify overrepresented subsequences (represented by the consensus sequence of their clusters) that have a score above a threshold.

Afterwards, an enumeration is performed to reconstruct the structured motif by choosing the consensus sequence of an R'MES high scoring cluster as box1. Then the cluster is expanded, finding all short sequences with high R'MES score above a predefined threshold to use as box2 choices. For each box2 candidate and for each possible gap value in a predefined range, the genome is scanned, counting the number of occurrences of the structured motif using Dsouza's scan and match algorithm [24]. Then, all found structured motif are ranked, based on their number of occurrences and box2 R'MES score. Finally, the top N among them are chosen.

For experimental results, the model was tuned using the whole genome of *S. coelicolor* [6], and the sigma factor *SigR* regulon set of structured motifs was used to validate the model. Then the tuned model was applied to find the structured motifs in the promoter region of *dagA* gene which is not regulated by *SigR*. The model found a total of 10 structured motifs, three of which were known structured motifs out of the four known for *dagA* gene (detecting only one of the two boxes for the fourth one). Other results shown by the authors also suggest good potential for the proposed algorithm.

However, the authors do not justify the choices of using local similarity for clustering and Dsouza's algorithm for match scanning while other methods may produce better results. Neither complexity analysis was presented nor testing against other known algorithms was conducted. Their focus was to analyze the quality of the result against known data to assess the performance of their model. Finally, like the previous discussed algorithm, this algorithm is specifically designed for extracting two-box structured motifs and does not scale well if the number of boxes increase.

2.4 Discussion

In this chapter, we reviewed the current state of the field for the motif extraction problem. Specifically, we looked at six structured motif extraction algorithms having distinct approaches to solve the problem. Each algorithm had its advantages and disadvantages. The reviewed algorithms fell into two categories. First, Exmotif and SMF solve the general extraction problem (not limited by the size of the structured motif), but they require an unreasonable amount of prior knowledge about the pattern to be extracted. Namely, the user has to specify the number of boxes, the sizes of these boxes, the ranges of the in-between gaps and the maximum allowed mismatches, in order to obtain a good quality result. Increasing the sizes of these parameters exponentially increases their time complexity, which makes it infeasible to search the large space of possible parameter values. Similarly, the Multi-Objective GA algorithm solves the general extraction problem, but also requires user specified upper bounds for the mentioned parameters. However, the results are sensitive to the cross-over and mutation rates as well as the number of generations to converge on a good quality result. Also, based on the algorithm's objective measures, the Multi-Objective GA algorithm is biased towards finding structured motifs with smaller gap ranges, which might not be ideal in all cases. The second category consists of BioProspector and the HMM algorithms. They are primarily designed to solve the two-box structured motif extraction problem, require multiple runs to produce candidate results, and they do not scale very well for patterns with more boxes.

In the following chapters, we address two major problems. First, as concluded by Tompa *et al.* [8], real datasets are not ideal for objective evaluation of motif extraction algorithms; in addition to the lack of available tools to generate synthetic datasets. This motivated our efforts to build *SMGenerate* (Chapter 3), a flexible and comprehensive synthetic dataset generator for the structured motif extraction problem. Second, current algorithms require unreasonable amount of user knowledge to produce good quality results. Therefore, we present a novel algorithm *SMExtract* (Chapter 6) that requires little to no knowledge about the pattern to be extracted. Chapters 4 and 5 present *SMAlign* and *SMCluster*, respectively, as essential algorithms for the successful operation of SMExtract.

CHAPTER 3
GENERATING SYNTHETIC DATASETS

3.1 Overview

Effective evaluation of motif extraction algorithms requires comprehensive controlled datasets to test the outcome in terms of quality of found motifs compared to the actual known motifs, and to evaluate the time and space complexity of such algorithms under different conditions. However, real datasets for such specific conditions are not easily available. Some efforts have been made to build standard real datasets mainly for the simple motif extraction problem [8], however, the structured motif problem lacks such comprehensive real datasets. This has motivated the development of *SMGenerate*, a flexible tool with user defined parameters to provide full range of conditions under which a specific algorithm can be evaluated specifically for the structured motif extraction problem.

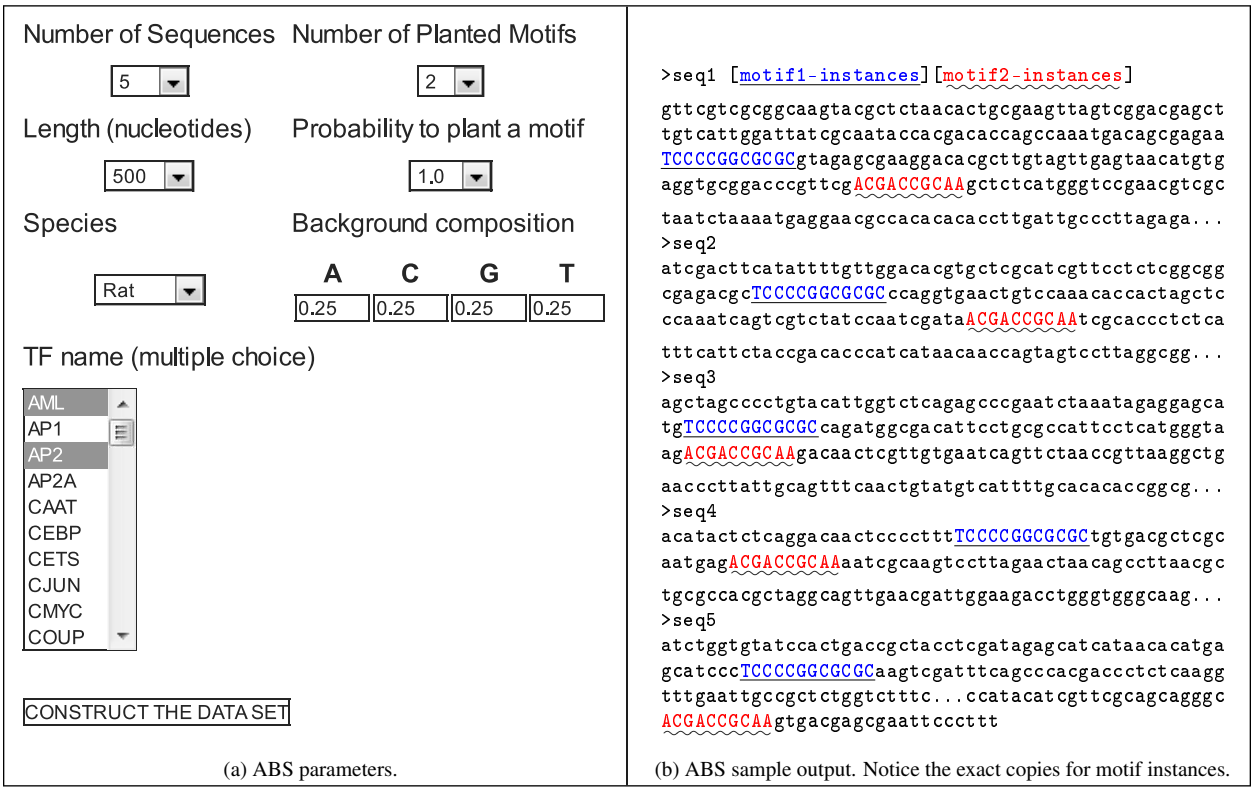
To our knowledge, *ABS Database Construction of Benchmarks (ABS)* [25] and *RSA-tools Build Control Sets (RSA)* [26] are the only publicly available tools to generate simulated datasets for the motif discovery problem. Each tool has two main tasks. Based on given user parameters, they first generate a set of background sequences, then implant simple motif instances in these sequences. This generates a single file that represents a specific test scenario. To generate a collection of such files, each representing a different test scenario, the user has to manually repeat the process. The collection of all such files is referred to as the full dataset.

ABS and RSA tools are useful for generating simple motif datasets, however, they lack many important features for generating datasets for the structured motif problem, which we will examine next. To generate the background sequences, ABS (Figure 3.1) requires the user to provide values for the following parameters:

1. Number of sequences t.
2. Length of each sequence n (all sequences have the same length).
3. Independent probabilities for the nucleotides symbols in Σ_{DNA}.

In biologically relevant background sequences, consecutive nucleotides typically satisfy Markov dependence of order $m \geq 1$. However, ABS does not provide the user with the option to specify a Markov order with a sample DNA sequence to generate the background sequences. For implanting motif instances, the user has to specify values for the following parameters:

1. Maximum number of simple motifs to implant in each sequence.



(a) ABS parameters.

(b) ABS sample output. Notice the exact copies for motif instances.

Fig. 3.1: Screenshot of ABS Database Construction of Benchmarks.

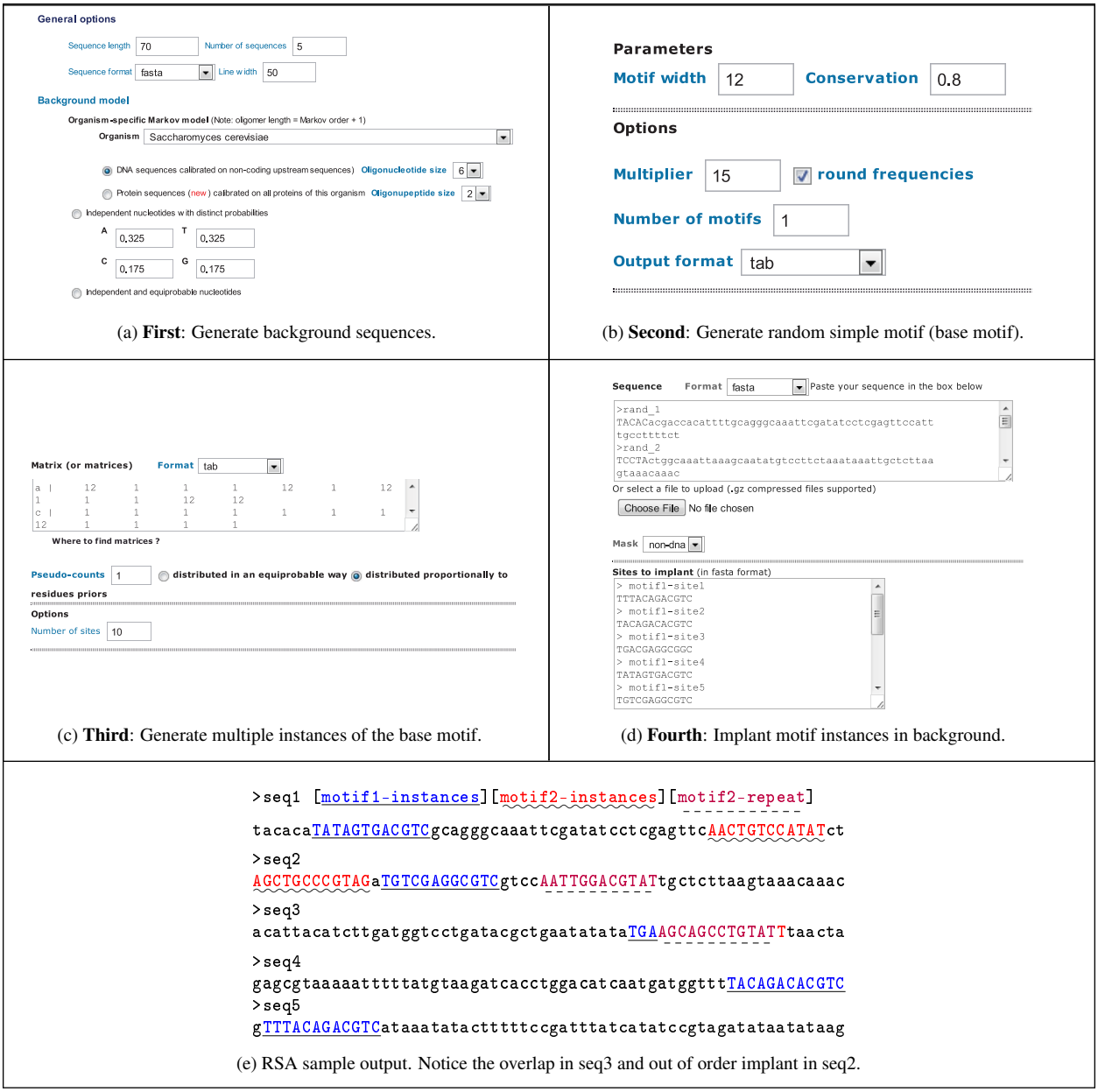
2. Probability to implant each simple motif.
3. The actual simple motifs to be implanted, chosen from a list of real motifs for different species (human, mouse and rat).

The user is restricted to the provided list of simple motifs to implant and no option is provided for a user supplied simple or structured motifs. Furthermore, there are no options to specify the probability of motif repeats in a single sequence or the probability to implant on the complementary strand. ABS implants exact copies of the specified motifs; no option is provided to specify possible mismatches (Figure 3.1b). Finally, ABS is not equipped to generate or implant structured motifs.

RSA (Figure 3.2) uses a four-step process to implant simple motifs into generated background sequences. First, RSA generates background sequences by asking the user to provide values for the following parameters (Figure 3.2a):

1. Number of sequences t.
2. Length of each sequence n (all sequences have the same length).
3. Either of the following options:
   (a) Independent probabilities for the nucleotide symbols in Σ_{DNA}.
   (b) Markov order m with a sample DNA from a user selected species.

Second, the user generates one or more simple motifs by specifying the length of the motif and the degree to which its positions are degenerate (Figure 3.2b). Third, the user generates multiple instances of the motif to be implanted (Figure 3.2c). Finally, the user implants the generated instances in the background sequences at randomly selected sites (Figure 3.2d). Although the background generation portion of RSA has considerable customization, it is severely lacking in the motif implanting part. For example, RSA is not equipped to generate or implant structured motifs with gap constraints; the user has no control over the probabilities for implanting a motif, repeating within a sequence, or implanting on the complementary strands, which are necessary for generating biologically relevant test cases. An examination of RSA's output (Figure 3.2e) shows that it implants simple motif instances in a totally random manner (in terms of their locations within



(a) **First**: Generate background sequences.

(b) **Second**: Generate random simple motif (base motif).

(c) **Third**: Generate multiple instances of the base motif.

(d) **Fourth**: Implant motif instances in background.

```
>seq1 [motif1-instances][motif2-instances][motif2-repeat]
tacacaTATAGTGACGTCgcagggcaaattcgatatcctcgagttcAACTGTCCATATct
>seq2
AGCTGCCCGTAGaTGTCGAGGCGTCgtccAATTGGACGTATtgctcttaagtaaacaaac
>seq3
acattacatcttgatggtcctgatacgctgaatatataTGAAGCAGCCTGTATTtaacta
>seq4
gagcgtaaaaatttttatgtaagatcacctggacatcaatgatggtttTACAGACACGTC
>seq5
gTTTACAGACGTCataaatatacttttttccgatttatcatatccgtagatataatataag
```

(e) RSA sample output. Notice the overlap in seq3 and out of order implant in seq2.

Fig. 3.2: Screenshot of RSA-tools Build Control Sets.

a sequence and among the sequences). This makes it difficult for the user to generate specific test cases. In addition, RSA may implant motif instances on top of each other (overlapping sites) that may render the generated data undesirable.

Both ABS and RSA procedures generate only a single file, which consists of the background sequences with implanted simple motifs. This file represents a single test case. Generating multiple test cases having different characteristics for the background sequences and/or the implanted motifs

(size, quantity, position, mismatches, etc...) requires a laborious manual repetition of the above procedures, especially when using RSA. This lack of flexibility to easily generate multiple test cases, the lack of user control of important parameters and the lack of support for generating and implanting structured motifs motivated the development of *SMGenerate*, a data generation tool that we introduce in the next section.

3.2 SMGenerate

This tool generates the background sequences, implants structured motifs, and generates full datasets for evaluating the results quality of structured motif extraction algorithms. The components of SMGenerate are discussed in detail in the following subsections.

3.2.1 Generating Background Sequences

SMGenerate generates biologically relevant background sequences given user-provided values for the following parameters:

1. Number of sequences t.
2. Length of each sequence $n_i \in [min, max]$
3. One of the following options:
   (a) Independent probabilities for the nucleotide symbols in Σ_{DNA}.
   (b) Markov order $m \geq 0$ with a sample DNA sequence.
   (c) Markov order $m \geq 0$ without a sample DNA sequence.

Unlike ABS and RSA, in SMGenerate, the length of a sequence varies within a user specified range. Also, SMGenerate has the ability to use Markov model to generate the background sequences with or without user supplied sample DNA sequence, as explained next.

Consider generation of the background sequences $\mathcal{S} = \{S_i\}_{i=1}^{t}$, where $|S_i| = n_i$, for a Markov order $m \geq 0$. First, SMGenerate counts the number of *l-mers* in the given sample DNA sequence,

where $l = m + 1$, and uses these counts as explained in (3.1). If a sample sequence is not given, then the count for each possible *l-mer* $\in (\Sigma_{\text{DNA}})^l$ is sampled from a uniform distribution in the range $[0, \sum n_i]$.

In both cases, the probability of an *l-mer* ($X_{i,1:l}$), where $i = 1 \ldots |\Sigma_{\text{DNA}}|^l$, is:

$$P(X_{i,1:l}) = \frac{\textit{count}(X_{i,1:l}) + \alpha}{\sum_i \textit{count}(X_{i,1:l}) + \alpha|\Sigma_{\text{DNA}}|^l}. \tag{3.1}$$

$\alpha \geq 0$ is a user defined pseudo count, usually $\alpha = 1.0$, that acts as an additive smoothing term [27] (Laplace smoothing) to correct for zero *l-mer* counts when the supplied DNA sequence sample is small. Next, SMGenerate calculates the entries of the m^{th} order transitional probability matrix as:

$$P(X_{i,l}|X_{i,1:l-1}) = \frac{P(X_{i,1:l})}{\sum_{y \in \Sigma_{\text{DNA}}} P(X_{i,1:l-1}, y)}. \tag{3.2}$$

Lower order Markov transitional probability matrices are recursively generated (from $l - 1$ down to zero order) as in (3.2) using the fact that $P(X_{i,1:l-1})$, used in the next matrix calculation, has already been calculated in the denominator of (3.2). We use these matrices to generate the first m symbols of each sequence S_i such that the symbols' distribution of the whole sequence conforms with the supplied model (Markov order and sample sequences). Figures 3.3a and 3.3b show two sample files each containing five background sequences, for Markov order $m = 0$ and $m = 2$ respectively, generated in FASTA format [28]. Figure 3.3c shows the calculated Markov transitional probability matrices for the pseudo count $\alpha = 1.0$ and for the given sample DNA sequence.

```
>seq1 size:70bp
aattcagttttaacatccatatagctaaaccattctttactttggccggc
taatgaaattggacgttctc
>seq2 size:70bp
ttcaaacagctatatgccccattgaatccatataggtcaattctgacatt
ttgattccagatttttcct
>seq3 size:70bp
atgtttcaacgtttactgcagatttcttattatttgattttatagagttc
aaatggaaaaaaaaattagc
>seq4 size:70bp
taaatgattccgattagtgtcacactatttgcgggtcacttcaattttcg
tttagtaagacactataggg
>seq5 size:70bp
tttcaaaccttggactaacgccattaaatccgaatcaattggcgtatacc
tctcatccgaaagttttttt
```

(a) Background FASTA file (BG0.seq), $m = 0$.

```
>seq1 size:70bp
atctgcaaatagataaggttactcagcaagcagcacagatataaggtggg
tggggttcattactacaggg
>seq2 size:70bp
gatccagcatttggtgaaggggactggatacttcgtgatgtacaaaatgg
tttaattcagaatctacttt
>seq3 size:70bp
caatcagctaacagtgagtgattcaatgaagcgtgataatatacttgata
acattcggacttcaggatct
>seq4 size:70bp
aaattgaatgatactgatccagcatttagcttttactggatgtaaggcgc
tggaaatttcaacatacgat
>seq5 size:70bp
gccatacaaattcttcaactggatgaactttagatttgataaagatataa
atttttttcagcctacaatt
```

(b) Background FASTA file (BG2.seq), $m = 2$.

```
Given Sample:
ttttaatctaacaggattacaattcagcaagct
tgggtatatactccattgatactttaa

Pseudo Count=1.00

Normalized Probability Matrices:
Markov=2
     A       C       G       T
AA: 0.125   0.250   0.250   0.375
AC: 0.375   0.125   0.125   0.375
AG: 0.143   0.429   0.286   0.143
AT: 0.364   0.182   0.091   0.364
CA: 0.333   0.111   0.333   0.222
CC: 0.400   0.200   0.200   0.200
CG: 0.250   0.250   0.250   0.250
CT: 0.250   0.250   0.125   0.375
GA: 0.167   0.167   0.167   0.500
GC: 0.333   0.167   0.167   0.333
GG: 0.286   0.143   0.286   0.286
GT: 0.400   0.200   0.200   0.200
TA: 0.333   0.333   0.083   0.250
TC: 0.286   0.286   0.143   0.286
TG: 0.333   0.167   0.333   0.167
TT: 0.308   0.154   0.231   0.308

Markov=1
    A       C       G       T
A: 0.261   0.217   0.174   0.348
C: 0.429   0.143   0.071   0.357
G: 0.250   0.250   0.333   0.167
T: 0.346   0.154   0.115   0.385

Markov=0
    A       C       G       T
    0.328   0.172   0.141   0.359
```

(c) Markov probabilities file showing the given sample DNA sequence.

Fig. 3.3: Generated DNA background sequences, where $t = 5$, $n = 70$. The Markov order m is as shown in (a) and (b), also the transitional probability matrices, the pseudo count and the sample DNA sequence are shown in (c).

3.2.2 Implanting Structured Motifs

To generate and implant simple as well as structured motifs in the background sequences, SMGenerate requires the following parameters to generate a motif:

1. **The base motif**:
   (a) Number of boxes k; $k = 1$ for a simple motif and $k > 1$ for a structured motif.
   (b) *Or* the actual motif to be implanted as a profile matrix [2] (Figure 3.4).
   (c) *Or* a sequence using Σ_{IUPAC} alphabet and the gap symbol '$-$' (e.g., `ACGG-MATT-GATTK` is a three-box structured motif).
2. **Motif box sizes**: In case of *1a*, the user specifies a range, e.g., $[4, 12]$ such that each box size

is randomly sampled from this range. This is ignored otherwise.

3. **Motif gap sizes**: The user specifies a range for the gap, e.g., $[2, 15]$. Gap lengths for each instance are randomly sampled from this range.
4. **Maximum mismatches per box**: Integer $d_i \geq 0$ that specifies the number of position specific/arbitrary (Figure 3.4) per box mismatches. If a single integer is specified, then the same number of mismatches is applied to each box.
5. **Motif Markov order**: The Markov order $m \geq 0$; used to generate the motif boxes (required for *1a*).
6. **Example/Weights**: Independent probabilities for the symbols in Σ_{DNA} or a sample DNA sequence used to generate the motif boxes (optional).

(a) A simple motif whose instances have $d = 2$ position specific mismatches. The Hamming distance between two instances is at most 2bp.

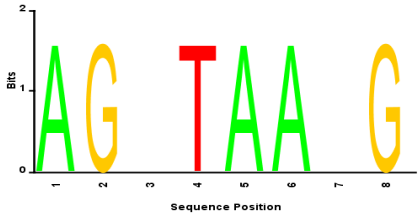
(b) A simple motif whose instances have $d = 2$ arbitrarily positioned mismatches. The Hamming distance between two instances is at most 4bp.

A	1.0	0	0.2	0	1.0	1.0	0.2	0
C	0	0	0.2	0	0	0	0.2	0
G	0	1.0	0.4	0	0	0	0.2	1.0
T	0	0	0.2	1.0	0	0	0.4	0
–	0	0	0	0	0	0	0	0

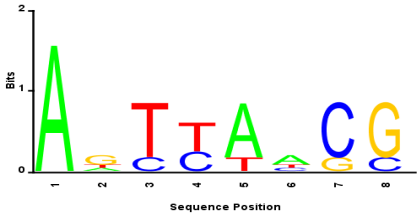
(c) Profile matrix (frequency matrix).

A	1.0	0.2	0	0	0.8	0.6	0	0
C	0	0	0.2	0.4	0	0.2	0.8	0.2
G	0	0.6	0	0	0	0	0.2	0.8
T	0	0.2	0.8	0.6	0.2	0.2	0	0
–	0	0	0	0	0	0	0	0

(d) Profile matrix (frequency matrix).



(e) Motif logo.



(f) Motif logo.

Fig. 3.4: Showing two simple motifs having (a) position specific and (b) arbitrarily positioned mismatches.

To implant the generated motif, the following parameters are required:

1. **Motif implant probability**: the probability of implanting a motif in a given background sequence.

2. **Motif repeat probability**: the probability of repeating a motif in the same sequence given at least one implanted.
3. **Motif complementary strand probability**: the probability of implanting a motif on the complementary strand.

Using these parameters, SMGenerate implants instances of the specified simple/structured motif in the background sequences avoiding overlapping instances and also avoiding partial implants. For example, Figure 3.5 shows a generated file based on the following parameter values:

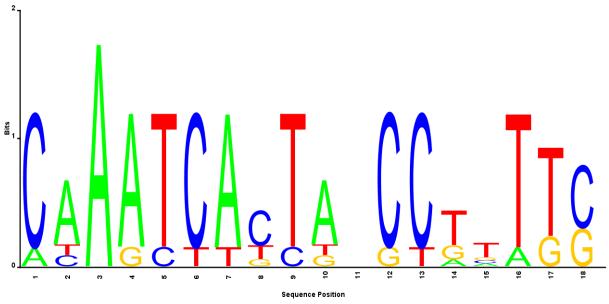
1. **Background file**: 'BG2.seq' (generated in the previous section, Figure 3.3b).
2. **The base motif**: number of boxes $k = 2$.
3. **Motif box sizes**: randomly sampled from $[4, 12]$.
4. **Motif gap sizes**: randomly sampled from $[2, 15]$.
5. **Maximum mismatches per box**: $d_i = 2$, for all $i = 1, 2$, arbitrary per box mismatches.
6. **Motif Markov order**: $m = 0$.
7. **Example/Weights**: 'BG' (this is a special value that instructs SMGenerate to use the background model to generate the motif boxes, so $m = 0$ is ignored).
8. **Motif implant probability**: 100%.
9. **Motif repeat probability**: 30%.
10. **Motif complementary strand probability**: 40%.

3.2.3 Generating Full Dataset

The previous example shows a single file generated corresponding to a single test case. However, SMGenerate can generate the full dataset (the collection of files representing all test cases; based on user provided parameters) by automatically enumerating all possible combinations of the provided values. For example, Table 3.1 shows multiple values for some of the parameters discussed above. This results in generating a total of 66 files representing various test cases, all generated in one shot, without much effort on the user's part. Figure 3.6 contains the file summarizing the full dataset.

```
>seq1 size:70bp
aattcagttttaacCAAATCTCTActaaaccattctCCTATGCggccggct
aatgaaattggacgttctc
>seq2 size:70bp
ttcaaacagctatatgcccCAAACCACTAatataggtcaattctGCATTTC
tgattccagatttttcct
>seq3 size:70bp
atGATGAGGcgttAAGTGATTGGttcttattaCTAATCATTAtagagttCC
GTTTGaaaaaaaaattagc
>seq4 size:70bp
taaAAAATCAGTAttCCGTTTCcCAAAAAGgcgggtcacttcaatCAGTGA
TTTGtaagacactataggg
>seq5 size:70bp
tttGCACAGGtggactaacTGGTGACTTGcCAAAAGGattggcgtatacct
cTAATAATTTGgttttttt
```

(a) Generated file showing two-box structured motif implanted on both strands with repeats.



(b) Implanted motif logo. The 11^{th} position is a variable gap of size $\in [2, 15]$.

Fig. 3.5: Example generated file showing implanted instances of a two-box structured motif.

Figure 3.7 shows two examples out of the 66 files in the dataset. Both files show the background sequences with the implanted motifs along with their respective logos. Clearly SMGenerate allows for flexible generation of comprehensive and biologically relevant datasets.

Table 3.1: Multiple values given to SMGenerate to generate a full dataset in one shot.

Parameters	**Value1**	**Value2**	**Value3**
Background files:	'BG0.seq'	'BG2.seq'	
The base motif boxes:	1	2	
Motif box sizes:	$[4, 12]$		
Motif gap sizes:	$[2, 15]$		
Maximum mismatches per box:	0	2	
Motif Markov order:	0		
Example/Weights:	'BG'[1]		
Motif implant probability:	0%[2]	80%	100%
Motif repeat probability:	0%	30%	
Motif complementary strand probability:	0%	40%	

[1] 'BG' is a special value that instructs SMGenerate to use the background model to generate the motif boxes, so $m = 0$ is ignored.

[2] 0% implanting probability will add the 2 background files as part of the dataset (total files=(2*2*1*1*2*1*1*2*2*2)+2=64+2=66).

```
'bg/' folder contains the original background sequences files.
'instances/' folder contains the implanted motifs' instances details.
'sequences/' folder contains the actual sequences with implanted motifs in the following subfolders:
        '/main/'   have the main promoter sequences.
        '/comp/'   have the main and reverse complement as separate sequences.
        '/concat/' have each sequence appended with its reverse complement.

This dataset has 66 files with the following parameters:
01) Promoter Background Files=
        'bg/BG0.seq' (num_seq=5, lengths=70, markov=0, example/weights= see 'BG0_transProb.txt' file)
        'bg/BG2.seq' (num_seq=5, lengths=70, markov=2, example/weights= see 'BG2_transProb.txt' file)
02) Number of Motif boxes or Actual Motifs={1,2}
03) Motifs Box Sizes={[4 12]}
04) Motifs Gap Sizes={[2 15]}
05) Max Mismatches={0,2} Arbitrary Position.
06) Motif Markov Model Order={0} Pseudo Count=1.00 (Markov model is ignored when Example='BG')
07) Example/Weights for Motif Markov={'BG'}
08) Prob. of Motif Implanting={0,0.8,1}
09) Prob. of Repeats={0,0.3} Non-overlapping
10) Prob. of Implanting on Complement Strand={0,0.4}

No. --> Parameters (as above list) --> Base Motif Implanted --> (Instances in Sequences out of Total)
1  'bg/BG0.seq',1,[4 12],[2 15],0,0,'BG',0,0,0          ---                          (0 in 0 out of 5)
2  'bg/BG0.seq',1,[4 12],[2 15],0,0,'BG',0.8,0,0        GTTTCC                       (4 in 4 out of 5)
3  'bg/BG0.seq',1,[4 12],[2 15],0,0,'BG',0.8,0,0.4      CTTAAATCGC                   (4 in 4 out of 5)
4  'bg/BG0.seq',1,[4 12],[2 15],0,0,'BG',0.8,0.3,0      ACAAG                        (5 in 4 out of 5)
...
9  'bg/BG0.seq',1,[4 12],[2 15],0,0,'BG',1,0.3,0.4      TGATACGTC                    (7 in 5 out of 5)
10 'bg/BG0.seq',1,[4 12],[2 15],2,0,'BG',0.8,0,0        WATNCM                       (4 in 4 out of 5)
...
13 'bg/BG0.seq',1,[4 12],[2 15],2,0,'BG',0.8,0.3,0.4    RTYATTCT                     (4 in 4 out of 5)
14 'bg/BG0.seq',1,[4 12],[2 15],2,0,'BG',1,0,0          ATTAATTT                     (5 in 5 out of 5)
15 'bg/BG0.seq',1,[4 12],[2 15],2,0,'BG',1,0,0.4        CCCCAA                       (5 in 5 out of 5)
16 'bg/BG0.seq',1,[4 12],[2 15],2,0,'BG',1,0.3,0        GATTGTMTGAC                  (5 in 5 out of 5)
17 'bg/BG0.seq',1,[4 12],[2 15],2,0,'BG',1,0.3,0.4      AGTTAACT                     (6 in 5 out of 5)
...
20 'bg/BG0.seq',2,[4 12],[2 15],0,0,'BG',0.8,0.3,0      TATACAGCTT-AACTAGA           (7 in 5 out of 5)
21 'bg/BG0.seq',2,[4 12],[2 15],0,0,'BG',0.8,0.3,0.4    AATTTCACTT-GCTACA            (4 in 4 out of 5)
...
25 'bg/BG0.seq',2,[4 12],[2 15],0,0,'BG',1,0.3,0.4      TTTAATTAA-CCCTAG             (7 in 5 out of 5)
26 'bg/BG0.seq',2,[4 12],[2 15],2,0,'BG',0.8,0,0        MTCTCGCTCTT-CATAGTAT         (5 in 5 out of 5)
27 'bg/BG0.seq',2,[4 12],[2 15],2,0,'BG',0.8,0,0.4      AAATCTAAGMT-AAAAACTATC       (3 in 3 out of 5)
28 'bg/BG0.seq',2,[4 12],[2 15],2,0,'BG',0.8,0.3,0      CTTTCGTGAAA-ATCTCGTTTCCT     (6 in 5 out of 5)
29 'bg/BG0.seq',2,[4 12],[2 15],2,0,'BG',0.8,0.3,0.4    MCGT-TGTTAG                  (4 in 4 out of 5)
...
32 'bg/BG0.seq',2,[4 12],[2 15],2,0,'BG',1,0.3,0        GATATACTG-AGCGAG             (6 in 5 out of 5)
33 'bg/BG0.seq',2,[4 12],[2 15],2,0,'BG',1,0.3,0.4      CAAATCACTA-CCTTTTC           (8 in 5 out of 5)
34 'bg/BG2.seq',1,[4 12],[2 15],0,0,'BG',0,0,0          ---                          (0 in 0 out of 5)
35 'bg/BG2.seq',1,[4 12],[2 15],0,0,'BG',0.8,0,0        TTCATAGTTTGA                 (4 in 4 out of 5)
36 'bg/BG2.seq',1,[4 12],[2 15],0,0,'BG',0.8,0,0.4      TTTAGAC                      (4 in 4 out of 5)
...
43 'bg/BG2.seq',1,[4 12],[2 15],2,0,'BG',0.8,0,0        GTGAGCTTCTGA                 (5 in 5 out of 5)
44 'bg/BG2.seq',1,[4 12],[2 15],2,0,'BG',0.8,0,0.4      GACTGATAA                    (3 in 3 out of 5)
45 'bg/BG2.seq',1,[4 12],[2 15],2,0,'BG',0.8,0.3,0      CAACA                        (5 in 4 out of 5)
46 'bg/BG2.seq',1,[4 12],[2 15],2,0,'BG',0.8,0.3,0.4    TGACTTCT                     (5 in 4 out of 5)
...
49 'bg/BG2.seq',1,[4 12],[2 15],2,0,'BG',1,0.3,0        ACAGCGAA                     (7 in 5 out of 5)
50 'bg/BG2.seq',1,[4 12],[2 15],2,0,'BG',1,0.3,0.4      YTTAY                        (6 in 5 out of 5)
51 'bg/BG2.seq',2,[4 12],[2 15],0,0,'BG',0.8,0,0        ACCTTCCAGGG-TCATA            (3 in 3 out of 5)
52 'bg/BG2.seq',2,[4 12],[2 15],0,0,'BG',0.8,0,0.4      TCTGATGAGATA-CTTTCCG         (5 in 5 out of 5)
...
58 'bg/BG2.seq',2,[4 12],[2 15],0,0,'BG',1,0.3,0.4      AGGTGAAC-GATGGGTAC           (6 in 5 out of 5)
59 'bg/BG2.seq',2,[4 12],[2 15],2,0,'BG',0.8,0,0        TSRRA-SRTAR                  (4 in 4 out of 5)
...
64 'bg/BG2.seq',2,[4 12],[2 15],2,0,'BG',1,0,0.4        AATACGAT-ACTAAACTCGCT        (5 in 5 out of 5)
65 'bg/BG2.seq',2,[4 12],[2 15],2,0,'BG',1,0.3,0        GSTTTCAAGG-AYGAGG            (6 in 5 out of 5)
66 'bg/BG2.seq',2,[4 12],[2 15],2,0,'BG',1,0.3,0.4      CRAGCC-GCAAGGTTG             (5 in 5 out of 5)
```

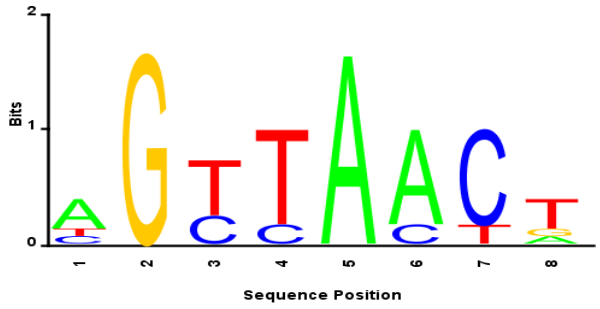
Fig. 3.6: The generated dataset's summary file (some lines are omitted for brevity).

```
>seq1 size:70bp
aattAGCCAACTacatccatatagcAGTTAACGtctttactttggc
cggctaatgaaattggacgttctc
>seq2 size:70bp
ttcaaacagctatatgccccattgaatccatataAGGTAACTctga
cattttgattccagatttttcct
>seq3 size:70bp
atgtttcaacgtttacAGTTAATGtcttattatttgattttataga
gttcaaatggaaaaaaaattagc
>seq4 size:70bp
taaatgattccgattagtgtcacactatttgcgggtcacttcaatt
ttcgAGCTAACAgacactataggg
>seq5 size:70bp
tttcaaaccttggactaacgccattaaatccgaatcaTGTTAACTa
tacctctcatccgaaagttttttt
```

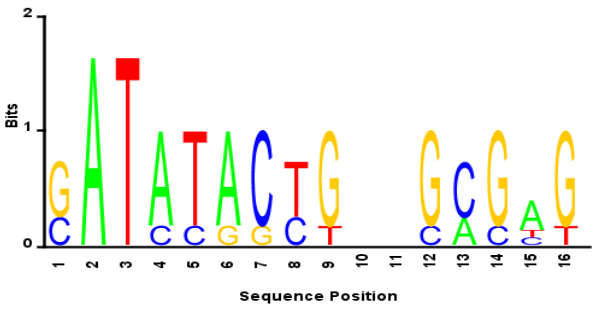
(a) File No.17 showing one-box simple motif implanted on both strands with repeats.

```
>seq1 size:70bp
aattcaCATACACTGtccatatGGCGCGaccattctttactttggc
cggctaatgaaattggacgttctc
>seq2 size:70bp
ttcaaaGATATACCGgccACAGAGaatccatataggtGATCTACTT
cattttgattccaTGAGAGttcct
>seq3 size:70bp
atgtttcaacgtGATATAGTGatttAGCCTGatttgattttataga
gttcaaatggaaaaaaaattagc
>seq4 size:70bp
taaaCATATACCGttagCGCGATactatttgcgggtcacttcaatt
ttcgtttagtaagacactataggg
>seq5 size:70bp
tttcaaaccttggactaacgccattaaatccgaatcaattggcGAT
ATGCTGtcAGCGAGaagttttttt
```

(b) File No.32 showing two-box structured motif implanted with repeats.



(c) File No.17 implanted simple motif logo.



(d) File No.32 implanted structured motif logo. The 10^{th} position is a variable gap of size $\in [2, 15]$, while the 11^{th} position is a single highly degenerate position.

Fig. 3.7: Two examples from the generated dataset showing one-box and two-box implanted motifs.

3.3 Discussion

In this chapter we have argued that existing synthetic data generators lack some important features to generate adequate test cases for the motif extraction problem in general and the structured motif extraction problem specifically. This motivated the development of SMGenerate, a flexible and comprehensive synthetic dataset generator for the simple and structured motif extraction problems. Table 3.2 summarizes the features of SMGenerate and compares them against RSA and ABS. SMGenerate proved to be a valuable tool for assessing the performance of our proposed structured motif extraction algorithm under various interesting test cases, discussed in Chapter 6. A MATLAB implementation and an online version of SMGenerate can be accessed from *http://bioproject.syr.edu/smtools*.

Table 3.2: Comparing the features of SMGenerate, RSA and ABS.

Features	SMGenerate	RSA	ABS
Generating Background Sequences:			
Fixed length sequences	✔	✔	✔
Variable length sequences	✔		
Independent DNA symbols probabilities	✔	✔	✔
Based on Markov model *with* sample DNA sequence	✔	✔	
Based on Markov model *without* sample DNA sequence	✔		
Implanting Motifs:			
Generates and implants simple motifs	✔	✔	✔*
Implants user supplied simple motifs	✔	✔	
Generates and implants structured motifs	✔		
Implants user supplied structured motifs	✔		
User supplied probability for implanting a motif	✔		✔
User supplied probability for repeating a motif	✔		
User supplied probability for implanting on the comp. strand	✔		
User supplied position specific mismatches among instances	✔	✔	
User supplied arbitrarily positioned mismatches among instances	✔		
Avoids overlapping multiple instances	✔		✔
Avoids partial implants near sequence boundary	✔		
Easily generates multiple test cases (complete dataset) in one shot	✔		

* ABS does not generate simple motifs, but provide a list of real simple motifs to be implanted.

Chapter 4
Pairwise Alignment of Structured Motifs

This chapter addresses the problem of aligning a pair of structured motifs, and serves as the basis for providing an *affinity* measure between such sequences to aid in multiple alignment in Chapter 5.

The Needleman-Wunsch *global* alignment [29] and the Smith-Waterman *local* alignment [22] algorithms are the most widely used exact pairwise alignment algorithms [1]. The former identifies similarity regions considering the full length of the two given sequences, whereas the latter identifies the most similar subsequences. Although these algorithms are very useful in many bioinformatics applications, such as determining the evolutionary distance between a pair of biological sequences, they are unsatisfactory for alignment of structured motifs; specifically, they do not take into account constraints on the location of the gaps and the allowed length of such gaps within each sequence.

A recent survey by Mahony *et al.* [30], concluded that the best pairwise motif alignment algorithm is to use an ungapped extended Smith-Waterman local alignment (*SWLocal*) to identify a core motif alignment region, used as an alignment guide, and then extend the alignment beyond the core. Although, their algorithm works best for aligning simple motifs, no tests have been done on aligning structured motifs.

Our proposed *SMAlign* algorithm, based on the Needleman-Wunsch global alignment algorithm

(*NWGlobal*), first solves a relaxed version of the problem using dynamic programming, then a branch and bound (B&B) search algorithm is applied, followed by recovery of the optimal alignment from the best path obtained by B&B. Experimental results show that SMAlign is optimal in accomplishing the alignment task with computational complexity comparable to the previously mentioned local and global alignment algorithms.

4.1 Overview

In this section, first we illustrate an example of the structured motif alignment problem. Second, we describe how the problem constraints are addressed and present an outline of the new SMAlign algorithm.

Example: Consider the following structured motifs (sequences with gap constraints):

$$\mathcal{M} = \texttt{AGCAT[0,3]TCG[2,5]GCTC}$$

$$\mathcal{N} = \texttt{AGC[14,30]TCTC}$$

$\mathcal{M}$ is a three-box structured motif whose first gap location is from 0 to 3bp long, and its second gap location is 2 to 5bp long. Similarly, $\mathcal{N}$ is a two-box structured motif whose first and only gap location is 14 to 30bp long. A natural question is whether the two can be considered instances of the same structured motif. Hence our objective is to align the two sequences maximizing a similarity score (based on a user-specified similarity scoring matrix **U**, such as in Table 4.1) subject to the given gap constraints.

Table 4.1: Default similarity scoring matrix for SMAlign.

U	A	C	G	T	–
A	4	-1	-1	-1	0
C	-1	4	-1	-1	0
G	-1	-1	4	-1	0
T	-1	-1	-1	4	0
–	0	0	0	0	0

Table 4.2 contains the results of applying *NWGlobal* [29] and the *ungapped extended SWLocal* [30] algorithms to align $\mathcal{M}$ and $\mathcal{N}$ along with the desired alignment that maximizes the score subject

Table 4.2: Example: the desired alignment of the pair $\mathcal{M}$ and $\mathcal{N}$ is shown compared with the results of *NWGlobal* and *ungapped extended SWLocal* alignments using three input forms. None of the results matched the desired alignment in maximizing the similarity score while satisfying the gap constraints.

Original Sequences with Gap Constraints	$\mathcal{M}$ = AGCAT[0,3]TCG[2,5]GCTC $\mathcal{N}$ = AGC[14,30]TCTC								
Correct Desired Alignment: **Score** = 20 **Satisfactory** = Yes	AGCAT---TCG----GCTC-- ||| || AGC----------TCTC								
Input (3 Forms)	**1-No Gaps Form:** AGCATTCGGCTC AGCTCTC	**Score**	**Satisfactory**	**2-Compact Form:** AGCAT-TCG-GCTC AGC-TCTC	**Score**	**Satisfactory**	**3-Expanded Form:** AGCATTCG--GCTC AGC----------TCTC	**Score**	**Satisfactory**
NWGlobal Alignment Gap Penalty: open=8, ext=1 (Alignment score using **U**)	AGCATTCGGCTC |||X ||| AGCT-----CTC	11 (23)	No	AGCAT-TCG-GCTC ||| | ||| AGC-T------CTC	15 (28)	No	AGCATTCG--G-------CTC ||| ||| AGC----------TCTC	10 (24)	No
Ungapped Extended SWLocal Alignment Gap Penalty: open=∞, ext=∞	AGCATTCGGCTC |||XX|| AGCTCTC-----	18	No	AGCAT-TCG-GCTC ||| | || AGC-TCTC------	24	No	AGCATTCG--GCTC------- ||| AGC----------TCTC	12	Yes

to the gap constraints. Since neither *NWGlobal* nor *SWLocal* take gap constraints as parameters, we have experimented with presenting the inputs to these algorithms in three different forms:

1. *No Gaps*: Concatenate the structure motifs' boxes together, ignoring the gap constraints.
2. *Compact*: Substitute each gap range with a single gap symbol.
3. *Expanded*: Substitute each gap range with the minimum number of gap symbols in the range.

Although neither algorithm produced the desired alignment, *SWLocal* comes close in producing a satisfactory, although not optimal, alignment when given the input in *expanded* form.

Another alternative solution for this combinatorial optimization problem is to use an exhaustive search approach. In our example, this requires presenting 272 different inputs to *NWGlobal*, spanning all possible gap sizes in the input sequences. However, this approach is computationally prohibitive as it depends on the number of gaps and the sizes of the gaps' ranges in the input structured motifs.

It is apparent that neither algorithm is suitable for the task, thus motivating the development of a new efficient alignment algorithm, SMAlign, that maximizes the similarity score subject to the following constraints:

1. *Gap locations*: gaps are only opened in the specified locations within each given sequence.

2. *Gaps minimum size*: gaps' sizes should not be less than the specified minimums.
3. *Gaps maximum size*: gaps' sizes should not exceed the specified maximums.
4. *Max sequence length*: the resulting alignment should not extend any of the given sequences beyond a specified maximum length (optional) even if the individual gap locations and their respective minimum and maximum sizes were not violated.

SMAlign modifies and extends *NWGlobal* by solving the problem in three stages:

1. *Dynamic programming*: We solve a relaxed version of the problem satisfying constraints 1 and 2; the resulting pointers matrix represents a directed acyclic graph (DAG).
2. *Branch and Bound*: We search for the maximum scoring alignment that satisfies constraints 3 and 4 as an optimal path finding problem in the DAG, which represents a reduced search space of possible paths (alignments).
3. *Optimal alignment*: We recover the optimal alignment, satisfying all constraints, and output its frequency matrix [2] with its gap ranges.

The following section presents the notation used and the details of each of the three stages of SMAlign.

4.2 SMAlign: Structured Motif Alignment

We first revisit the definition of a structured motif, given in Section 1.2, in more details. Let $\Sigma = \{A, C, G, T, -\}$ be the DNA alphabet, plus the gap symbol denoted by '$-$', and $\Sigma_{\text{IUPAC}} = \{A, C, G, T, R, Y, K, M, S, W, B, D, H, V, N\}$ be the extended DNA alphabet. We then formally present a DNA sequence with gap constraints (a structured motif) as a 4-tuple $\mathcal{M} = \langle \mathbf{M}, \mathbf{G}^M, L^M, C^M \rangle$. $\mathbf{M}$ is the input DNA sequence given in *compact form* as a *literal string* or a *frequency matrix*:

$$\mathbf{M} = \mathbf{M}_1 \oplus \mathbf{Z}_1 \oplus \ldots \oplus \mathbf{M}_{b_M - 1} \oplus \mathbf{Z}_{b_M - 1} \oplus \mathbf{M}_{b_M}$$

where $\oplus$ denotes the concatenation operator between $\mathbf{M}_i$ boxes and $\mathbf{Z}_j$ gap locations for $i \in [1, b_M]$ and $j \in [1, b_M - 1]$ (b_M is the total number of boxes in motif $\mathcal{M}$). In a *literal string* representation, $\mathbf{M}_i \in (\Sigma_{\text{IUPAC}})^{l_i}$ and $\mathbf{Z}_j \in \{-\}^{x_j}$. In a *frequency matrix* representation:

$$\mathbf{M}_i = \begin{bmatrix} f_{A,1} & \cdots & f_{A,l_i} \\ f_{C,1} & \cdots & f_{C,l_i} \\ f_{G,1} & \cdots & f_{G,l_i} \\ f_{T,1} & \cdots & f_{T,l_i} \\ f_{-,1} & \cdots & f_{-,l_i} \end{bmatrix}_{5\times l_i} \qquad \mathbf{Z}_j = \begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ 1 & \cdots & 1 \end{bmatrix}_{5\times x_j}$$

where $l_i > 0$. In the matrix representation of $\mathbf{M}_i$, $f_{\alpha,c}$ is the relative frequency of letter $\alpha \in \Sigma$ in column $c \in [1, l_i]$ such that $f_{-,c} < 1$. In $\mathbf{Z}_j$, $x_j = 1$ since $\mathbf{M}$ is in *compact* form. $\mathbf{G}^M$ represents the gap constraints matrix:

$$\mathbf{G}^M = \begin{bmatrix} g_1^{\min} & \cdots & g_{b_M-1}^{\min} \\ g_1^{\max} & \cdots & g_{b_M-1}^{\max} \end{bmatrix}_{2\times(b_M-1)}$$

where $[g_j^{\min}, g_j^{\max}]$ is the range for the j^{th} gap location in $\mathbf{M}$ and $0 \le g_j^{\min} \le g_j^{\max}$. L^M and C^M are optional parameters that define the maximum allowed length of $\mathbf{M}$ in the final alignment (default=∞) and the number of aligned sequences within $\mathbf{M}$ (default=1) respectively. Thus, formally, our inputs in Table 4.2 are:

$$\mathcal{M} = \langle \texttt{AGCAT-TCG-GCTC}, \left[\begin{smallmatrix} 0 & 2 \\ 3 & 5 \end{smallmatrix}\right], \infty, 1 \rangle$$

$$\mathcal{N} = \langle \texttt{AGC-TCTC}\qquad\quad, \left[\begin{smallmatrix} 14 \\ 30 \end{smallmatrix}\right], \infty, 1 \rangle$$

SMAlign takes $\mathcal{M}$ and $\mathcal{N}$ as inputs. It converts any input *literal strings* into *frequency matrices* and then converts these frequency matrices from *compact* to *expanded* form, by setting $x_j = g_j^{\min}$, and works with that form hereafter. Thus, we denote the total length of $\mathbf{M}$ (same for $\mathbf{N}$) in its *expanded* form as:

$$l_M = l_{b_M} + \sum_{j=1}^{b_M-1} (l_j + g_j^{\min}).^{\S} \tag{4.1}$$

[§]This is the same as Equation (1.1).

If $0 < L^M < l_M$, then SMAlign sets $L^M = l_M$. Also, the i^{th} column, in the *expanded frequency matrix*, is denoted by $M_{*,i}$ for $i \in [1, l_M]$.

SMAlign augments the gap constraints matrix with a third row specifying the actual positions immediately after which gaps are allowed. The j^{th} entry in that row is:

$$G^M_{3,j} = \sum_{y=1}^{j} (g^{\text{min}}_{y-1} + l_y), \text{ where } g^{\text{min}}_0 = 0. \tag{4.2}$$

4.2.1 Stage1: Dynamic Programming

In this stage, we solve a relaxed version of the problem (satisfying constraints 1 and 2 only) using dynamic programming (DP) to generate the scores $\mathbf{S}$ and the pointers $\mathbf{P}$ matrices (see Figure 4.1). SMAlign's DP is based on *NWGlobal* with the following differences and extensions:

- Our scoring matrix $\mathbf{S} = [S_{i,j,k}]$ is three-dimensional, where $i \in [0, l_N]$, $j \in [0, l_M]$ and $k \in \{1, 2, 3\}$. We store all three values of the diagonal, left and up scores in $\mathbf{S}$, sorted in descending order along the third dimension whereas *NWGlobal* only stores the maximum scores.
- Since $\mathbf{S}$ stores all three scores, we also store the associated directions in $\mathbf{P} = [P_{i,j}]$. Each digit in $\mathbf{P}$'s cells represents a direction where 1 denotes the diagonal (towards $P_{0,0}$), 2 denotes the left, and 3 denotes the up, prioritized from left to right based on their respective scores.
- Not all three directions are allowed in all cells. Diagonal directions are allowed in cells $\mathbf{P}_{[1,l_N],[1,l_M]}$. However, up directions are only allowed in columns $j \in \{G^M_{3,*}\}^{\S} \cup \{0, l_M\}$ and left directions are only allowed in rows $i \in \{G^N_{3,*}\} \cup \{0, l_N\}$.
- A new *forward gaps* $\mathbf{FG}$ sparse matrix is required to store gaps data for each cell in $\mathbf{P}$ (section 4.2.1.2).
- Two new user specified parameters are introduced, λ and μ, to calculate an *affine match bonus* score controlling the final alignment (section 4.2.1.3).

[§] $\{G^M_{3,*}\}$ = the set of all entries in the 3^{rd} row of matrix $\mathbf{G}^M$.

4.2.1.1 *Scores and Pointers Matrices*

We first initialize the matrices and then calculate their entries for $i, j > 0$. We set $S_{0,0,k} = 0$, $P_{0,0} = 0$, $P_{0,j} = 2$, and $P_{i,0} = 3$. Also, we initialize the first row and column of $\mathbf{S}$ using the gap

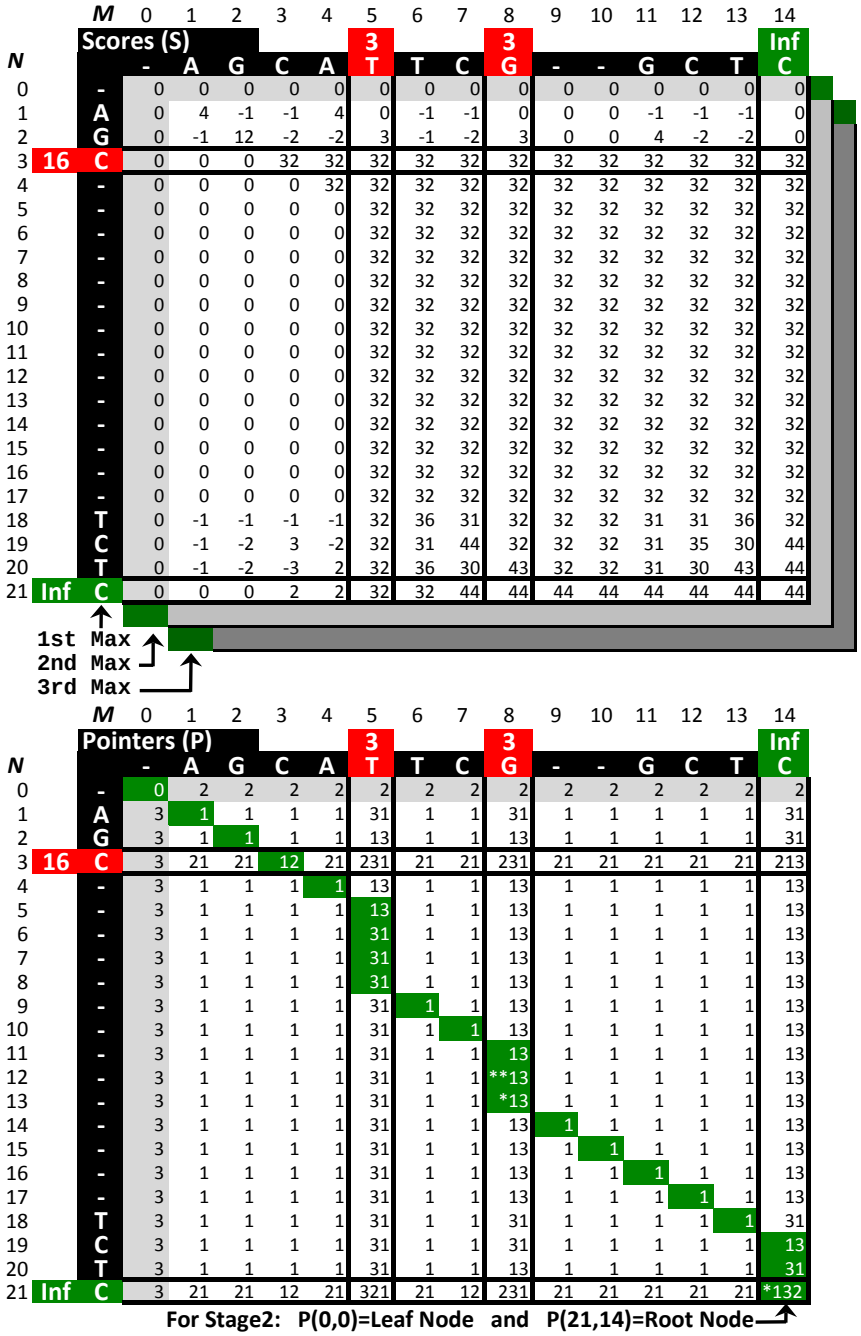


Fig. 4.1: $\mathbf{S}$ and $\mathbf{P}$ matrices, where $\mathbf{P}$ represents a DAG. Desired alignment (path) is shown. The main direction of '*' marked cells, $P_{21,14}$ and $P_{13,8}$, have changed from diagonal to up during B&B. '**' marked cell, $P_{12,8}$, is the maximum score *satisfactory node* found during B&B with its main direction changed from diagonal to up as well.

scores in $\mathbf{U}$:

$$\begin{aligned} S_{0,j,k} &= S_{0,j-1,k} + U_{-,*}\ \widetilde{M}_{*,j} \\ S_{i,0,k} &= S_{i-1,0,k} + U^{\mathrm{T}}_{*,-}\ \widetilde{N}_{*,i} \end{aligned} \tag{4.3}$$

where $\widetilde{\mathbf{M}}$ and $\widetilde{\mathbf{N}}$ are as defined in (4.5).

Next, we calculate the other entries of $\mathbf{S}$ and $\mathbf{P}$. $S_{i,j,*}$ is a vector containing the diagonal ($\mathcal{D}$), left ($\mathcal{L}$) and up ($\mathcal{U}$) scores, in descending order, representing (mis)match, gap in $\mathbf{N}$, and gap in $\mathbf{M}$ respectively. Their values are:

$$\begin{aligned} \mathcal{D} &= S_{i-1,j-1,1} + \widetilde{N}^{\mathrm{T}}_{*,i}\mathbf{U}\widetilde{M}_{*,j} + \lambda B_{i,j} \\ \mathcal{L} &= S_{i,j-1,1} \quad + U_{-,*}\ \widetilde{M}_{*,j} + h_{N,i} \\ \mathcal{U} &= S_{i-1,j,1} \quad + U^{\mathrm{T}}_{*,-}\ \widetilde{N}_{*,i} + h_{M,j} \end{aligned} \tag{4.4}$$

where

$$\widetilde{M}_{*,j} = \begin{cases} \frac{\left(f_{A,j}, f_{C,j}, f_{G,j}, f_{T,j}, 0\right)^{\mathrm{T}}}{\|f_{A,j}, f_{C,j}, f_{G,j}, f_{T,j}, 0\|_1} & \text{if } f_{-,j} < 1 \\ M_{*,j} & \text{otherwise} \end{cases}$$

$$h_{M,j} = \begin{cases} 0 & \text{if } j \in \{G^{M}_{3,*}\} \cup \{l_M\} \\ -\infty & \text{otherwise} \end{cases} \tag{4.5}$$

$\lambda B_{i,j} =$ Affine Match Bonus (defined in 4.2.1.3).

$\widetilde{N}_{*,i}$ and $h_{N,i}$ are defined similarly. The purpose of using $\widetilde{\mathbf{M}}$ instead of $\mathbf{M}$ is to eliminate the effect of partial gaps ($f_{-,j} < 1$) to give an unbiased alignment for weakly conserved signals. SMAlign also provides an option to ignore terminal gaps regardless of gap values in $\mathbf{U}$.

$\mathbf{P}$ matrix entries store the associated directions from which the scores in $\mathbf{S}$ are calculated preserving the descending order of their respective scores. For instance, $P_{21,8} = 231$ indicates that the maximum score is associated with the left direction, thus the *main direction*, followed by the score of the up direction and then the diagonal direction has the lowest score. Likewise, $P_{20,8} = 13$ indicates that the maximum score in that cell propagated from the diagonal direction and the least score is from the up direction; left direction is not allowed in cell $P_{20,8}$. The path formed

by following the main directions from one cell to another is identified as the *main path*.

4.2.1.2 Forward Gaps Matrix

The *forward gaps* associated with cell $P_{i,j}$ are gaps opened in $\mathbf{M}$ and $\mathbf{N}$ while following the main path from $P_{i,j}$ to $P_{0,0}$ and are stored in $\mathbf{FG}^M = [FG^M_{i,j,k_M}]$ and $\mathbf{FG}^N = [FG^N_{i,j,k_N}]$ matrices respectively, where $k_M \in [1, b_M - 1]$ and $k_N \in [1, b_N - 1]$. Entries of these three-dimensional matrices are calculated during the DP stage. For instance, in Figure 4.1, $FG^M_{10,7,*} = \langle 3, 0\rangle$ and $FG^N_{10,7,*} = \langle 0\rangle$ imply that following the main path from $P_{10,7}$ to $P_{0,0}$ opens only three gaps in the *first* gap location of $\mathbf{M}$, but no gaps in its *second* gap location or in $\mathbf{N}$. Similarly, $FG^M_{3,10,*} = \langle 0, 0\rangle$ and $FG^N_{3,10,*} = \langle 7\rangle$ shows that following the main path from $P_{3,10}$ to $P_{0,0}$ opens no gaps in $\mathbf{M}$, but opens seven gaps in $\mathbf{N}$. We define the matrix $\mathbf{FG}$ such that $FG_{i,j,*} = \langle FG^M_{i,j,*}, FG^N_{i,j,*}\rangle$, which we will use in the next stage (section 4.2.2).

4.2.1.3 Affine Match Bonus

Analogous to the affine gap concept in *NWGlobal*, the purpose of the affine match bonus score, $\lambda B_{i,j}$ in (4.4), is to provide a mechanism to bias the alignment score towards contiguous substring matches, within each box, rather than segmented subsequence matches. Depending on a user specified value for $\lambda \geq 0$, $B_{i,j}$ is the *score cumulative product* of the previous contiguous matches when $0 < \lambda \leq 1$ or $B_{i,j}$ is the *count cumulative sum* of the previous contiguous matches when $\lambda > 1$.

We determine a score to be a match based on a user specified value for $\mu \in [0, 1]$, the nucleotide purity level, such that:

$$\widetilde{N}^{\mathrm{T}}_{*,i}\mathbf{U}\widetilde{M}_{*,j} \geq \mu(\mathbf{U}_{\max} - \mathbf{U}_{\min}) + \mathbf{U}_{\min} \tag{4.6}$$

This allows us to count degenerate matches in the bonus score when $\mu < 1$. For example, given the pair of sequences `TCAC[1,5]TCCT` and `TGAC[1,3]TCAY`, Table 4.3 shows two possible alternative alignments. Although the left alignment is optimal without bonus score ($\lambda = 0$), the right alignment might be more desirable as it maximizes the number of contiguous aligned nucleotides. The affine

match bonus score, controlled by λ and μ, biases the score towards the new optimal right alignment. When $\lambda = 0$, we consider the score as a *similarity* measure. But when $\lambda > 0$, we consider the score as an *affinity* measure that we found to be very useful for structured motif multiple alignment discussed in Chapter 5.

4.2.2 Stage2: Branch and Bound Search

Branch and Bound (B&B) is a well known paradigm for solving combinatorial optimization problems [31, 32]. It consists of a branching strategy, a search strategy, and bounding conditions to prune non-promising search paths. Our branching strategy is already determined by $\mathbf{P}$ when used as a directed acyclic graph (DAG). The problem then reduces to an optimal path finding problem, in the DAG, from the root node P_{l_N,l_M} to the leaf node $P_{0,0}$ that satisfies constraints 3 and 4. A path is considered a feasible solution and further exploration is stopped when we find a *satisfactory node* $P_{i,j}$ (defined in 4.2.2.1) along this path. Therefore, the best scoring alignment is represented by a maximum scoring *satisfactory node* among all encountered. Notice that any attempts to satisfy constraints 3 and 4 during the previous stage may lead to a suboptimal solution.

We define the score for a full path passing through node $P_{i,j}$ as:

$$PS_{i,j,k} = AS_{i,j} + S_{i,j,d(k)} \tag{4.7}$$

Table 4.3: Best scoring alignment reported by SMAlign (in bold) for different values of λ and μ

λ	μ	TCAC-TCCT |X|| ||X: TGAC-TCAY			-----TCAC-TCCT |||: TGAC-TCAY-----		
0	–	$x =$	$5\times4+1.5-2$	=**19.5**	$y =$	$3\times4+1.5$	=13.5
0.5	1.0	$x+$	$\lambda(4+4)$	=**23.5**	$y+$	$\lambda(4+16)$	=23.5
1.0	1.0	$x+$	$\lambda(4+4)$	=27.5	$y+$	$\lambda(4+16)$	=**33.5**
3.0	1.0	$x+$	$\lambda(1+1)$	=**25.5**	$y+$	$\lambda(1+2)$	=22.5
7.0	1.0	$x+$	$\lambda(1+1)$	=33.5	$y+$	$\lambda(1+2)$	=**34.5**
0.5	0.5	$x+$	$\lambda(4+4)$	=23.5	$y+$	$\lambda(4+16+64)$	=**55.5**
1.0	0.5	$x+$	$\lambda(4+4)$	=27.5	$y+$	$\lambda(4+16+64)$	=**97.5**
3.0	0.5	$x+$	$\lambda(1+1)$	=25.5	$y+$	$\lambda(1+2+3)$	=**31.5**
7.0	0.5	$x+$	$\lambda(1+1)$	=33.3	$y+$	$\lambda(1+2+3)$	=**55.5**

where $S_{i,j,d(k)}$[§] is the best possible score for the path segment from $P_{i,j}$ to the leaf node for each of its directions $k \in \{1,2,3\}$ and $d = \langle \mathcal{D}, \mathcal{L}, \mathcal{U} \rangle$. $AS_{i,j}$ is the actual score for the path segment from the root node to $P_{i,j}$, reached through a possible path during B&B and calculated by setting $AS_{l_N,l_M} = 0$, and in general:

$$AS_{i,j} = -S_{i,j,1} + \begin{cases} AS_{i+1,j+1} + S_{i+1,j+1,\mathcal{D}} & \text{if } \textit{ParentDir}{=}1 \\ AS_{i,j+1} \quad + S_{i,j+1,\mathcal{L}} & \text{if } \textit{ParentDir}{=}2 \\ AS_{i+1,j} \quad + S_{i+1,j,\mathcal{U}} & \text{if } \textit{ParentDir}{=}3 \end{cases} \tag{4.8}$$

for $i < l_N$ and $j < l_M$. *ParentDir* is the direction followed from the parent node to reach $P_{i,j}$.

4.2.2.1 Satisfactory Node

To define a *satisfactory node*, we first need to define a node's *backward gaps*. Analogous to *forward gaps*, *backward gaps* are gaps opened in $\mathbf{M}$ and $\mathbf{N}$ while following a possible path, during B&B, from the root node to $P_{i,j}$ and are stored in the vectors BG^M_{i,j,k_M} and BG^N_{i,j,k_N}, respectively, for each node. For instance, in the path shown in Figure 4.1, $BG^M_{5,5,*} = \langle 3,2 \rangle$ and $BG^N_{5,5,*} = \langle 0 \rangle$ indicate that following the path from the root node to $P_{5,5}$, three gaps were opened in the first gap location and two gaps in the second gap location of $\mathbf{M}$ and none opened in $\mathbf{N}$. We define the vector $BG_{i,j,*} = \langle BG^M_{i,j,*}, BG^N_{i,j,*} \rangle$.

Therefore, a *backward satisfactory node* $P_{i,j}$ satisfies the following two conditions:

1. Maximum allowed gaps (constraint 3):

$$BG_{i,j,*} \leq \textit{MaxG} \tag{4.9}$$

2. Maximum $\mathbf{M}$ and $\mathbf{N}$ lengths (constraint 4):

$$\begin{aligned} \|BG^M_{i,j,*}\|_1 &\leq L^M - l_M \\ \|BG^N_{i,j,*}\|_1 &\leq L^N - l_N \end{aligned} \tag{4.10}$$

[§]For example, $S_{i,j,d(3)} = S_{i,j,\mathcal{U}}$ is the up direction score at $P_{i,j}$.

where ***MaxG*** $= \langle G^M_{2,*}, G^N_{2,*} \rangle - \langle G^M_{1,*}, G^N_{1,*} \rangle$ is the maximum allowed gap openings within a path for each gap location per entry. On the other hand, if we replace BG with the *total gaps* $TG = FG+BG$ and the same two conditions are satisfied, then $P_{i,j}$ is a *satisfactory node* as it is both *backward* and *forward satisfactory*.

4.2.2.2 B&B Search Strategy

We first define the minimum score gain when gapping as:

$$\alpha = \min_{>0}(S_{*,*,1} - S_{*,*,2}) \tag{4.11}$$

Also, the number of *forward gaps*, in the current path, that violate the ***MaxG*** limit is:

$$X_{i,j} = \sum_{c} (TG_{i,j,c} - \min(TG_{i,j,c}, \textbf{\textit{MaxG}}_c)) \tag{4.12}$$

where $c \in [1, (b_M - 1) + (b_N - 1)]$. Then the adjusted main path score is defined as:

$$\widetilde{PS}_{i,j,\textbf{\textit{MainDir}}} = PS_{i,j,\textbf{\textit{MainDir}}} - \alpha X_{i,j} \tag{4.13}$$

Next, we use an *A** search strategy to efficiently visit our DAG nodes by using a priority queue sorted based on the following criteria:

1. Descending by $\widetilde{PS}_{i,j,\textbf{\textit{MainDir}}}$
2. Ascending by $X_{i,j}$

$\widetilde{PS}_{i,j,\textbf{\textit{MainDir}}}$ is non-increasing along a path and will never underestimate the actual final path score. This is based on the DP property and the fact that the path score will only decrease with added gap constraints. Thus, it is a consistent heuristic for our search priority. However, the optimal path also depends on the number of gaps opened such that it does not exceed a set limit. So, our second criterion promotes nodes that have less chance of violating that limit and it never overestimates the number of *forward gaps* that violate the limit, so it is an admissible heuristic.

Since both of our criteria are admissible, but not consistent, our B&B algorithm (Figure 4.2) is complete and always finds a maximum scoring *satisfactory node*. Yet, its efficiency has room for improvement since using an inconsistent heuristic to search a DAG, as opposed to a tree, will result in expansion of revisited nodes. Therefore, we improve the efficiency by using *bounding conditions* and a *dominance relation* [32] as explained next.

1: **function** BNB($\mathbf{S}, \mathbf{P}, \mathbf{FG}, \langle \mathbf{G}^M, L^M, l_M \rangle, \langle \mathbf{G}^N, L^N, l_N \rangle$)
2: **if** P_{l_N,l_M} is *satisfactory* **then**
3: **return** P_{l_N,l_M} ▷ Return root as optimal node
4: $LB \leftarrow -\infty$ ▷ Initialize LowerBound
5: $\mathit{Visits}_{(l_N+1)\times(l_M+1)} \leftarrow 0$ ▷ Initialize nodes' visits
6: $Q \leftarrow \Phi$ ▷ Initialize Priority Q
7: enQ(P_{l_N,l_M}) ▷ Enqueue root node
8: **while** $Q \neq \Phi$ **do**
9: *Parent* $\leftarrow$ deQ()
10: **for all** $\mathit{Child}_{i,j} \in$ ChildOf(*Parent*) **do**
11: Compute($\widetilde{PS}_{i,j,\mathit{MainDir}}$) ▷ see (4.13)
12: **if** $\widetilde{PS}_{i,j,\mathit{MainDir}} \leq LB$ **then continue** ▷ 1-Prune child
13: Compute($BG_{i,j,*}$) ▷ see Sec. 4.2.2.1
14: Compute($IG_{i,j,*}$) ▷ see Sec. 4.2.2.4
15: **if** $\mathit{Child}_{i,j}$ is ¬*backward satisfactory* **then**
16: **continue** ▷ 2-Prune child
17: **if** $\mathit{Child}_{i,j}$ is *satisfactory* $\wedge\ \widetilde{PS}_{i,j,\mathit{MainDir}} > LB$ **then**
18: $\mathit{Visits}_{i,j} \leftarrow 0$ ▷ Ignore previous visits
19: **if** $\mathit{Visits}_{i,j} > 0$ **then**
20: *Discard* $\leftarrow$ *False*
21: **for all** $x \in [1, \mathit{Visits}_{i,j}]$ **do**
22: Dominance($\mathit{Child}_{i,j,*}, \mathit{Prev}_{i,j,*}(x)$) ▷ see Figure 4.3
23: **if** ChildOf($\mathit{Child}_{i,j}$) is Φ **then** *Discard* $\leftarrow$ *True*; **break**
24: **if** *Discard* **then continue** ▷ 3-Prune Child
25: **if** $\mathit{Child}_{i,j}$ is *satisfactory* $\wedge\ \widetilde{PS}_{i,j,\mathit{MainDir}} > LB$ **then**
26: $LB \leftarrow \widetilde{PS}_{i,j,\mathit{MainDir}}$ ▷ Update the lower bound
27: *OptNode* $\leftarrow \mathit{Child}_{i,j}$ ▷ Store optimal node
28: **else**
29: enQ($\mathit{Child}_{i,j}$) ▷ Enqueue this node
30: $\mathit{Visits}_{i,j} \leftarrow \mathit{Visits}_{i,j} + 1$ ▷ Increment visits counter
31: $\mathit{Prev}_{i,j}(\mathit{Visits}_{i,j}) \leftarrow \mathit{Child}_{i,j}$ ▷ Store visit
32: **return** *OptNode* ▷ Return maximum scoring *satisfactory node*

Fig. 4.2: Branch and bound algorithm to compute optimal node.

4.2.2.3 B&B Bounding Conditions

First, we discard any *backward unsatisfactory* node. Second, if $P_{i,j}$ is a *satisfactory node* and $\widetilde{PS}_{i,j,MainDir} > LB$ (the lower bound, initially $LB = -\infty$), then we store node $P_{i,j}$ as the best node so far, set $LB = \widetilde{PS}_{i,j,MainDir}$, and discard nodes having $\widetilde{PS}_{i,j,MainDir} \leq LB$.

4.2.2.4 Dominance Relation

We first define $IG_{i,j,k}$ as $P_{i,j}$'s *immediate gaps*. That is the number of *backward gaps* in the same row or the same column to that node in the current path, where $k \in \{1, 2, 3\}$. For example, in the path shown in Figure 4.1, $IG_{11,8,*} = \langle 0, 0, 2 \rangle$ means two gaps were opened *immediately leading* to this node in the *up* direction ($k = 3$). Note that by definition, it is always true that $IG_{11,8,1} = 0$, which is the number of immediate gaps for the *diagonal* direction ($k = 1$).

We use the dominance relation in our B&B algorithm, Figure 4.2 line 22, to reduce the number of expansions of revisited nodes to greatly improve the efficiency of our $A*$ search. Since we are traversing a DAG, some nodes are reachable from the root node by multiple paths. If this happens, we compare the actual score $AS_{i,j}$ and the *immediate gaps* $IG_{i,j,k}$ of the current $Child_{i,j}$ node with its previous encounters $Prev_{i,j}(x)$ and determine the dominant node for each of its possible directions, as shown in Figure 4.3. For example, in Figure 4.1, node $P_{13,8} = 13$ implies that it has *diagonal* and *up* possible directions. This node can be reached by multiple paths from the root node. Therefore, on the second visit, $Child_{13,8}$ will be compared with $Prev_{13,8}(1)$. If $Child_{13,8}$ dominates in the *up* direction but not in the *diagonal* direction, then we prune $Prev_{13,8}(1)$'s *up* direction to

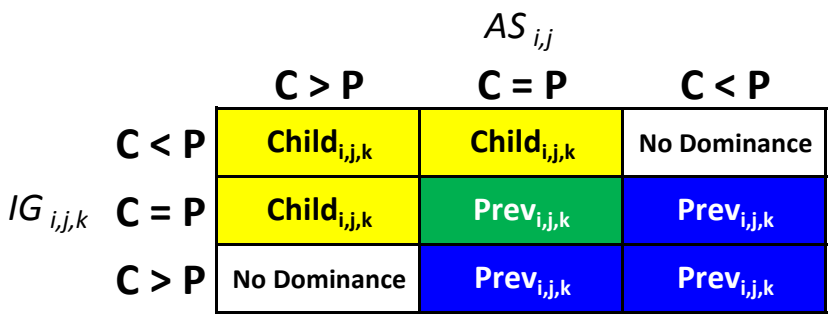


Fig. 4.3: **C***hild*$_{i,j,k}$ vs. **P***rev*$_{i,j,k}$ dominance relation for each node direction $k \in \{1, 2, 3\}$.

prevent its examination. If we prune the main direction of a node, then the next direction, based on the node's direction priority, becomes the main direction. Then we recalculate $FG_{i,j,*}$ and check if $\widetilde{PS}_{i,j,NewMainDir} \leq LB$ for possible pruning. Finally, $Child_{i,j}$ is discarded if completely dominated (Figure 4.2 line 24), also, $Prev_{i,j}(x)$ is discarded when completely dominated.

4.2.3 Stage3: Optimal Alignment Recovery

After finding the maximum scoring *satisfactory node* $P_{i,j}$ using our B&B search, we trace through the ancestors from $P_{i,j}$ to the root node adjusting their main directions to point towards the path to $P_{i,j}$. Then we backtrack along the whole path from the root node to the leaf node following the new adjusted main path, i.e., the marked path in Figure 4.1. This path uncovers the optimal alignment that satisfies all four constraints.

While following the optimal path, we gather some extra information about the path's gaps for later recovery of the new gap ranges and any constraint on the final length of the consensus structured motif and the combined frequency matrix. For instance, given the pair of sequences `AGCAT[2,5]TC[0,7]G[2,5]GCC` and `AGC[17,20]CC`, SMAlign aligns them and recovers the new gap ranges as follows:

```
    M: AGCAT-----TC-G-----GCC
       |||                 ||
    N: AGC-----------------CC
 Cons: AGCAT-----TC-G-----GCC
```

The resulting consensus structured motif with its ranges is `AGCAT[5,5]TC[1,4]G[5,5]GCC` without constraint on the maximum length. Alternatively, if we change N to `AGC[25,28]CC`, the alignment becomes:

```
    M: AGCAT-----TCG--GCC------------
       |||
    N: AGC-------------------------CC
 Cons: AGCAT-----TCG--GCC----------CC
```

As a result, the new consensus structured motif becomes `AGCAT[5,5]TC[0,3]G[2,5]GCC[10,13]CC` having constraint on the maximum length not to exceed 33bp.

SMAlign outputs the alignment score and the 4-tuple $\mathcal{R} = \langle \mathbf{R}, \mathbf{G}^R, L^R, C^R \rangle$ where $\mathbf{R}$ is the resulted frequency matrix, in *compact* form, given by default as:

$$R_{*,i} = \frac{C^M M_{*,i} + C^N N_{*,i}}{C^M + C^N} \tag{4.14}$$

$\mathbf{G}^R$ is the new gap constraints matrix, L^R is the maximum length constraint on $\mathbf{R}$ (if none, $L^R = \infty$) and $C^R = C^M + C^N$ for the default $\mathbf{R}$. Two other variations are provided by SMAlign to ignore partial gaps in $\mathbf{R}$ depending on application.

4.3 Experimental Results

The purpose of this section is to evaluate the time and space complexities of SMAlign, and compare against *NWGlobal* and the *Extended SWLocal*. We focused on simulated data for controlled test cases; real datasets are considered in Chapter 6 which discusses the use of SMAlign in a multi-alignment context to solve the structured motif extraction problem in co-regulated genes.

We implemented SMAlign in MATLAB and executed the following experiments using simulated DNA sequences on a Windows7 64-bit machine with Intel Core i5-430UM processor having 1.7GHz max speed with 4GB of RAM by giving the following command:

$$\textit{SMAlign}\,(\mathcal{M}, \mathcal{N}, \mathbf{U}, \lambda = 1, \mu = 0.5)\,.$$

Table 4.4 summarizes the experimental parameters and total run time. SMAlign's experimentally observed time and space complexities are of order $O(l_M l_N)$ and $O(b l_M l_N)$ where $b = \max(b_M, b_N)$. The time and space scale quadratically in proportion to the length of the expanded sequences, which depend on the minimum gap sizes, when varying the number of boxes and the box sizes (Figure 4.4 and 4.5), but they are less affected by the size of the gap ranges (Figure 4.6). Actually, the larger the gap range size, the less likely constraint 3 will be violated and SMAlign will more likely find the optimal alignment sooner.

SMAlign's optimality and efficiency results are presented in Table 4.5, tested using 200 pairwise

Table 4.4: SMAlign's experimental parameters and total run time.

Fig. No.	Varying Parameter	Range	Data Points	Alignments/ Data Point	Generated DNA Sequences[§] Boxes	BoxSize	GapMin	GapMax	Average Length	Total Alignments	Run Time	Remarks
4.4	Boxes	[1,10]	10	168	-	[8,10]	[0,5]	[4,7]	61bp	1681	34min	Aligns up to 10-box length sequences (max 145bp).
4.5	BoxSize	[5,23]	10	157	4	-	[0,5]	[4,7]	67bp	1574	13min	Aligns 4-box length sequences (max 107bp) for which exhaustive search estimated to take 6.5+ days!
4.6	GapRangeSize	[4,13]	10	104	4	[8,10]	[0,2]	[3,15]	39bp	1041	2min	Aligns 4-box length sequences (max 46bp).

[§]Random sequences generated using weights $w = \langle 15, 10, 8, 12, 1, 1, 3, 1, 2, 1, 0, 0, 0, 0, 0 \rangle$ corresponding to nucleotides in Σ_{IUPAC}.

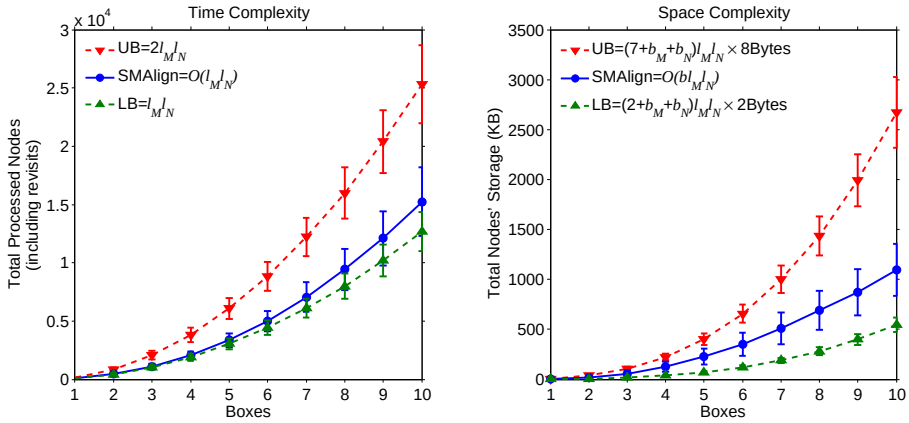


Fig. 4.4: Varying the number of boxes.

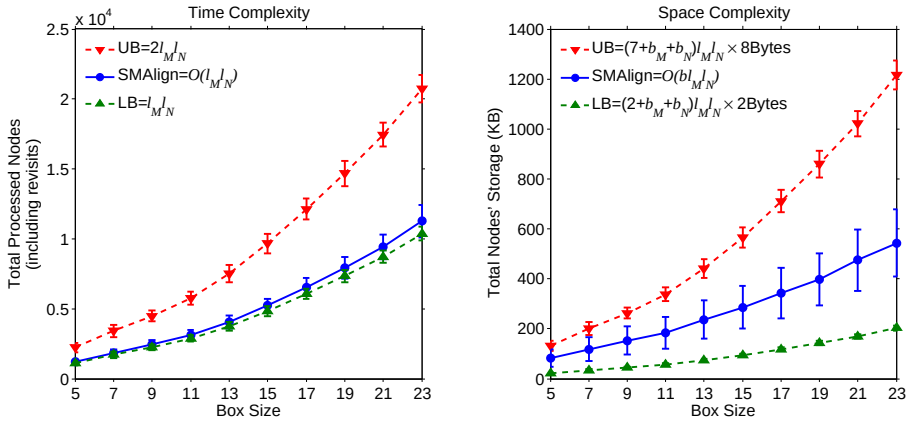


Fig. 4.5: Varying the size of boxes.

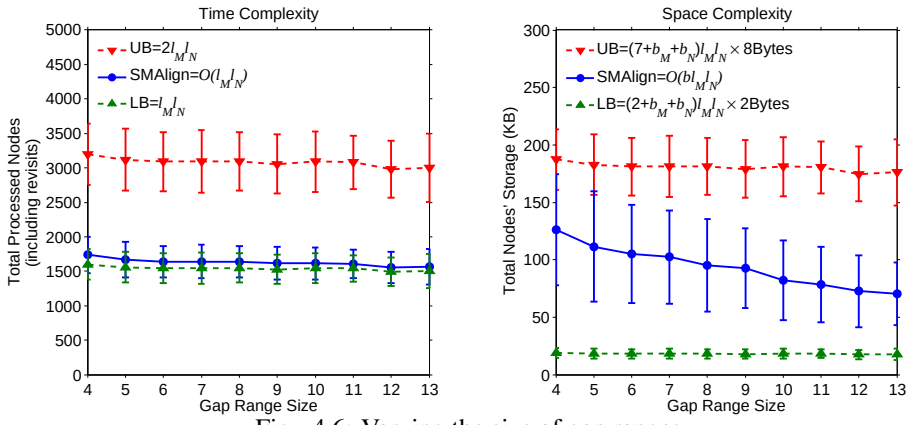


Fig. 4.6: Varying the size of gap ranges.

Table 4.5: SMAlign's optimality and efficiency (200 pairwise alignments).

Algorithms	**Constraint Satisfaction (% of 200)**	**Optimal Alignment (% of 200)**	**Extra Processed Nodes**[1] **(% of $l_M l_N$)** median	mean	std
A-Exhaustive Search	100	100	3.1x10^5	5.0x10^5	7.0x10^5
B-*SMAlign*: w/ Dominance					
B1-*A** ($\widetilde{PS}, X$)	100	100	4	6	4
B2-*A** ($\widetilde{PS}$)	100	100	5	6	4
B3-*A** (PS)	100	100	5	6	5
B4-Depth First Search[2]	100	100	6	9	9
B5-Greedy (S)	100	100	11	16	14
B6-Greedy (AS)	100	100	15	19	17
B7-Breadth First Search	100	100	15	21	19
C-*SMAlign*: w/o Dominance					
C1-*A** ($\widetilde{PS}, X$)	100	100	6	9	11
C2-*A** ($\widetilde{PS}$)	100	100	6	9	11
C3-*A** (PS)	100	100	7	10	12
C4-Depth First Search[2]	100	100	9	16	20
C5-Greedy (S)	100	100	17	40	57
C6-Greedy (AS)	100	100	21	35	46
C7-Breadth First Search	100	100	25	65	108
D-NWGlobal:					
D1-GapPenalty: open=ext=∞	100	2	0	0	0
D2-GapPenalty: open=8,ext=1	44	3	0	0	0
E-Extended SWLocal:					
E1-GapPenalty: open=ext=∞	100	2	0	0	0
E2-GapPenalty: open=8,ext=1	48	3	0	0	0

[1] Extra Processed Nodes (*EPN*) = Total Processed Nodes $- l_M l_N$. *SMAlign*'s statistics were calculated using alignments having *EPN* > 0 (87% of the 200).

[2] Children expanded based on their direction priority in **P** (local best).

Table 4.6: Statistical significance (*p-values*) of using algorithm *B1*.

Algorithms	B2	B3	C1	C2	C3
B1	9.4x10^{-4}	1.3x10^{-16}	2.1x10^{-22}	2.4x10^{-24}	1.7x10^{-24}

alignments of 400 randomly generated structured motifs having parameters as used for Figure 4.5. We tested SMAlign using various *A** heuristics and other search strategies (with and without dominance). We also tested *NWGlobal* and *Extended SWLocal*, to which inputs were given in *expanded* form. Since all tested algorithms share the same DP stage ($l_M l_N$ processed nodes), we are only interested in counting the *Extra Processed Nodes (EPN)* after that stage. Out of the 200 alignments, 173 (87%) have *EPN* > 0. The results show that our presented algorithm (*B1*) is the best performing. Its optimality for this data was supported by executing an exhaustive search algorithm, whose results were obtained by performing computations for approximately 19.5 hours versus *B1*'s 37 seconds!

We also examined the statistical significance of *B1*'s results, compared to the others, using

the non-parametric paired signed-rank hypothesis test since *EPN*, as a random variable, followed a gamma distribution. The p-values in Table 4.6 (*all* < 0.01) suggest a statistically significant improvement when using *B1*.

4.4 Extending SMAlign: Discrete Gap Constraints

In the previous sections, we demonstrated SMAlign's ability to solve the structured motif pairwise alignment when the gap constraints are represented by continuous integer ranges of possible gap sizes. In this section, we extend SMAlign to solve the discrete gap constraints case, i.e., a gap constraint is a set of discontinuous gap sizes. To handle this case, we first need to extend our structured motif definition, given in Section 1.2, for the discrete case, then use the new definition to extend SMAlign. Recall that we defined the structured motif M as:

$$M = m_1 G_1 m_2 G_2 m_3 \cdots m_{b_M-1} G_{b_M-1} m_{b_M},$$

where b_M is the number of boxes, $m_i \in (\Sigma_{\text{IUPAC}})^{|m_i|}$, and G_i is the gap constraint between m_i and m_{i+1} for $i = 1 \ldots b_M - 1$. We extend the definition by allowing the gap constraint G_i to be either a continuous integer sequence, i.e., a range of possible gap sizes $G_i = [g_i^{\min}, g_i^{\max}]$, where $0 \leq g_i^{\min} \leq g_i^{\max}$; or a discontinuous integer sequence, i.e., a set of possible gap sizes $G_i = \{x_y : \text{where } x_y \text{ are prespecified non-negative integers}\}$.

Example: Suppose we want to align the following two structured motifs:

$$M \texttt{ = GGCWGAT[1,2]GGYC\{2,5,6\}ARTG}$$

$$N \texttt{ = GAGTBCA[5,7]TCTRC[0,2]CACT}$$

Notice that the second gap location in M is a discontinuous sequence of possible gap sizes (henceforth, we call it a *discrete gap constraint*) while the rest of the gap locations are continuous gap ranges. This is an interesting problem as we are interested in the optimal alignment that maximizes the alignment score while satisfying all gap constraints, which does not allow for gap sizes of 3bp

and 4bp in the discrete gap location of this example. Solving this case by exhaustive search yields the following optimal alignment:

```
                                *
M:  ---GGCWGAT-GGYC----ARTG
      X:||      ||     |X|
N:  GAGTSCA------TCTRC-CACT-
Cons: GAGKGCAGAT-GGTCTRC-CAMTG
```

The score for this alignment is 17.67 based on the scoring matrix **U** (Table 4.1). Suppose that we substitute the discrete gap by the continuous range $[2, 6]$, and execute *SMAlign*$(M, N, \mathbf{U}, \lambda = 0)$ (we are not using the affine bonus for now, $\lambda = 0$). Then the optimal alignment would be:

```
M:  ---GGCWGAT-GGYC----ARTG
      X:||      ||     |X|
N:  GAGTSCA------TCTRCCACT-
Cons: GAGKGCAGAT-GGTCTRCCAMTG
```

SMAlign reported the same score, in this case, yet the reported optimal alignment has the size of the second gap in $\mathcal{M}$ as 4bp. Therefore, when we present discrete gap constraints to SMAlign, it should reject any alignment that does not satisfy those constraints, and continue its search until a satisfactory alignment, having maximum score, is found.

4.4.1 Satisfactory Node Adjustment

Recall that an optimal alignment reported by SMAlign is the equivalent of finding the maximum scoring *satisfactory node* in the directed acyclic graph represented by the pointers matrix **P** (Section 4.2.2.1). Inequality (4.9) is the one responsible for checking that the current node is a satisfactory node, by making sure that opened gaps do not exceed the maximum gap sizes. Therefore, we need to adjust this inequality for discrete gaps such that the number of opened gaps are in the given domain.

Suppose *GDomain*$_i$ is the domain for the i^{th} gap location, where $i \in [1, k_M + k_N - 2]$ spanning all gap locations in M and N respectively. For instance, in our example above, *GDomain*$_1 = [1, 2]$, *GDomain*$_2 = \{2, 5, 6\}$, *GDomain*$_3 = [5, 7]$ and *GDomain*$_4 = [0, 2]$. Also, suppose that $BG^*_{i,j,k}$ is the backward gaps at node $P_{i,j}$, as defined in Section 4.2.2.1, but substituting the backward gaps in

the same row or column to $P_{i,j}$ with zero to avoid unfinished gaps (hence the asterisk '*'). Then the adjusted inequality (4.9) becomes:

$$BG^*_{i,j,k} \in \{x - y : \forall x \in \textit{GDomain}_k \textit{ and } y = \min(\textit{GDomain}_k)\} \tag{4.15}$$

that should hold true for all $k \in [1, b_M + b_N - 2]$ to consider node $P_{i,j}$ as *backward satisfactory*. Also, if (4.15) holds true when substituting BG^* with the total gaps $TG = FG + BG$, then node $P_{i,j}$ is a *satisfactory node* in the discrete case. Note that we subtracted the minimum gap size (y in (4.15)) because by definition the backward gaps are the number of gaps opened on top of the minimum gap.

4.4.2 Dominance Relation Adjustment

Recall in Section 4.2.2.4 we proposed a dominance relation (Figure 4.3) to minimize node revisits (i.e., improve SMAlign's efficiency) as we traverse the directed acyclic graph to find the optimal path. In the continuous gap constraints case, it was sufficient to check if the path leading to node $\textit{Child}_{i,j}$ has lower number of immediate gaps ($IG_{i,j,k}$ for all directions $k \in \{1, 2, 3\}$) opened than the path leading to the same node's previous visit (node $\textit{Prev}_{i,j}$). However, for the discrete case, this does not necessarily hold true as we need to check that the number of opened gaps are conforming to the prespecified discrete domain of possible gap sizes in that location.

We propose a new dominance relation for the discrete case as in Figure 4.7. Suppose $FIG_{i,j,k}$ is the *full immediate gaps* at node $P_{i,j}$; that is the number of *forward* and *backward* gaps in the same row or column to that node (extending our definition of *immediate gaps* in Section 4.2.2.4). We define $f(X)$ as a boolean function that evaluates to one when the number of *full immediate gaps* (*FIG*) of node X is in the discrete domain of possible gap sizes *GDomain* as follows:

$$f(X_{i,j,k}) = \begin{cases} 1 & \text{if } FIG_{i,j,k} \in \textit{GDomain} \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

$FIG_{i,j,k}$				$AS_{i,j}$: C > P	C = P	C < P
	f(C)	&	~ f(P)	$Child_{i,j,k}$	$Child_{i,j,k}$	No Dominance
	f(C)	xnor	f(P)	No Dominance	No Dominance	No Dominance
~	f(C)	&	f(P)	No Dominance	$Prev_{i,j,k}$	$Prev_{i,j,k}$

Fig. 4.7: Discrete gap constraints case dominance relation for $\mathbf{C}hild_{i,j,k}$ vs. $\mathbf{P}rev_{i,j,k}$ for each node direction $k \in \{1, 2, 3\}$. $f(X)$ is a boolean function that evaluates to 1 when the *full immediate gaps* of node X is in the discrete domain of possible gap sizes *GDomain*, where $X \in \{\mathbf{C}hild, \mathbf{P}rev\}$.

Notice that in the discrete case dominance relation, when nodes *Child* and *Prev* are both satisfactory at the same time, but having different values, we cannot decide on which node dominates the other regardless of their actual path scores (AS, Equation (4.8)). That is because one satisfactory value may be better suited for the rest of the alignment than another value, which cannot be decided at that moment.

Using the adjusted satisfactory node definition, Equation (4.15), and the new dominance relation, SMAlign would be able to solve the above stated example problem for the discrete case alignment. However, we need to handle one last issue in the next subsection to complete our solution.

4.4.3 Simultaneous Gap Opening Problem

In order to complete our solution to the discrete gap constraints case, we need to handle what we call the *simultaneous gap opening* problem. Suppose we want to align the following pair of structured motifs:

M = GGCWGAT[1,2]GGYC{2,8}ARTG

N = GAGTBCA[5,7]TCT{0,2}R{0,2}C{0,2}CACT

The optimal alignment would be:

```
                          ** **
M:  ---GGCWGAT-GGYC--------ARTG
      X:||       ||         |X|
N:  GAGTSCA------TCTR--C--CACT-
Cons: GAGKGCAGAT-GGTCTR--C--CAMTG
```

The '*' marked columns in the optimal alignment are *simultaneous gaps* that need to be opened in both input structured motifs to satisfy the discrete gap constraints and maximize the alignment

score. This type of gaps are not possible to open in the backtracking stage (Section 4.2.3) because recall that we are only allowed to move left (gap in N), move up (gap in M), or move diagonal (match together the corresponding positions in M and N). Thus, there is no possible move in the backtracking stage to add such *simultaneous gaps*.

In order to solve this issue, let's examine SMAlign's solution to this alignment when we substitute the discrete gap constraints with their corresponding continuous gap constraints, i.e., solve the following easier problem:

M = GGCWGAT[1,2]GGYC[2,8]ARTG

N = GAGTBCA[5,7]TCT[0,2]R[0,2]C[0,2]CACT

SMAlign's solution would be:

```
   M: ---GGCWGAT-GGYC----ARTG
         X:||       ||     |X|
   N: GAGTSCA------TCTRCCACT-
Cons: GAGKGCAGAT-GGTCTRCCAMTG
```

Notice that this solution resembles the desired one for the discrete case and it is only missing the simultaneous gaps. Therefore, we can transform this solution to the desired one in two steps. First, notice that the second gap location in M (4-gaps) is overlapping with the 2^{nd} (0-gaps), 3^{rd} (0-gaps), and 4^{th} (0-gaps) gap locations of N. Second, we need to add 4 more gaps to the second gap location of M to satisfy its discrete gap constraint {2,8}, but at the same time, we need to add corresponding 4 gaps in the overlapped gap locations in N to maintain the same alignment score. Thus, the question is how to distribute the 4 gaps we need to add among the three gap locations of N? This question can be transformed into a constraint satisfaction problem called the *Upper-Restricted Weak k-Compositions of Integer n*, where k is the number of overlapping gap locations in N, and n is the total number of gaps that we need to add. This problem is briefly introduced in the next subsection.

4.4.4 The Restricted Weak k-Compositions of Integer n

The basic problem of the k-compositions of an integer n is stated as follows. Given an integer n, we would like to find the set of all k-tuples, $\langle x_1, x_2, \ldots, x_k \rangle$, such that:

$$n = \sum_{i=1}^{k} x_i, \tag{4.17}$$

where $1 \leq x_i \leq n$ and x_i is an integer [33]. For example, when $n = 5$ and $k = 3$, the following set of 3-tuples is the set of all possible solutions to the problem:

$$\begin{gathered}\langle 1, 1, 3 \rangle \\ \langle 1, 2, 2 \rangle \\ \langle 1, 3, 1 \rangle \\ \langle 2, 1, 2 \rangle \\ \langle 2, 2, 1 \rangle \\ \langle 3, 1, 1 \rangle\end{gathered}$$

The *weak variant* of the stated problem allows the values of x_i to be zero, i.e., $0 \leq x_i \leq n$. Thus, the *weak variant* solution to the above example problem ($n = 5$ and $k = 3$) is the following set of 3-tuples:

$$\begin{gathered}\langle 0, 0, 5 \rangle \\ \langle 0, 1, 4 \rangle \\ \langle 0, 2, 3 \rangle \\ \langle 0, 3, 2 \rangle \\ \langle 0, 4, 1 \rangle \\ \langle 0, 5, 0 \rangle \\ \langle 1, 0, 4 \rangle \\ \langle 1, 1, 3 \rangle \\ \langle 1, 2, 2 \rangle \\ \langle 1, 3, 1 \rangle \\ \langle 1, 4, 0 \rangle \\ \langle 2, 0, 3 \rangle \\ \langle 2, 1, 2 \rangle \\ \langle 2, 2, 1 \rangle \\ \langle 2, 3, 0 \rangle \\ \langle 3, 0, 2 \rangle \\ \langle 3, 1, 1 \rangle \\ \langle 3, 2, 0 \rangle \\ \langle 4, 0, 1 \rangle \\ \langle 4, 1, 0 \rangle \\ \langle 5, 0, 0 \rangle\end{gathered}$$

This *weak variant* of the problem has been solved efficiently, without redundant enumeration, in 1968 by Fenichel [34]. Another variant is the *restricted bounds variant*, where $1 \leq l \leq x_i \leq u \leq n$, i.e., x_i can be lower and/or upper bounded by l and u respectively. This *restricted bounds variant* has been efficiently solved recently in 2010 by Opdyke [35].

We are interested in the combination of the *weak variant* and the *restricted bounds variant* to solve our original problem of *simultaneous gaps*. In other words, we would like to solve the problem such that $0 \leq l \leq x_i \leq u \leq n$. Therefore, we have adopted Opdyke solution with a simple, yet non-trivial, modification to obtain our desired solution. Suppose $\delta(n, k, l, u)$ is Opdyke's function that finds the set of all k-tuples to solve the restricted bounds k-compositions of integer n. Then we define the *weak variant* of such restricted bounds solution as follows:

$$\delta_{weak}(n, k, l, u) = \delta(n + k, k, l + 1, u + 1) - 1. \tag{4.18}$$

For example, $\delta_{weak}(n = 5, k = 3, l = 0, u = 3) = \delta(n = 5 + 3, k = 3, l = 0 + 1, u = 3 + 1) - 1$. The solution set for Opdyke's function $\delta(n = 5 + 3, k = 3, l = 0 + 1, u = 3 + 1)$ is:

$$\begin{gathered}
\langle 1, 3, 4 \rangle \\
\langle 1, 4, 3 \rangle \\
\langle 2, 2, 4 \rangle \\
\langle 2, 3, 3 \rangle \\
\langle 2, 4, 2 \rangle \\
\langle 3, 1, 4 \rangle \\
\langle 3, 2, 3 \rangle \\
\langle 3, 3, 2 \rangle \\
\langle 3, 4, 1 \rangle \\
\langle 4, 1, 3 \rangle \\
\langle 4, 2, 2 \rangle \\
\langle 4, 3, 1 \rangle
\end{gathered}$$

Subtracting *one* from all entries yield the final solution of the *weak variant upper bounded* $k = 3$ compositions of integer $n = 5$, upper bounded by $u = 3$, and lower bounded by $l = 0$ hence solving the *weak variant*). Therefore, the final solution set that we obtain is:

$\langle 0, 2, 3 \rangle$
$\langle 0, 3, 2 \rangle$
$\langle 1, 1, 3 \rangle$
$\langle 1, 2, 2 \rangle$
$\langle 1, 3, 1 \rangle$
$\langle 2, 0, 3 \rangle$
$\langle 2, 1, 2 \rangle$
$\langle 2, 2, 1 \rangle$
$\langle 2, 3, 0 \rangle$
$\langle 3, 0, 2 \rangle$
$\langle 3, 1, 1 \rangle$
$\langle 3, 2, 0 \rangle$

Now we can use Equation (4.18) to find the solution to our original question of how to distribute the total number of gaps needed among the overlapping gap locations. This is discussed in the next subsection.

4.4.5 The Extended SMAlign

We now revisit our stated example problem to concretely put together the full solution to the discrete gap constraints alignment problem. Recall that we would like to align the following pair of structured motifs:

M = GGCWGAT[1,2]GGYC{2,8}ARTG

N = GAGTBCA[5,7]TCT{0,2}R{0,2}C{0,2}CACT

SMAlign would normally build its *Scores* and *Pointers* matrices in the first stage. Then, in the second stage, while searching for the maximum scoring satisfactory node, $P_{i,j,k}$, SMAlign checks the following for each of the node's directions ($k \in \{1, 2, 3\}$):

1. If the current node being examined is part of a continuous gap constraint location, then:
   (a) Check if it is a satisfactory node as per Section 4.2.2.1. If it is backward unsatisfactory, then prune that node.
   (b) Check its dominance against previous visits to the same node as per Section 4.2.2.4.
2. If the current node being examined is part of a discrete gap constraint location, then:

(a) Check if it is a satisfactory node as per Section 4.4.1. If it is backward unsatisfactory, then:

   i. Check if the offending backward gap location has overlapping gap locations in the other sequence.

   ii. If there is no overlapping gap locations, then prune that node.

   iii. If there is k overlapping gap locations, then solve a *simultaneous gap opening* problem, as per Section 4.4.4, using Equation (4.18). Then pick the k-tuple, from the resulted set, that satisfies the k gap locations domains.

(b) Check its dominance against previous visits to the same node as per Section 4.4.2.

The above steps complete our extension of SMAlign to handle the discrete gap constraints case. For the above stated example problem, recall that we needed to open 4 gaps in the second gap location of M to satisfy the discrete gap constraints, and we had three overlapping gap locations in N. Therefore, we set $k = 3$ and $n = 4$ and set our upper limit $u = max(0, 2)$, i.e., the maximum possible gap size in all overlapping gap locations. Then, using Equation (4.18), $\delta_{weak}(n = 4, k = 3, l = 0, u = 3)$, we efficiently obtain the following set of possible gap distributions among the overlapping gap locations:

$$\begin{gathered}\langle 0,1,3\rangle \\ \langle 0,2,2\rangle \\ \langle 0,3,1\rangle \\ \langle 1,0,3\rangle \\ \langle 1,1,2\rangle \\ \langle 1,2,1\rangle \\ \langle 1,3,0\rangle \\ \langle 2,0,2\rangle \\ \langle 2,1,1\rangle \\ \langle 2,2,0\rangle \\ \langle 3,0,1\rangle \\ \langle 3,1,0\rangle\end{gathered}$$

Then, we pick the 3-tuple that satisfies the three gap locations' discrete domains, i.e., $\langle 0, 2, 2\rangle$, to obtain SMAlign's final solution:

```
                                  ** **
M:  ---GGCWGAT-GGYC--------ARTG
       X:||       ||         |X|
N:  GAGTSCA------TCTR--C--CACT-
Cons: GAGKGCAGAT-GGTCTR--C--CAMTG
```

Figure 4.8 shows a visualization of SMAlign's stages solving the discrete gap constraints example stated above. Notice in the alignment recovery stage (Stage 3), the marked lines in the matrix represent added *simultaneous gaps* to satisfy the discrete gap constraint.

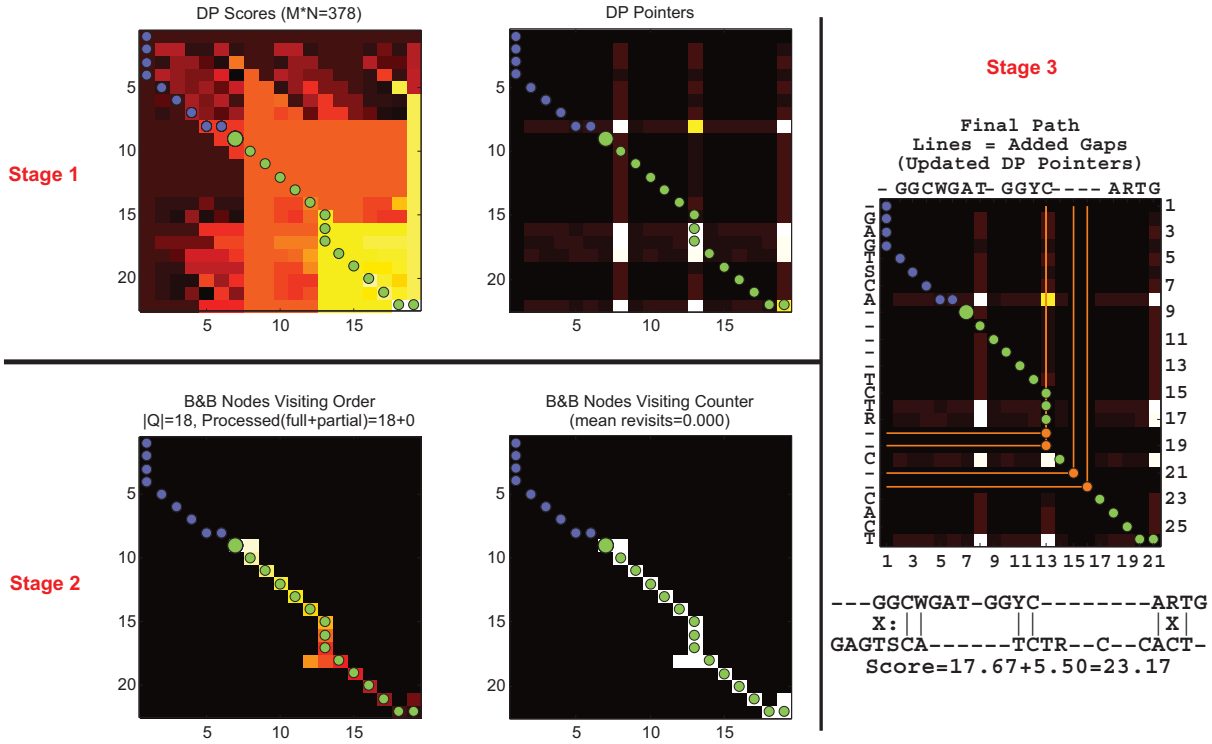


Fig. 4.8: Example alignment problem solved by SMAlign correctly handling the discrete gap constraints. *Stage 1*) SMAlign builds its *Scores* and *Pointers* matrices. *Stage 2*) Branch and bound search for the maximum scoring satisfactory node (marked as big green circle). *Stage 3*) Recovery of the optimal alignment showing the added *simultaneous gaps* (orange lines) to satisfy the discrete gap constraint.

4.5 Discussion

In this chapter, we presented SMAlign as an efficient algorithm that optimally aligns a pair of DNA sequences with gap constraints (structured motifs), using dynamic programming and a branch and bound search algorithm. It has the ability to bias the alignment towards matching localized

substrings rather than segmented subsequence matches using an optional affine match bonus ($\lambda \geq 0$). When $\lambda = 0$, we consider the score as a *similarity* measure, but when $\lambda > 0$, we consider the score as an *affinity* measure that we found to be very useful for structured motif multi-alignment, discussed in the next chapter. Also, SMAlign correctly aligns even weakly conserved signals by eliminating the effect of partial gaps on the alignment algorithm. SMAlign is flexible in defining its similarity scoring matrix **U** (Table 4.1) to suite different applications.

Finally, we extended SMAlign to also handle the discrete gaps sizes case. This helps in eliminating unlikely alignments by eliminating gap sizes that are known to be impossible, which leads to a better quality alignment. A MATLAB implementation and an online version of SMAlign can be accessed from *http://bioproject.syr.edu/smtools*.

CHAPTER 5
MULTIPLE ALIGNMENT OF STRUCTURED MOTIFS

5.1 Overview

Exact multiple sequence alignment is known to be an NP-complete problem [36]. Traditionally, this has been addressed by progressive-multiple-alignments based on hierarchical pairwise alignment, also known as hierarchical clustering. Some well known multiple sequence alignment algorithms are ClustalW [37], T-COFFEE [38], M-COFFEE [39], and STAMP [30].

SMCluster is a new progressive multiple alignment algorithm that is unique in its use of SMAlign (Chapter 4) as its underlying pairwise alignment algorithm. In this chapter we introduce SMCluster and its unique features, and compare it to the aforementioned algorithms. SMCluster is an essential part of our structured motif extraction algorithm SMExtract, introduced later in Chapter 6.

5.1.1 Structured Motif Multiple Alignment

Suppose **V** (Table 5.1) is a table of scores for matches, penalties for mismatches, and cost for insertion and deletion, dynamic programming can be used to optimally align two sequences. However, when we need to align a set of sequences of size greater than three (i.e., the multiple

sequence alignment problem), the dynamic programming approach becomes computationally prohibitive. In [36], it has been shown that the multiple sequence alignment problem is NP-complete. Consequently, reasonable heuristic is the only alternative for this task. RNA, DNA, and protein sequences are assumed to have an evolutionary relationship. Therefore, an appealing heuristic is to apply a sequence of pairwise alignments, i.e., first align the most closely related pair of sequences and progressively add more and more sequences of the given set, in order of their decreasing closeness. This method, known as the progressive multiple alignment, has become the standard technique for multiple alignment.

Suppose $\mathcal{M} = \{M_i\}_{i=1}^{n}$ is a set of n structured motifs each of the form as defined in Section 4.4 (i.e., the extended definition that handles discrete gap constraints). For instance, in Figure 5.1a, structured motif no.1 has a continuous gap constraint, and structured motif no.4 has a discrete gap constraint. The goal is to find a multiple alignment of $\mathcal{M}$ that maximizes the overall alignment score, based on a scoring matrix such as **V** (Table 5.1), while satisfying all gap constraints (gap locations and sizes).[§]

Table 5.1: Default similarity scoring matrix for SMCluster

V	A	C	G	T	–
A	4	-1	-1	-1	-0.1
C	-1	4	-1	-1	-0.1
G	-1	-1	4	-1	-0.1
T	-1	-1	-1	4	-0.1
–	-0.1	-0.1	-0.1	-0.1	-0.1

Figure 5.1a shows an example set of heterogeneous structured motifs. It contains motifs that differ in terms of number of boxes, and types of gap constraints. Figure 5.1b contains the ideal multiple alignment that maximizes the sum of pairwise scores, based on **V**, while satisfying all gap constraints. In the next section we discuss current multiple alignment algorithms and show that they are not suitable for aligning sequences with known gap constraints such as structured motifs.

[§]Notice the difference in the gap symbol '$-$' penalties in **V** compared to SMAlign's default **U** matrix (Table 4.1), which has zeros. These penalties were added to bias a tie breaker towards minimal gapping.

```
 1) ACGACG[8,24]CTCCTAG
 2) GACGGC[4,20]TGCTCCTAGT[6,22]AGAGAGCC
 3) CGACGGCC{0,7,21}CTCCT[8,24]AAGAGAGCCG
 4) GCTCCTAGTG{5,10,19}AAAGAGAG
 5) CGACGGCC[4,20]GCTCCTAG
 6) GACGGCC[19,49]AAAGAGAG
 7) ACGACGG[25,55]AGAGCCGGT
 8) CGACGG[7,23]CTCCTA[6,22]AAAGAGAGC
 9) ACGGCCT[4,20]CTCCTAGTG[5,21]AGAGAGCC
10) CTAGTGG[4,18]AAAGAGAGCCGGTA
```

(a) A set of 2-box and 3-box structured motifs. Notice the discrete as well as continuous gap constraints in 3 and 4.

```
 1) ACGACG----------CTCCTAG--------------------
 2) --GACGGC------TGCTCCTAGT--------AGAGAGCC----
 3) -CGACGGCC-------CTCCT----------AAGAGAGCCG---
 4) ---------------GCTCCTAGTG-----AAAGAGAG------
 5) -CGACGGCC------GCTCCTAG---------------------
 6) --GACGGCC--------------------AAAGAGAG------
 7) ACGACGG--------------------------AGAGCCGGT-
 8) -CGACGG---------CTCCTA--------AAAGAGAGC-----
 9) ---ACGGCCT------CTCCTAGTG-------AGAGAGCC----
10) -------------------CTAGTGG----AAAGAGAGCCGGTA
```

(b) Ideal multiple alignment that maximizes the score (no column contains a mismatch, ignoring the gap symbol) while satisfying all gap constraints.

Fig. 5.1: Ideal multiple alignment of a set of structured motifs.

5.1.2 Existing Algorithms

In this section, we briefly discuss existing multiple alignment algorithms and argue that they are not suitable to solve the structured motif multiple alignment problem.

ClustalW [37] is a well known progressive multiple sequence alignment algorithm. It carries out the multiple alignment in three steps. First, ClustalW calculates a pairwise local alignment score between each pair of sequences and builds a pairwise distance matrix. Second, using the distance matrix, a guide tree is built by progressively pairing together close sequences using the neighbor joining method. Third, the branches of the guide tree are followed from the leaf nodes to the root node to build the multiple alignment, where at each step, a pair of sequences (or a pair of profile matrices for previously aligned clusters) are aligned using local alignment.

The novel approach of ClustalW is dynamically adapting the gap open penalty and the gap extension penalties (required for the alignments in the first and last steps) from their user defined initial values to more suitable values based on specific criteria. For instance, ClustalW lowers the gap penalties at positions where there already exist some gaps from previous alignment steps. Also, the gap open penalty is increased at positions close to existing gaps (8bp away by default or a user specified distance). The authors argue that these adaptive gap penalty values improve the overall multiple alignment result of ClustalW. However, ClustalW does not offer the option of specifying gap constraints. Therefore, it is not suitable to solve the structured motif multiple alignment problem. Figure 5.2b shows ClustalW's multiple alignment result for the example problem given in the previous section (restated in Figure 5.2a). Since ClustalW does not take gap

constraints as parameters, the structured motifs are reduced to sequences with as many gap symbols '$-$' as the lower bound of the gap range. For example, the 5^{th} structured motif in Figure 5.2a (`CGACGGCC[4,20]GCTCCTAG`) became the sequence `CGACGGCC----GCTCCTAG`. It can be seen that the resulting alignments violate the gap constraints and columns mismatches occur, concluding that ClustalW is not suited for this task.

```
 1) ACGACG[8,24]CTCCTAG
 2) GACGGC[4,20]TGCTCCTAGT[6,22]AGAGAGCC
 3) CGACGGCC{0,7,21}CTCCT[8,24]AAGAGAGCCG
 4) GCTCCTAGTG{5,10,19}AAAGAGAG
 5) CGACGGCC[4,20]GCTCCTAG
 6) GACGGCC[19,49]AAAGAGAG
 7) ACGACGG[25,55]AGAGCCGGT
 8) CGACGG[7,23]CTCCTA[6,22]AAAGAGAGC
 9) ACGGCCT[4,20]CTCCTAGTG[5,21]AGAGAGCC
10) CTAGTGG[4,18]AAAGAGAGCCGGTA
```

(a) A set of 2-box and 3-box structured motifs given as input to ClustalW. Each gap constraint is substituted with the minimum gap symbols before given to ClustalW (e.g., no.5 becomes `CGACGGCC----GCTCCTAG`).

```
          *****    *                    ******** **
 1) ----ACGACG--------------------------CTCCTAG--------------- *
 2) ------GACGGC----------------------TGCTCCTAGT------AGAGAGCC *
 3) -----CGACGGCCCTCCT--------------AAGAGAGCCG----------------
 4) GCTCCTAGTG--------------------------AAAGAGAG--------------- *
 5) -----CGACGGCC----------------------GCTCCTAG--------------- *
 6) ------GACGGCC----------------------AAAGAGAG---------------
 7) ----ACGACGG-------------------------AGAGCCGGT--------------
 8) -----CGACGG-------------CTCCTA------AAAGAGAGC--------------
 9) -------ACGGCCT----------------------CTCCTAGTG-----AGAGAGCC *
10) ----CTAGTGG-------------------------AAAGAGAGCCGGTA--------- *
```

(b) The resulted multiple alignment from ClustalW. Many gap constraints are violated (* marked rows) with mismatches in many columns (* marked columns).

Fig. 5.2: ClustalW multiple alignment of a set of structured motifs.

T-COFFEE [38] is another progressive multiple sequence alignment algorithm. Similar to ClustalW, T-COFFEE uses the three step process to generate its multiple alignment. Unlike ClustalW, T-COFFEE starts by performing both pairwise global alignments and local alignments for all pairs in the given input. Then a position-specific weight is calculated for each matched pair of nucleotides that is proportional to their respective full sequences' pairwise alignment score. For example, suppose A_1 is the first nucleotide of sequence A and B_1 is the first nucleotide of sequence B and they are matched in a global or a local alignment, then a weight proportional to the overall alignment score of A and B is stored in a library of weights as $w(A_1, B_1)$. This library of weights is then extended for triplets of sequences. For instance, if A_1 matches B_1, B_1 matches C_2, and A_1 matches C_2 then we add the minimum of the three weights to all of them, i.e., $w(A_1, B_1) = w(A_1, B_1) + \min(w(A_1, B_1), w(B_1, C_2), w(A_1, C_2))$ and similarly for $w(B_1, C_2)$ and $w(A_1, C_2)$. These weights are used to influence the progressive alignment in the final step. This gives the effect of using the information of all pairwise alignments in each step of the progressive multiple alignment.

Figure 5.3b shows T-COFFEE's multiple alignment result for the example problem given in Figure 5.3a. Exactly as with ClustalW, the structured motifs are represented by sequences with gaps to T-COFFEE. It is easy to observe from the results that the multiple alignment violates almost all gap constraints with mismatches in many columns. Also, notice that T-COFFEE did not maintain the minimum given gaps in structured motifs 6 and 7. This clearly shows that T-COFFEE is not the right tool to solve the structured motif multiple alignment problem.

```
 1) ACGACG[8,24]CTCCTAG
 2) GACGGC[4,20]TGCTCCTAGT[6,22]AGAGAGCC
 3) CGACGGCC{0,7,21}CTCCT[8,24]AAGAGAGCCG
 4) GCTCCTAGTG{5,10,19}AAAGAGAG
 5) CGACGGCC[4,20]GCTCCTAG
 6) GACGGCC[19,49]AAAGAGAG
 7) ACGACGG[25,55]AGAGCCGGT
 8) CGACGG[7,23]CTCCTA[6,22]AAAGAGAGC
 9) ACGGCCT[4,20]CTCCTAGTG[5,21]AGAGAGCC
10) CTAGTGG[4,18]AAAGAGAGCCGGTA
```

(a) A set of 2-box and 3-box structured motifs given as input to T-COFFEE. Each gap constraint is substituted with the minimum gap symbols before given to T-COFFEE.

```
        ********   ********* ***  *********
 1) ACGACG--------CTCCTA------G-------- *
 2) GACGGCTG----CTCCTAGT------A-GAGAGCC *
 3) CGACGGCCCTCCT--------AAGAGA-GCCG--- *
 4) GCTCCTA-----GTGAAAGA------GAG------ *
 5) CGACGGCC----GCTCCTAG---------------
 6) -------------GACGGCC------A-AAGAGAG *
 7) ACGACGG----AGAGCCGGT--------------- *
 8) CGACGG-------CTCCTAA------A-AGAGAGC *
 9) ACGGCCTC----TCCTAGTG------AGAGAGCC- *
10) CTAGTGG----AAAGAGAGCCGGTA----------
```

(b) The resulted multiple alignment from T-COFFEE. Many gap constraints are violated (* marked rows) with mismatches in many columns (* marked columns).

Fig. 5.3: T-COFFEE multiple alignment of a set of structured motifs.

M-COFFEE [39] is a meta-mode extension for T-COFFEE such that instead of using pairwise global and local alignments to build its library of weights in the first step, M-COFFEE builds the library of weights combining many multiple alignment results from different algorithms. M-COFFEE provides a total of 11 pairwise alignments and 8 multiple alignment algorithms, and the user can choose any number of combinations, the results of which are combined to build the library of weights, which in turn influences the progressive multiple alignment. The default M-COFFEE algorithm uses all 8 multiple alignment algorithms to produce its results. Figure 5.4b shows M-COFFEE's multiple alignment result for the example problem given in Figure 5.4a. As before, we presented the input structured motifs to M-COFFEE by substituting the gap constraints with the minimum gaps. Figure 5.4b shows the resulting multiple alignment, violating all gap constraints with mismatches in many columns. Like T-COFFEE, M-COFFEE does not maintain the minimum given gaps in almost all of the structured motifs, only motifs 8 (the second gap) and 10 maintained the minimum gaps. Also, many scattered gaps were opened in previously ungapped locations. This clearly shows that M-COFFEE is also not the right tool to solve this problem.

```
 1) ACGACG[8,24]CTCCTAG
 2) GACGGC[4,20]TGCTCCTAGT[6,22]AGAGAGCC
 3) CGACGGCC{0,7,21}CTCCT[8,24]AAGAGAGCCG
 4) GCTCCTAGTG{5,10,19}AAAGAGAG
 5) CGACGGCC[4,20]GCTCCTAG
 6) GACGGCC[19,49]AAAGAGAG
 7) ACGACGG[25,55]AGAGCCGGT
 8) CGACGG[7,23]CTCCTA[6,22]AAAGAGAGC
 9) ACGGCCT[4,20]CTCCTAGTG[5,21]AGAGAGCC
10) CTAGTGG[4,18]AAAGAGAGCCGGTA
```

(a) A set of 2-box and 3-box structured motifs given as input to M-COFFEE. Each gap constraint is substituted with the minimum gap symbols before given to M-COFFEE.

```
      *   **   *******    **    *    *
 1) ACG-A-----C-GCTCCTA-------------G *
 2) --G-A--CGGCTGCTCCTAGT-AGAGAGC----C *
 3) -CG-A--CGGCCCTCCTA----AGAGAGCC---G *
 4) ------------GCTCCTAGTGAAAGAGA----G *
 5) -CG-A--CGGCCGCTCCTA--------------G *
 6) --G-A--CGGCC----------AAAGAGA----G *
 7) ------ACG-------AC----GGAGAGCCGG-T *
 8) -CG-A--CGGCTCCTA------AAAGAGA-G--C *
 9) ----A--CGGCCTCTCCTAGTGAGAGAGC----C *
10) --CTAGTGG-------AA----AGAGAGCCGGTA *
```

(b) The resulted multiple alignment from M-COFFEE. All gap constraints are violated (* marked rows) with mismatches in many columns (* marked columns).

Fig. 5.4: M-COFFEE multiple alignment of a set of structured motifs.

STAMP [30] is a multiple sequence alignment algorithm that also utilizes the three steps approach of using pairwise alignment to build a distance matrix, build a guide tree and then build the multiple alignment using the tree. At each step STAMP provides the user with alternative algorithms. For instance, in the first step, a user can choose global alignment, local alignment, or a modified local alignment they call ungapped local alignment. The pairwise scoring is also based on user choice of alternative methods, like, Pearson correlation coefficient, sum of squared distances (between two profile matrix columns), Kullback-Leibler divergence, and others. In the second step, the user is provided with the choice of using UPGMA (unweighted pair group method with arithmetic mean) or SOTA (self-organizing tree algorithm, based on neural networks) to build the guide tree. In the third step, the tree is followed progressively or iteratively (using the leave-one-out method) based on user choice.

Given all the possible combination of choices, their study concludes by recommending a choice of algorithm for each step. For the first step, they recommend the ungapped local alignment with Pearson correlation coefficient scoring. For the second step, they recommend the UPGMA approach to build the guide tree. For the final step, they recommend the iterative alignment method. Figure 5.5 shows STAMP's multiple alignment result for the example problem given in Section 5.1.1 using the recommended algorithms. Although the result shows that STAMP satisfied the gap constraints, but the multiple alignment score is not maximized, evident from the number of columns with mismatches. In general, using the recommended pairwise ungapped local alignment method, STAMP will always satisfy the gap constraints as we would have already provided the minimum

allowed gaps and this method does not allow any opening of more gaps. However, maximizing the overall score of the multiple alignment is not always guaranteed.

```
 1) ACGACG[8,24]CTCCTAG
 2) GACGGC[4,20]TGCTCCTAGT[6,22]AGAGAGCC
 3) CGACGGCC{0,7,21}CTCCT[8,24]AAGAGAGCCG
 4) GCTCCTAGTG{5,10,19}AAAGAGAG
 5) CGACGGCC[4,20]GCTCCTAG
 6) GACGGCC[19,49]AAAGAGAG
 7) ACGACGG[25,55]AGAGCCGGT
 8) CGACGG[7,23]CTCCTA[6,22]AAAGAGAGC
 9) ACGGCCT[4,20]CTCCTAGTG[5,21]AGAGAGCC
10) CTAGTGG[4,18]AAAGAGAGCCGGTA
```

(a) A set of 2-box and 3-box structured motifs given as input to STAMP. Each gap constraint is substituted with the minimum gap symbols before given to STAMP.

```
          ****  **              *    ***
 1) ACGACG--------CTCCTAG--------------------
 2) --GACGGC----TGCTCCTAGT------AGAGAGCC------
 3) ------CGACGGCCCTCCT--------AAGAGAGCCG-----
 4) -------------GCTCCTAGTG-----AAAGAGAG------
 5) -CGACGGCC----GCTCCTAG--------------------
 6) --GACGGCC------------------AAAGAGAG------
 7) ACGACGG-------------------------AGAGCCGGT-
 8) -CGACGG-------CTCCTA------AAAGAGAGC-------
 9) ---ACGGCCT----CTCCTAGTG-----AGAGAGCC------
10) -----------------CTAGTGG----AAAGAGAGCCGGTA
```

(b) The resulted multiple alignment from STAMP. The gap constraints are not violated, however, some columns have mismatches (* marked).

Fig. 5.5: STAMP multiple alignment of a set of structured motifs.

None of the multiple alignment algorithms discussed above was able to solve the structured motif multiple alignment problem, evident from the given example (Figure 5.1a). This is understandable as they were not designed to solve this problem to multiply align DNA sequences with specific gap constraints (i.e., structured motifs). STAMP was the only algorithm that came close to solve this example. However, we show in this chapter that this observation is not consistent as other example problems show ClustalW gives a better result than STAMP.

The lack of algorithms that solve the structured motif multiple alignment problem motivated the development of SMCluster, discussed next. We also show that SMCluster produces a better alignment result than any of the above discussed algorithms.

5.2 SMCluster

In this section we present SMCluster, a progressive algorithm to solve the structured motif multiple alignment problem. SMCluster uses a three step process to generate its multiple alignment, similar to the one discussed above. However, instead of using global or local alignment used by the previous algorithms, SMCluster uses SMAlign (Chapter 4) as its underlying pairwise alignment algorithm.

In the next subsections, we first revisit SMAlign to briefly highlight its main features that gives SMCluster its desired behavior. We then use SMAlign's score to calculate a pairwise distance

measure between a pair of structured motifs. Then we introduce the details of SMCluster.

5.2.1 SMAlign Revisited

Multiple sequence alignment algorithms, discussed in the previous section, use either local alignment or global alignment to calculate distance measures between pairs of sequences in the input set. SMCluster, instead, uses SMAlign for its pairwise alignment. The following unique features of SMAlign give SMCluster its ability to successfully solve the structured motif multiple alignment problem:

1. SMAlign directly handles gap constraints, both continuous (Section 4.2) and discrete (Section 4.4), to efficiently find the optimal alignment of a pair of structured motifs without violating the constraints.
2. It has two alignment modes, a *similarity* mode and an *affinity* mode (Section 4.2.1.3). The affinity mode is a novel alignment mode that is essential for SMCluster and also for SMExtract discussed in Chapter 6.
3. For a pair of structured motifs M and N, it aligns N against both M and its reverse complement, and choose the better alignment. This feature is important for discovering patterns implanted on both DNA strands, discussed in detail in Chapter 6.
4. It does not penalize terminal gaps, regardless of the gap penalties in **V** (Table 5.1).

5.2.2 Distance Measure

SMCluster combines a pair of structured motifs M and N into a cluster by calling:

$$[R, \textit{affinity}] = \text{SMAlign}(M, N, \mathbf{V}, \lambda = 1, \mu = 0.5), \tag{5.1}$$

which returns both a new structured motif R, as the result of their alignment, and their *affinity(*M, N*)* score (since $\lambda > 0$, Section 4.2.1.3). SMCluster then converts the affinity score into a dissimilarity

measure:

$$\textit{dissimilarity}(M, N) = \frac{1}{\textit{affinity}(M, N)}, \tag{5.2}$$

and the new cluster, containing M and N, is represented by the profile matrix R. Our dissimilarity measure preserves the order of clustering based on the absolute pairwise affinity score, eliminating any biases related to the lengths of the aligned structured motifs. We have tested other alternative formulations for the dissimilarity measure based on mutual information (e.g, Jaccard's distance), however the results (not shown) were not as good for the purpose of solving the structured motif multiple alignment problem.

5.2.3 Progressive Multiple Alignment

Progressive multiple sequence alignment is performed using the classical hierarchical clustering algorithms, which follows two basic approaches. Both approaches require calculating a dissimilarity matrix of size n^2 for a given set of n sequences. However, they differ in their guide tree construction, which determines the order of sequence clustering. The first approach is fast but produces approximate results where the initial dissimilarity matrix is used to pre-build the entire guide tree. The dissimilarity between a pair of clusters is determined by the dissimilarity of their respective members using methods such as *complete-link*, *single-link*, or *UPGMA*, among others [40]. The second hierarchical clustering approach is slower but more accurate where the guide tree is built progressively at each clustering level. Thus the dissimilarity between a pair of clusters is the dissimilarity between their respective profile matrices. SMCluster provides the user with two variants that we call SMCluster1 (fast-approximate) and SMCluster2 (slow-accurate). Figure 5.6 shows the SMCluster algorithm (for both variants).

SMCluster produces a normalized and monotonic dendrogram whose heights are interpreted as follows. Suppose clusters Z and W are the result of combining clusters (X, Y) and (U, V),

1: Given a set of n structured motifs $\mathcal{M} = \{M_i\}_{i=1}^{n}$ each of minimum length $|M_i|_{\text{min}} = m$.
2: Initialize n clusters $\mathcal{C} = \{C_i\}_{i=1}^{n}$ each assigned one member ($C_i = M_i$).
3: Compute the $n \times n$ dissimilarity matrix **D** and the corresponding R_{ij} for each pair (C_i, C_j) using equations (5.1) and (5.2).
4: **while** (number of clusters > 1) **do**
5: Find the closest pair of clusters (C_i, C_j), among remaining, using **D**.
6: Combine (C_i, C_j) into a new cluster $C_k = R_{ij}$ ($k > n$).
7: **if** SMCluster1 **then** ▷ Fast-Approximate variant
8: Update **D** with a new row/column for C_k using the *complete-link* method.
9: Compute R_{ik} only for (C_i, C_k) where $D_{ik} = \min(D_{*k})$.
10: **else if** SMCluster2 **then** ▷ Slow-Accurate variant
11: Update **D** with a new row/column for C_k using equations (5.1) and (5.2), i.e., compute R_{ik} for all (C_i, C_k).
12: **endif**
13: **endwhile**

Fig. 5.6: SMCluster progressive multiple alignment algorithm for both variants (SMCluster1 and SMCluster2)

respectively, then the dendrogram height at which clusters Z and W are combined is:

$$\textit{height}(Z, W) = \delta \cdot \max \begin{cases} \textit{dissimilarity}(Z, W) \\ \textit{dissimilarity}(X, Y) \\ \textit{dissimilarity}(U, V) \end{cases} \tag{5.3}$$

where δ is a normalizing factor such that *height* $\in [0, 100]$. Also, for each cluster, SMCluster provides two measures to quantify the quality of the alignment within a cluster. This can serve as to guide the user to choose a suitable cut off threshold in the dendrogram to stop further clustering if these measures fall below user threshold. To define these measures, suppose R is the profile matrix representing the alignment within a cluster. The first measure is the *conservation* of R's columns:

$$\textit{conservation}_j = \max_{i \in \Sigma_{\text{DNA}}} (R_{ij}), \tag{5.4}$$

where $\Sigma_{\text{DNA}} = \{A, C, G, T\}$ and R_{ij} is the frequency of symbol i in column j (Figure 5.7c). The

second measure is *alignment quality* of R's columns:

$$\textit{quality}_j = \frac{\textit{conservation}_j}{\sum\limits_{i \in \Sigma_{\text{DNA}}} R_{ij}}. \tag{5.5}$$

This is similar to the conservation measure above except that we ignore the partial gaps within each column in R (i.e., renormalize the conservation ignoring the gap row). Note that when $\textit{conservation}_j = 0$, then $\textit{quality}_j = 0$. To calculate the overall conservation or quality of a cluster, we take the *average* measure across the columns of R, ignoring columns that are purely gaps. For example, Figure 5.7 shows the ideal alignment for the example problem presented in the previous section along with the alignment's profile matrix R. The conservation of R is 53% and the quality of R is 100%. Notice that columns {11,12,13,14} and {27,28,29,30} are purely gaps, thus ignored in the calculations.

To give some intuition for the two measures, generating the cluster logo using the *conservation* measure uncovers the parts of the result that are most conserved (frequent) in the aligned structured motifs (Figure 5.7d). On the other hand, generating the cluster logo using the *quality* measure uncovers the parts of the result that have the least mismatches in the multiple alignment (Figure 5.7e).

5.3 Time Complexity of SMCluster

We examine the time complexity of SMCluster's two variants (Figure 5.6) clustering n structured motifs each of minimum length m. First, Figure 5.6 line 3 requires $n(n+1)/2$ SMAlign operations each of order $O(m^2)$ (Section 4.3) and thus $O(n^2m^2)$. Second, line 5 requires a total of $O(n^3)$ search operations in worst case. Then, for each variant, the time complexity is as follows.

SMCluster1 (fast-approximate variant): Line 8 requires total $O(n^3)$ operations in worst case. Line 9 requires $O(n^2m^2)$ in worst case, assuming each combined structured motif increased its length by m (i.e., $m, 2m, 3m, \ldots, (n-1)m$). Therefore, the total time complexity of *SMCluster1* is

```
 1) ACGACG[8,24]CTCCTAG
 2) GACGGC[4,20]TGCTCCTAGT[6,22]AGAGAGCC
 3) CGACGGCC{0,7,21}CTCCT[8,24]AAGAGAGCCG
 4) GCTCCTAGTG{5,10,19}AAAGAGAG
 5) CGACGGCC[4,20]GCTCCTAG
 6) GACGGCC[19,49]AAAGAGAG
 7) ACGACGG[25,55]AGAGCCGGT
 8) CGACGG[7,23]CTCCTA[6,22]AAAGAGAGC
 9) ACGGCCT[4,20]CTCCTAGTG[5,21]AGAGAGCC
10) CTAGTGG[4,18]AAAGAGAGCCGGTA
```

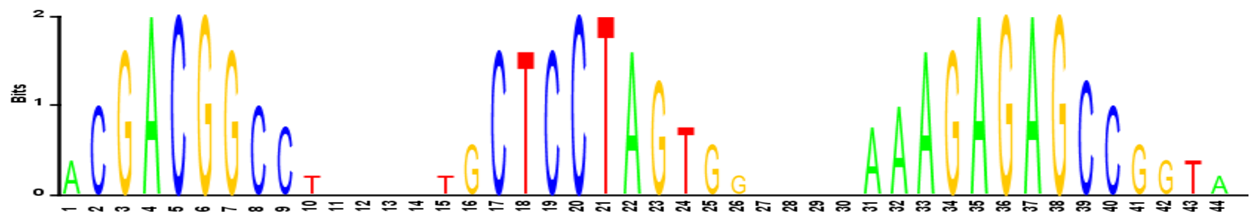
(a) A set of 2-box and 3-box structured motifs.

```
 1) ACGACG----------CTCCTAG--------------------
 2) --GACGGC------TGCTCCTAGT--------AGAGAGCC----
 3) -CGACGGCC-------CTCCT----------AAGAGAGCCG---
 4) ---------------GCTCCTAGTG-----AAAGAGAG------
 5) -CGACGGCC------GCTCCTAG--------------------
 6) --GACGGCC--------------------AAAGAGAG------
 7) ACGACGG--------------------------AGAGCCGGT-
 8) -CGACGG---------CTCCTA--------AAAGAGAGC-----
 9) ---ACGGCCT------CTCCTAGTG-------AGAGAGCC----
10) -------------------CTAGTGG----AAAGAGAGCCGGTA
```

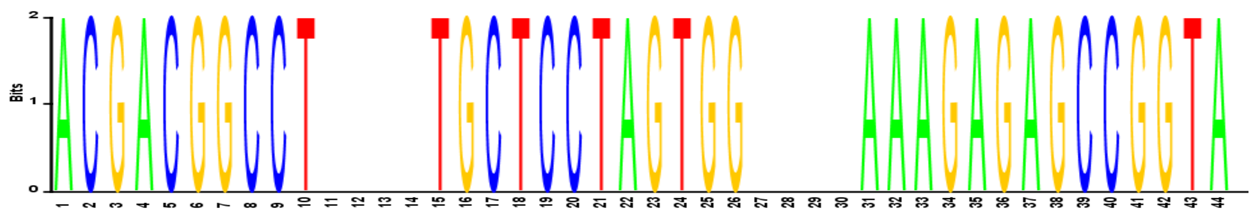
(b) The ideal multiple alignment.

A	0.2	0	0	0.8	0	0	0	0	0	0	0000	0	0	0	0	0	0	0	0.7	0	0	0	0	0000	0.4	0.5	0.7	0	0.8	0	0.8	0	0	0	0	0	0	0.1
C	0	0.5	0	0	0.8	0	0	0.5	0.4	0	0000	0	0	0.7	0	0.7	0.8	0	0	0	0	0	0	0000	0	0	0	0	0	0	0	0	0.6	0.5	0	0	0	0
G	0	0	0.7	0	0	0.8	0.7	0	0	0	0000	0	0.3	0	0	0	0	0	0	0.6	0	0.3	0.1	0000	0	0	0	0.7	0	0.8	0	0.8	0	0	0.3	0.2	0	0
T	0	0	0	0	0	0	0	0	0	0.1	0000	0.1	0	0	0.7	0	0	0.8	0	0	0.4	0	0	0000	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0
–	0.8	0.5	0.3	0.2	0.2	0.2	0.3	0.5	0.6	0.9	1111	0.9	0.7	0.3	0.3	0.3	0.2	0.2	0.3	0.4	0.6	0.7	0.9	1111	0.6	0.5	0.3	0.3	0.2	0.2	0.2	0.2	0.4	0.5	0.7	0.8	0.8	0.9

(c) The profile matrix R (frequency matrix) of the multiple alignment.



(d) The aligned sequence logo generated using the *conservation* measure.



(e) The aligned sequence logo generated using the *quality* measure. Notice that the ideal multiple alignment has no columns with mismatches.

Fig. 5.7: Example alignment showing profile matrix, and two logos representing alignment's *conservation* and *quality*.

$O(n^2m^2)+n^3+n^3+n^2m^2) = O(n^3+n^2m^2)$. If $m^2 \geq n$, then the complexity becomes $O(n^2m^2)$, otherwise $O(n^3)$.

SMCluster2 (slow-accurate variant): Line 11 requires total $O(n^3m^2)$ operations in worst case, assuming each combined structured motif increased its length by m (i.e., $m, 2m, 3m, \ldots, (n-1)m$). Therefore, the total time complexity of *SMCluster2* is $O(n^2m^2 + n^3 + n^3m^2) = O(n^3m^2)$.

5.4 Experimental Results

We implemented SMCluster in MATLAB as a command line function with a graphical user interface (Figure 5.8). We executed the following experiments on a Windows7 64-bit machine with Intel

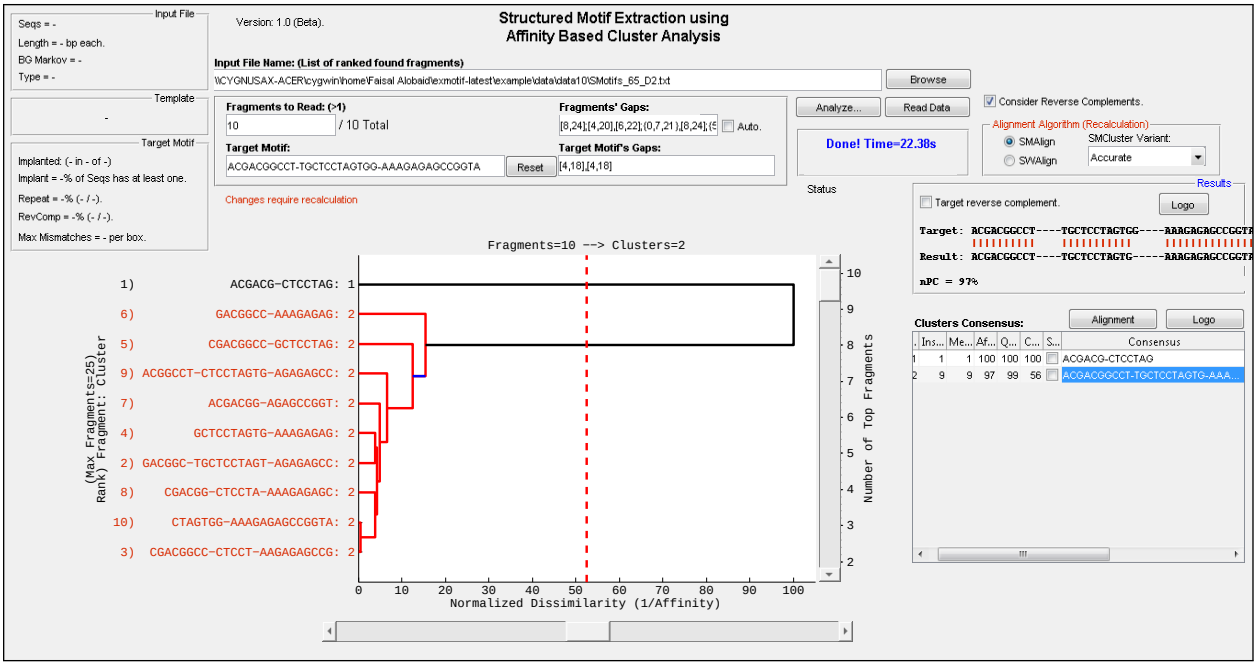


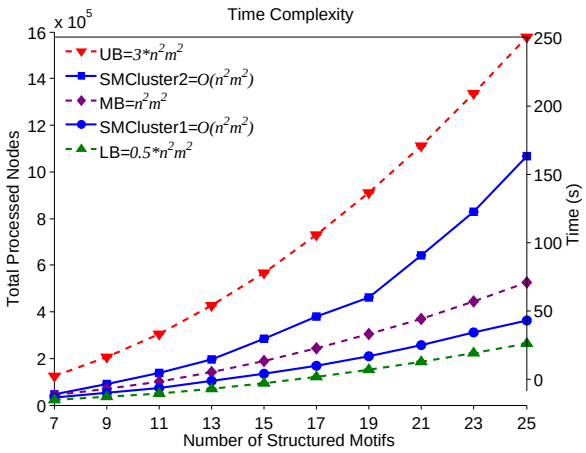
Fig. 5.8: SMCluster GUI screenshot. The top part shows user input of the file name that contains the set of structured motifs and the number of top motifs to multi-align. Top right shows the algorithm used for the underlying pairwise alignment and the variant of SMCluster to use (accurate vs approximate). The dendrogram of the multiple alignment result is shown in the middle. Also shown the cutoff threshold in the dendrogram (dashed vertical line) where the user can control its position using the bottom slider. On the right we see a list of clusters' consensus structured motifs (two clusters reported based on the chosen cutoff threshold; the second one is highlighted).

Core i5-430UM processor having 1.7GHz max speed with 4GB of RAM.

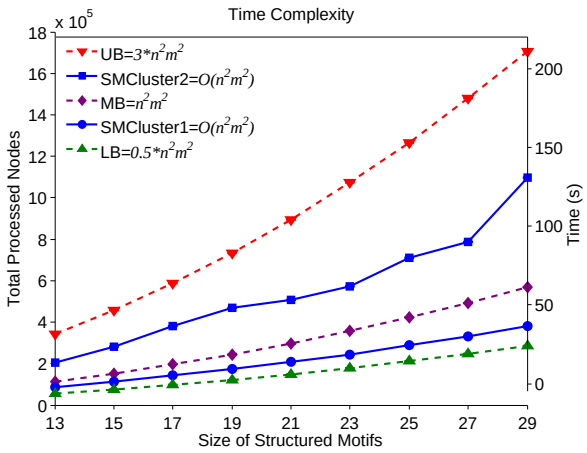
We first examine the empirical time complexity of SMCluster's two variants, in proportion to the number of aligned structured motifs n (Figure 5.9a) and the length of the aligned structured motifs m (Figure 5.9b). Notice the empirical time complexity of SMCluster2 (slow-accurate variant) is practically asymptotic to $O(n^2m^2)$ rather than its theoretical worst case of $O(n^3m^2)$.

Next we examine the alignment result of SMCluster against the discussed current multiple alignment algorithms. Figure 5.10 shows SMCluster multiple alignment result for the example problem presented in Section 5.1.1. SMCluster's result is better than the other algorithms' results satisfying all gap constraints while only having three columns with one mismatch each.

Table 5.2 shows five generated datasets each having 20 structured motifs. These datasets were generated to resemble the kind of datasets that we will encounter in the next chapter when we deal with solving the structured motif extraction problem. Table 5.3 shows a comparison between SMCluster and current algorithms tested on the generated datasets.



(a) The time complexity of SMCluster varying the number of aligned structured motifs $n \in [5, 23]$ (structured motif minimum length is fixed at $m = 30$).



(b) The time complexity of SMCluster varying the size of the aligned structured motifs $m \in [13, 29]$ (the number of structured motifs is fixed at $n = 24$).

Fig. 5.9: The empirical time complexity of SMCluster's two variants, SMCluster1 (fast-approximate) and SMCluster2 (slow-accurate). Also showing upper, middle and lower bounds for comparison.

```
 1) ACGACG[8,24]CTCCTAG
 2) GACGGC[4,20]TGCTCCTAGT[6,22]AGAGAGCC
 3) CGACGGCC{0,7,21}CTCCT[8,24]AAGAGAGCCG
 4) GCTCCTAGTG{5,10,19}AAAGAGAG
 5) CGACGGCC[4,20]GCTCCTAG
 6) GACGGCC[19,49]AAAGAGAG
 7) ACGACGG[25,55]AGAGCCGGT
 8) CGACGG[7,23]CTCCTA[6,22]AAAGAGAGC
 9) ACGGCCT[4,20]CTCCTAGTG[5,21]AGAGAGCC
10) CTAGTGG[4,18]AAAGAGAGCCGGTA
```

(a) A set of 2-box and 3-box structured motifs given as input to SMCluster.

```
                                                    ** *
 1) ACGACG----------CTCCTAG--------------------
 2) --GACGGC------TGCTCCTAGT--------AGAGAGCC----
 3) -CGACGGCC-------CTCCT----------AAGAGAGCCG---
 4) ---------------GCTCCTAGTG-----AAAGAGAG------
 5) -CGACGGCC------GCTCCTAG--------------------
 6) --GACGGCC---------------------AAAGAGAG------
 7) ACGACGG----------------------------AGAGCCGGT-
 8) -CGACGG---------CTCCTA--------AAAGAGAGC-----
 9) ---ACGGCCT------CTCCTAGTG-------AGAGAGCC----
10) ----------------CTAGTGG-------AAAGAGAGCCGGTA
```

(b) The resulted multiple alignment from SMCluster. All gap constraints are satisfied and only three columns have one mismatch each (* marked). The alignment quality is 99.1% (Equation (5.5)) and the gap constraints are satisfied in 100% of the rows.

Fig. 5.10: SMCluster multiple alignment of a set of structured motifs.

The results in Table 5.3 show the dominance of SMCluster (both variants) over the other algorithms. The overall winner is SMCluster2 (slow-accurate variant), but SMCluster1 was slightly better in dataset2 and came in second overall. ClustalW came in third in terms of alignment quality, however, it does not guarantee gap constraints satisfaction. T-COFFEE and M-COFFEE violated all gap constraints in different ways. T-COFFEE ignored the given minimum gaps and did not introduce any gaps in the multiple alignment. Also, M-COFFEE ignored given minimum gaps, but added gaps as it saw necessary in different places than allowed by the gap constraints. STAMP seems to be guaranteeing gap constraints satisfaction, but this guarantee is only due to using an ungapping pairwise alignment (ungapped local alignment). Therefore, it maintains the minimum

Table 5.2: The datasets used to compare SMCluster against current algorithms.

No.	Dataset1	Dataset2	Dataset3	Dataset4	Dataset5
1	AATC[1,22]TCTTA	TATG[2,10]TCTT	TCCA[1,11]AGCC	CAAA[1,10]TGAA	TATT[3,15]GGAT
2	AATCT[1,22]TCTT	GTTA[2,10]TTCT	TAGC[1,11]GTCC	AAGC[1,10]GAAC	GGAG[3,15]CCGG
3	ATCT[1,22]AATCT	GTGA[2,10]GTTA	TCCA[1,11]CACA	TGAA[1,10]TAGC	GGAG[3,15]CCGG
4	ATCTA[1,22]CTTA	TTAT[2,10]TCTT	TCCA[1,11]AGCC	AAAG[1,10]TGAA	CCGG[3,15]GAGG
5	CTAA[1,22]TCTTA	TTAT[2,10]TTCT	GATA[1,11]GTCC	CCGT[1,10]TTCC	CGGT[3,15]ATGG
6	ATCT[1,22]AATCT	AATG[2,10]TGTC	TCCA[1,11]CACA	ACCT[1,10]CGTC	GTTA[3,15]TGGA
7	ATCT[1,22]CTTAC	TGTC[2,10]GCGT	GATA[1,11]TCCA	AACC[1,10]GCCG	GTTA[3,15]TGGA
8	TCTA[1,22]ATCTT	GTTA[2,10]GCGT	GTCC[1,11]CCAC	TTAG[1,10]TCTT	AAAG[3,15]CCGG
9	ATCT[1,22]ATCTT	TGAA[2,10]TTAT	GTCC[1,11]CACA	AAGA[1,10]GGCC	AGGA[3,15]CCGG
10	ATCTA[1,22]TTAC	AATG[2,10]ATGT	GTCC[1,11]GCCA	AGCA[1,10]AACC	AGGA[3,15]CCGG
11	ATCT[1,22]TCTTA	TGTT[2,10]TCTT	CGAG[1,11]CCAC	TAGC[1,10]TTCC	GGAG[3,15]CGGT
12	AATC[1,22]AATCT	AATG[2,10]ATGT	CGAG[1,11]CCAC	TAGC[1,10]GGCC	AAGG[3,15]CCGG
13	AATC[1,22]TCTTA	GAAT[2,10]CTTG	GATA[1,11]GTCC	CTTA[1,10]GTCT	CCGG[3,15]GAGG
14	ATCTA[1,22]ATCT	TGAA[2,10]ATGT	GCGA[1,11]TCCA	CACC[1,10]TGAA	TTAT[3,15]TGGA
15	ATCTA[1,22]TCTT	TGTC[2,10]GCGT	GTCC[1,11]CCAC	GCCG[1,10]CTTC	AAAG[3,15]CCGG
16	ATCT[1,22]CTTAC	GAAT[2,10]TGTC	TAGC[1,11]GTCC	GCAC[1,10]TAGC	CGGT[3,15]ATGG
17	TCTAA[1,22]CTTA	ATGT[2,10]TCTT	TAGC[1,11]AGTC	CTTA[1,10]GGCC	TTAT[3,15]TGGA
18	AATC[1,22]AATCT	TGAA[2,10]TTAT	AGTC[1,11]GCCA	ACCT[1,10]AGCG	AAGG[3,15]CCGG
19	TCTAA[1,22]TCTT	GTGA[2,10]TATG	GATA[1,11]TCCA	AAGC[1,10]AACC	GGAG[3,15]CGGT
20	TCTAA[1,22]CTTA	TGTC[2,10]GGCG	AGTC[1,11]GCCA	GAAC[1,10]AGCG	TTAT[3,15]GAGG

Table 5.3: Comparing the result quality of SMCluster against current algorithms.

Algorithm	Dataset1		Dataset2		Dataset3		Dataset4		Dataset5		Average	
	Quality	Satisfaction	Quality	Satisfaction	Quality	Satisfaction	Quality	Satisfaction	Quality	Satisfaction	Quality	Satisfaction
ClustalW	**100**	**100**	72	**100**	90	90	60	66	74	**100**	79.2	91.2
T-COFFEE	59	0	48	0	48	0	45	0	52	0	50.4	0
M-COFFEE	88	0	60	0	65	0	61	0	67	0	68.2	0
STAMP	73	**100**	66	**100**	66	**100**	66	**100**	74	**100**	69.0	**100**
SMCluster1	**100**	**100**	**98**	**100**	**100**	**100**	88	**100**	93	**100**	95.8	**100**
SMCluster2	**100**	**100**	94	**100**	**100**	**100**	**93**	**100**	**99**	**100**	**97.2**	**100**

Quality %: The multiple alignment quality as measured using Equation (5.5).
Satisfaction %: The percentage of rows that satisfaied the gap constraints.

given gaps, but does not add any more gaps, and only slides the input structured motifs to produce its multiple alignment, which had a low overall quality score.

5.5 Discussion

In this chapter we introduced SMCluster, a new progressive multiple alignment algorithm specifically designed to solve the structured motif multiple alignment problem. Our empirical tests show that SMCluster always satisfy all gap constraints while maximizing the alignment quality as defined in (5.5). We presented two variants SMCluster1 (fast-approximate) and SMCluster2 (slow-accurate) for user choice based on application. We also showed that current multiple alignment algorithms are not suitable to solve the structured motif alignment problem.

In the next chapter we use SMCluster as an essential part to solve the structured motif extraction problem. A MATLAB implementation and an online version of SMCluster can be accessed from *http://bioproject.syr.edu/smtools*.

CHAPTER 6
EXTRACTION OF STRUCTURED MOTIFS

6.1 Overview

In this chapter we introduce a new algorithm *SMExtract* to solve the structured motif extraction problem. Existing algorithms require almost perfect information about the structured motif's size and gaps. Minor perturbations increase the time complexity and decrease the quality of results. The main contribution of SMExtract is to relax the user provided information without sacrificing the quality of the extracted motif.

6.2 SMExtract

In Chapter 2, we found that existing structured motif extraction algorithms suffer from three major drawbacks:

1. They are only capable of extracting patterns strictly conforming to user specified parameters that require an unreasonable level of prior knowledge.
2. Some algorithms are only capable of finding limited patterns, such as dyads.
3. The computational effort required by exact algorithms increases exponentially with the number of allowed mismatches in the pattern and the number of boxes in the given template.

In other words, existing algorithms require considerable knowledge about the unknown pattern to be extracted, are limited in the type of patterns that can be extracted, and are computationally expensive for longer patterns if mismatches are to be allowed.

In this section we describe SMExtract that addresses all three concerns mentioned above. That is, SMExtract requires less parameters about the pattern to be extracted, is versatile in finding different types of patterns from simple motifs to multi-box structured motifs without requiring specification of the pattern type by the user, and is more efficient in extracting longer patterns with mismatches.

To extract an unknown overrepresented pattern M from a set of DNA sequences $\mathcal{S}$ (as defined in Section 1.2), SMExtract performs three steps (Figure 6.1):

1. Formulates a two-box template $\mathcal{T}$ based on user specified parameters.
2. Extracts and ranks overrepresented two-box structured motifs (*fragments*) using Exmotif [9], i.e., execute $\mathcal{F} = \text{Exmotif}(\mathcal{S}, \mathcal{T})$, where $\mathcal{F} = \{F_i\}$ is the set of extracted fragments.
3. Multi-aligns (clusters) together the top x fragments using SMCluster, i.e., executes $\widetilde{M} = \text{SMCluster}(\mathcal{F}_{i=1\ldots x})$, where $\widetilde{M}$ is the recovered structured motif.

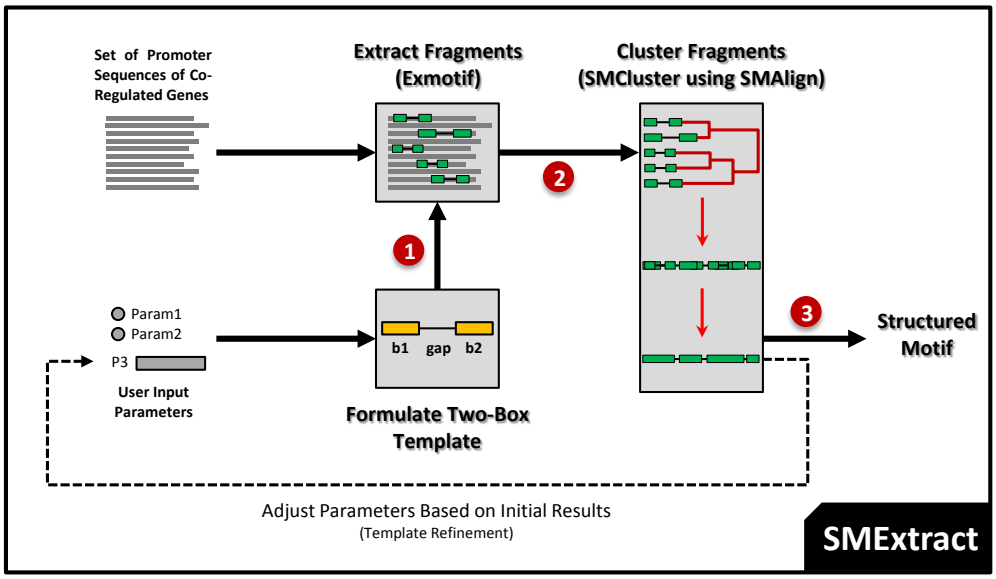


Fig. 6.1: SMExtract algorithm overview. 1) Formulate two-box template based on user parameters. 2) Extract two-box fragments. 3) Multi-align (cluster) fragments to recover the unknown pattern in the input sequences (e.g., structured motif).

The novel approach of SMExtract is that it extracts an overrepresented pattern M from $\mathcal{S}$ by first extracting M's fragments, then assembles these fragments to uncover the desired full pattern. This indirect approach makes SMExtract efficient and versatile, requiring less user knowledge about the pattern. This is achieved by the unique features of SMAlign used in SMCluster, as discussed in the previous two chapters (Chapter 4 and 5).

To extract a ranked set of fragments, $\mathcal{F}$, we chose Exmotif[§] because it is an exact algorithm that extracts and ranks patterns conforming to a user specified template.

6.2.1 Two-Box Template Formulation

In general, a two-box template $\mathcal{T}$ is as given below. Note that we need to specify up to nine parameters, which are four parameters for the size ranges of the two boxes $[b_i^{\min}, b_i^{\max}]$, $i = 1, 2$, two parameters for the gap in-between $[g^{\min}, g^{\max}]$, and three parameters for the template quorum $q_{\mathcal{T}}$ and the mismatches for the two boxes $d_i^{\mathcal{T}}$ measured in base pairs (bp).

$$\mathcal{T} = [b_1^{\min}, b_1^{\max}] : d_1^{\mathcal{T}} - [g^{\min}, g^{\max}] - [b_2^{\min}, b_2^{\max}] : d_2^{\mathcal{T}}, \;\; q_{\mathcal{T}} \tag{6.1}$$

For example, $\mathcal{T} = [3, 5] : 0 - [5, 10] - [3, 5] : 0, q_{\mathcal{T}} = 70\%$ is a two-box template where both boxes have size in the range $[3, 5]$ with zero mismatches and gap range $[5, 10]$, with 70% quorum. User's knowledge of the template is expected to result in better motif extraction, but in practice, the user does not know the exact specification. Instead, we expect the user to be more comfortable in specifying the following six parameters:

$$\left\langle [L_M^{\min}, L_M^{\max}], [C_M^{\min}, C_M^{\max}], d_{\mathcal{T}}, q_{\mathcal{T}} \right\rangle, \tag{6.2}$$

where

1. $[L_M^{\min}, L_M^{\max}]$: The range within which we expect the length of M to lie; ideally, $L_M^{\max} \geq$

[§]Exmotif can be replaced by any other suitable algorithm that can extract two-box structured motifs.

$|M|_{\text{max}} \geq |M|_{\text{min}} \geq L_M^{\text{min}} \geq 2bp$ (default, $[10, 30]$).

2. $[C_M^{\text{min}}, C_M^{\text{max}}]$: The range of conserved segments in M; conserved within $\leq d_{\mathcal{T}}$ mismatches (default, $[4, 6]$).
3. $d_{\mathcal{T}} \geq 0$: The maximum number of mismatches between any two fragments ($d_{\mathcal{T}} = d_1^{\mathcal{T}} + d_2^{\mathcal{T}}$; default, $d_{\mathcal{T}} = 0$).
4. $q_{\mathcal{T}} > 0\%$: The template quorum. This instructs Exmotif to report the two-box fragments that exists, at least once, in at least $q_{\mathcal{T}}\%$ of the sequences in $\mathcal{S}$ (default, $q_{\mathcal{T}} = 70\%$).

The user is expected to provide an educated guess for these parameters to extract an unknown pattern M. These six parameters are less demanding than requiring the user to formulate a fully fledged template having $5k - 1$ parameters (recall that, if the desired structured motif has k boxes, then a total of $5k - 1$ parameters are required (see Section 2.2.1)). This leads to a reduction in the number of parameters required by SMExtract and an increase in the relevance to the problem formulation, while at the same time, maintaining the flexibility and versatility of the algorithm.

Using the first four parameters of (6.2), we propose the following set of equations to formulate our two-box template $\mathcal{T}$:

$$\begin{aligned} b^{\text{min}} &= b_1^{\text{min}} = b_2^{\text{min}} = \max(2, \min(C_M^{\text{min}}, \lfloor L_M^{\text{min}}/2 \rfloor - 1)) \\ b^{\text{max}} &= b_1^{\text{max}} = b_2^{\text{max}} = \max(b^{\text{min}}, C_M^{\text{max}}) \\ g^{\text{min}} &= \max(1, \lfloor L_M^{\text{min}}/2 \rfloor - 2b^{\text{min}}) \\ g^{\text{max}} &= \max(g^{\text{min}}, L_M^{\text{max}} - 2b^{\text{min}}). \end{aligned} \tag{6.3}$$

This template formulation ensures maximum coverage in extracting an unknown pattern of length $\in [L_M^{\text{min}}, L_M^{\text{max}}]$. For example, suppose we formulate a template $\mathcal{T}$ from our knowledge of a given structured motif $M =$ `GCTAAGCTAA[3,15]TTTGAAGCC[5,10]TTGCAAC`, i.e., $[L_M^{\text{min}} = 34, L_M^{\text{max}} = 51]$, and we choose $[C_M^{\text{min}} = 4, C_M^{\text{max}} = 6]$ to be less than or equal to M's minimum and maximum box sizes respectively. Then, Figure 6.2 shows the template's coverage of M by showing a partial set of extracted fragments that conform to the formulated template, $\mathcal{T} = [4, 6] - [9, 43] - [4, 6]$,

using (6.3). Structured motifs with minimum or maximum lengths outside of the specified range ($[L_M^{\min}, L_M^{\max}]$) would, at least, be partially extracted by SMExtract.

```
M:   GCTAAGCTAA[3,15]TTTGAAGCC[5,10]TTGCAAC   ⟶   T:   [4,6]-[ 9,43]-[4,6]
F1:  GCTA------[3,15]TTTG                    ⟶         4  -[ 9,24]-  4
F2:  GCTAA-----[3,15]-TTGA                   ⟶         5  -[ 9,24]-  4
F3:    TAAG----[3,15]--TGAAG                 ⟶         4  -[ 9,24]-  5
F4:     AAGC---[3,15]---GAAG                 ⟶         4  -[ 9,24]-  4
F5:     AAGC---[3,15]----AAGCC               ⟶         4  -[10,25]-  5
F6:     AAGC---[3,15]---------[5,10]TTGC     ⟶         4  -[20,37]-  4
F7:      AGCTA-[3,15]---------[5,10]-TGCAA   ⟶         5  -[19,36]-  5
F8:       GCTAA[3,15]---------[5,10]-TGCAAC  ⟶         5  -[18,35]-  6
F9:                 TTTGA----[5,10]TTGCAA    ⟶         5  -[ 9,14]-  6
F10:                 TTGAA---[5,10]-TGCAA    ⟶         5  -[ 9,14]-  5
F11:                  TGAA---[5,10]--GCAA    ⟶         4  -[10,15]-  4
F12:                   GAAG--[5,10]--GCAAC   ⟶         4  -[ 9,14]-  5
F13:                   GAAGC-[5,10]---CAAC   ⟶         5  -[ 9,14]-  4
```

Fig. 6.2: Showing $\mathcal{T}$'s fragments (F_i) coverage of a given structured motif M, where $\mathcal{T} = [4,6] - [9,43] - [4,6]$ is formulated from M's parameters (i.e., $[L_M^{\min} = 34, L_M^{\max} = 51]$, $[C_M^{\min} = 4, C_M^{\max} = 6]$) using Equations (6.3).

It is important to choose the template's parameters carefully to avoid extracting spurious fragments that appear by chance in the given sequences. To achieve that, we examine the statistical properties of our template $\mathcal{T}$, specifically, we are interested in calculating the expected number of fragments conforming to $\mathcal{T}$ to be less than one for a set of t random sequences each of length n. The key concern is as follows: if the expected number of fragments, conforming to $\mathcal{T}$, in a set of t random sequences is high, then the discovered motif is more likely to be spurious. Alternatively, if the expected number is low, then an extracted motif has a much better chance of being a real motif. The following analysis is an extension to the one given in [41] to conform to our two-box template.

Suppose F is a two-box fragment conforming to template $\mathcal{T}$ (i.e., F has box sizes $b_1, b_2 \in [b^{\min}, b^{\max}]$ and its gap range is within the range $[g^{\min}, g^{\max}]$). For example, $F = \texttt{ACCG} - [3,9] - \texttt{TGCCC}$ is a fragment conforming to the template $\mathcal{T} = [3,5] : 0 - [5,10] - [3,5] : 0$ because F has $b_1 = 4$ and $b_2 = 5$ are within the gap range $[3,5]$, and F's gap range ($[3,9]$) is within $\mathcal{T}$'s gap range of $[5,10]$. Then the probability of fragment F occurring in a given position in a random sequence with at most $d_{\mathcal{T}}$ mismatches is:

$$P_1(F) = \sum_{i=0}^{\min(b_1+b_2, d_{\mathcal{T}})} \binom{b_1+b_2}{i} \left(\frac{3}{4}\right)^i \left(\frac{1}{4}\right)^{b_1+b_2-i}. \tag{6.4}$$

Then the *maximum* probability of fragment F appearing at least once in a random sequence of length n is:

$$P_2(F) = 1 - (1 - P_1(F))^{(n-(b_1+b_2+g^{\min})+1)\cdot(g^{\max}-g^{\min}+1)}. \tag{6.5}$$

This is an *overestimate* of the actual probability. Then the expected number of fragment F appearing at least once in at least $q_{\mathcal{T}}$ percent of t random sequences each of length n is:

$$E(F) = 4^{b_1+b_2} \cdot \sum_{i=\max(1,\lfloor t\cdot q_{\mathcal{T}} \rfloor)}^{t} \binom{t}{i} P_2(F)^i (1 - P_2(F))^{(t-i)}. \tag{6.6}$$

If the user chooses values for the template $\mathcal{T}$ such that $E(F) \geq 1$ for any F, then SMExtract alerts the user about the possibility of extracting spurious fragments, which lowers the confidence in the final result. Ideally, $\mathcal{T}$ must be chosen such that $E(F) < 1$ for all F; this ensures that each fragments F conforming to template $\mathcal{T}$ is not a spurious fragment. Suppose F_{min} is the minimum fragment conforming to $\mathcal{T}$, i.e., F_{min} has $b_1 = b_2 = b^{\min}$. It can be shown that if $E(F_{\text{min}}) < 1$ then $E(F) < 1$ for all F conforming to $\mathcal{T}$. Therefore, we only need to check if $E(F_{\text{min}}) < 1$ to make sure that our template $\mathcal{T}$ has a better chance of avoiding spurious fragments.

6.2.2 Extracting Structured Motifs

We now examine the behavior of SMExtract in the context of the sample problem posed at the end of Section 1.2. Recall that the set $\mathcal{S}$ contains $t = 100$ randomly generated sequences each of length $n = 1000$bp. In this set of sequences, we implanted 136 instances of the structured motif:

$$M = \texttt{ACGACGGCCT[4,18]TGCTCCTAGTGG[4,18]AAAGAGAGCCGGTA},$$

such that each sequence has one instance of M and the remaining 36 instances are distributed randomly among the sequences. Also, 62 of the 136 instances are on the complementary strand, and M has one arbitrarily positioned mismatch per box ($d_i^M = 1, \forall i \in [1, 3]$). The goal is to extract M from the given set using limited or no *a priori* information about the structure of M, i.e., we do

not assume exact knowledge of the number of boxes or gap sizes, except a vague idea about the required parameters specified in (6.2).

First, to detect patterns on both DNA strands, we append each sequence in $\mathcal{S}$ with its reverse complement as a preprocessing step. Then, we define our template parameters using the default values: $[L_M^{\text{min}}, L_M^{\text{max}}] = [10, 30]$, $[C_M^{\text{min}}, C_M^{\text{max}}] = [4, 6]$, $d_{\mathcal{T}} = 0bp$, and $q_{\mathcal{T}} = 70\%$. We are assuming no knowledge of the implanted pattern M, thus the chosen parameters need not strictly conform to M. We then formulate our two-box template $\mathcal{T} = [4, 6] : 0 - [1, 22] - [4, 6] : 0, q_{\mathcal{T}} = 70\%$ as per (6.3). Note that for this template, $E(F_{\text{min}}) = 4.7 \times 10^{-13} < 1$. Next, we execute $\mathcal{F}$=Exmotif($\mathcal{S}, \mathcal{T}$) to generate a ranked list of overrepresented two-box fragments. Finally, we execute SMCluster2($\mathcal{F} = \{F\}_{i=1}^{40}$) to cluster together the top 40 fragments out of 330 total fragments reported by Exmotif (henceforward, SMExtract by default uses SMCluster2, the slow-accurate variant descriped in Section 5.2.3). We usually start by taking the top x fragments, where $x \leq \sqrt[3]{2nt}$ (e.g., $(x = 40) \leq (\sqrt[3]{2 \times 1000 \times 100} = 58)$), to balance result quality and time complexity, as discussed later in Section 6.3.

Figure 6.3a shows the result of Exmotif (top 40 reported fragments), and Figure 6.3b shows the implanted pattern M with the resulting consensus structured motif, and the multiple alignment produced by SMCluster. Notice the unique behavior of SMCluster when multi-aligning the two-box fragments. One of the two boxes acts as an anchor while the other box explores the surrounding area to uncover the hidden pattern. This is analogous to constructing a panoramic image by stitching together multiple smaller images having overlapping regions. Current popular multiple alignment algorithms do not possess such behavior, which is very useful for SMExtract to reconstruct successfully the unknown overrepresented pattern.

Table 6.1 shows the result of multi-aligning the top 40 fragments using SMCluster versus other popular methods. These methods do not take gap constraints as parameters (as discussed in Section 5.1.2), so we present them with the fragments having the minimum gap between the two boxes (e.g., fragment number 38 in Figure 6.3 is presented as `AGAG-CGGTA` instead of `AGAG[1,22]CGGTA`). Also, since none of these methods automatically check for the reverse complement of the input

```
 38) AGAG[1,22]CGGTA
  8) AGAG[1,22]GCCGG
 14) CCGGC[1,22]CTCT
  2) CCGGC[1,22]CTTT
 16) AAAG[1,22]GCCGG
 26) GCTCC[1,22]AAGA
 21) TAGGA[1,22]GGCC
 10) CGGC[1,22]TCTTT
 37) AAAGA[1,22]GCCG
 27) CGGCT[1,22]CTTT
 17) GGCTC[1,22]CTTT
 23) AAAG[1,22]GAGCC
 28) GTGG[1,22]AGAGC
  6) AAGAG[1,22]GCCG
 35) CTCT[1,22]CACTA
 31) TAGTG[1,22]AAAG
 20) CTCTT[1,22]CTAG
 36) CTCT[1,22]ACTAG
 39) TCTTT[1,22]CTAG
  5) CTCCT[1,22]AAGA
  4) TCCT[1,22]AAAGA
 18) TCTTT[1,22]GGAG
 34) TCTTT[1,22]AGGA
  1) CTTT[1,22]CTAGGA
 19) TCCTAG[1,22]AAAG
 25) CTTT[1,22]TAGGA
 30) CTTT[1,22]CTAGG
  3) GTGG[1,22]GAGAG
  7) CTCTC[1,22]CCAC
 22) AGTG[1,22]GAGAG
 12) AGTG[1,22]AAAGA
 29) TCTTT[1,22]CCAC
 32) AGTGG[1,22]AAAG
 40) AGTGG[1,22]AAGA
 11) TCTCT[1,22]CCAC
 13) AGTG[1,22]AGAGA
 15) TCTCT[1,22]CACT
 24) GTGG[1,22]AGAGA
  9) CTCT[1,22]CCACT
 33) AGTGG[1,22]AGAG
```

(a) The top 40 fragments reported by Exmotif showing the rank for each fragment. The list is sorted as in the multiple alignment result in (b).

```
Implanted:  ACGACGGCCT----TGCTCCTAGTGG----AAAGAGAGCCGGTA
                 ||||      ||||||||||    ||||||||||||||
Output:     -----GGCC------GCTCCTAGTGG----AAAGAGAGCCGGTA

     38)    ----------------------------------AGAG-CGGTA
      8)    -------------------------------AgAG---GCCGG--
r    14)    -------------------------------AgAG---GCCGG--
r     2)    -------------------------------AAAG---GCCGG--
     16)    -------------------------------AAAG---GCCGG--
     26)    ---------------GCTCC-----------AAGA---------
r    21)    -----GGCC--------TCCTA----------------------
r    10)    -------------------------------AAAGA--GCCG---
     37)    -------------------------------AAAGA--GCCG---
r    27)    -------------------------------AAAG--AGCCG---
r    17)    -------------------------------AAAG-GAGCC----
     23)    -------------------------------AAAG-GAGCC----
     28)    ----------------------GTGG--------AGAGC-----
      6)    --------------------------------AAGAG-GCCG---
r    35)    --------------------TAGTG-----AgAG----------
     31)    --------------------TAGTG-----AAAG----------
r    20)    -------------------CTAG--------AAGAG--------
r    36)    -------------------CTAGT--------AGAG--------
r    39)    -------------------CTAG-------AAAGA---------
      5)    -----------------CTCCT----------AAGA---------
      4)    -----------------TCCT---------AAAGA---------
r    18)    -----------------CTCC----------AAAGA---------
r    34)    -----------------TCCT---------AAAGA---------
r     1)    -----------------TCCTAG-------AAAG----------
     19)    -----------------TCCTAG-------AAAG----------
r    25)    -----------------TCCTA--------AAAG----------
r    30)    -----------------CCTAG-------AAAG----------
      3)    ----------------------GTGG-------GAGAG------
r     7)    ----------------------GTGG-------GAGAG------
     22)    ----------------------AGTG--------GAGAG------
     12)    ----------------------AGTG-----AAAGA---------
r    29)    ----------------------GTGG-----AAAGA---------
     32)    ----------------------AGTGG----AAAG----------
     40)    ----------------------AGTGG-----AAGA---------
r    11)    ----------------------GTGG------AGAGA-------
     13)    ----------------------AGTG-------AGAGA-------
r    15)    ----------------------AGTG-------AGAGA-------
     24)    ----------------------GTGG------AGAGA-------
r     9)    ----------------------AGTGG------AGAG--------
     33)    ----------------------AGTGG------AGAG--------
```

(b) The Result of SMCluster multi-aligning the top 40 fragments. 'r' indicates that SMCluster used the reverse complement of the reported fragment in the multiple alignment.

Fig. 6.3: The Result of SMExtract($\mathcal{S}, \mathcal{T}$), where $\mathcal{S}$ is a set of $t = 100$ input sequences each of length $n = 1000$bp, and $\mathcal{T} = [4, 6] : 0 - [1, 22] - [4, 6] : 0,\ q_{\mathcal{T}} = 70\%$ is the formulated two-box template using default user parameters. First, Exmotif reports the top fragments based on $\mathcal{T}$, then SMCluster multi-align these fragments to uncover the unknown pattern.

fragments, we present the list of fragments in the correct orientation (based on the result in Figure 6.3b) to give them the best chance. The result in Table 6.1 shows that none of the methods produced a better alignment compared to SMCluster. We quantified the quality of their resulted alignment

Table 6.1: SMCluster vs. Other Methods*

Implanted: ACGACGGCCT-TGCTCCTAGTGG-AAAGAGAGCCGGTA	*nPC*
SMCluster: GGCC-GCTCCTAGTGG-AAAGAGAGCCGGTA	**81%**
ClustalW[37]: GGCCCTAGTGG-GAAAGAGAGCCGGTA	59%
T-COFFEE[38]: AGAGAAAGGG	18%
M-COFFEE[39]: GCTGCCTCCTAGTGGGAGAGG	30%
STAMP[30]: TCGTGGGAAAGAGCCGGC	26%

*The consensus structured motif for each method resulted from multi-aligning the top 40 fragments.

against the implanted pattern using the nucleotide performance coefficient (*nPC*) measure [8]:

$$nPC = \frac{nTP}{nTP + nFP + nFN}, \tag{6.7}$$

where *nTP*, *nFP* and *nFN* are the nucleotide true positive, false positive and false negative respectively. The results in Table 6.1 indicate that other popular methods are not suitable for the task at hand, motivating the use of SMCluster, confirming our conclusion in Chapter 5. ClustalW produced the next best result after SMCluster.

6.2.3 Template Refinement

Another unique feature of SMCluster is that it propagates the gap constraints throughout the hierarchical clustering. Thus, the full consensus structured motif for the above example problem, including the gap constraints, is reported as `GGCC[0,20]GCTCCTAGTGG[1,15]AAAGAGAGCCGGTA`. Therefore, as an optional feedback step, we now have the ability to adjust the template's parameters based on this initial result. If the result of the second run is consistent with the initial result, then this increases our confidence in the extracted pattern. On the other hand, if the new results deviates drastically from the first one, then further analysis of the initial dendrogram and further tweaking of the initial template is necessary to obtain better results.

For instance, the maximum length of the initial resulting consensus structured motif (`GGCC[0,20]GCTCCTAGTGG[1,15]AAAGAGAGCCGGTA`) is 64bp. Updating the user parameter $L_M^{\max}$ from the default value of 30 to 64 and using equations (6.3), we obtain the new template $\mathcal{T} = [4,6] - [1,56] - [4,6]$. Using this new template in SMExtract, we obtained the result in Table 6.2. The new resulting structured motif matches closely with the initial result in Figure 6.3b (i.e., both results show 3-box structured motifs, and their $nPC = 71\%$), which increases our confidence in the extracted structured motif. Also, we can multi-align together the results of multiple runs of SMExtract to obtain even better results, as seen in this example (Table 6.2).

So far, we have always picked the alignment in the root cluster of the dendrogram to represent the extracted structured motif. However, the user can closely inspect the produced dendrogram, as

Table 6.2: The result of updating $\mathcal{T}$ based on the initial result.

	nPC[1]
Implanted: ACGACGGCCT-TGCTCCTAGTGG-AAAGAGAGCCGGTA	
Initial: -----GGCC---GCTCCTAGTGG-AAAGAGAGCCGGTA	81%
Updated: ACGACGGC---TGCTCCTAGTGG-AAAGAGAGCCG---	86%
Initial+Updated[2]: ACGACGGCC--TGCTCCTAGTGG-AAAGAGAGCCGGTA	97%

[1] nPC measured between resulted structured motif and implanted.
[2] Aligning the initial result with the updated result yielded an even better result.

in Figure 6.4, and exercise judgment, using the *conservation* (5.4) and the *quality* (5.5) measures, of the best cutoff threshold to use, and then pick the best cluster to represent the extracted motif.

Finally, if the initial template $\mathcal{T}$ did not extract any fragments, or few ones, then the user can do any or all of the following to adjust the template's parameters defined in (6.2):

1. Decrease the quorum $q_{\mathcal{T}}$.
2. Increase the range of the conserved segments $[C_M^{\min}, C_M^{\max}]$.
3. Increase the range of the motif length $[L_M^{\min}, L_M^{\max}]$.
4. Increase the number of mismatches $d_{\mathcal{T}}$.

The refined template should then be checked against extracting spurious fragments, using Equation (6.6), such that $E(F_{\min}) < 1$.

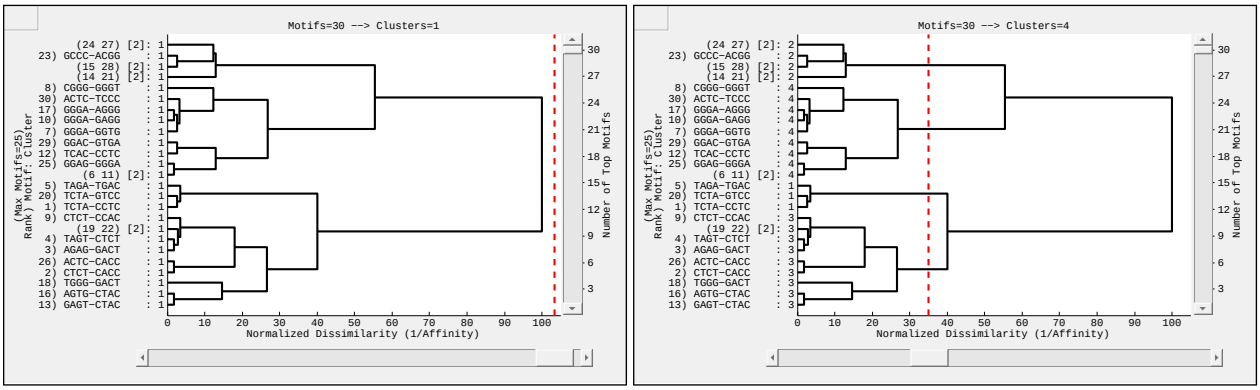


Fig. 6.4: The user can set the cutoff threshold (dashed vertical line) in SMCluster's dendrogram to determine the number of clusters to report as output. a) the root cluster is chosen to represent the extracted motif, b) four clusters are chosen to represent four extracted motifs.

6.2.4 Discrete Gap Constraints

In the previous section we showed how SMExtract extracts an unknown pattern in the input sequences by clustering the most highly ranked (overrepresented) two-box fragments obtained from Exmotif. So far we have assumed that each two-box fragment has a gap constraint as given in $\mathcal{T}$ (Figure 6.3a). In this section, we introduce a heuristic to recover the actual gap sizes of all fragments from the input sequences. This restricts the gap constraints which, in turn, helps in eliminating false alignments as explained next.

To recover the actual gap sizes for each fragment, we modified Exmotif to output *occurrence data*, without affecting Exmotif's time complexity. That is, for each fragment found, Exmotif outputs the fragment's number of occurrences in each sequence and the gap size of each of these occurrences. Figure 6.5 shows example occurrence data for ten fragments in six sequences. The format of the data is (`SequenceNo.:No.ofOccurrences {ActualGapSizes}`). For example, fragment no.3 is not present in sequences {1,2,4}, but has 1 occurrence in sequence {6} (actual gap size={22}) and 2 occurrences in each of sequences {3,5} (actual gap sizes={12,17,20}). We calculate the occurrence agreement for each sequence, which indicates the percentage of fragments that have the same number of occurrences in that sequence.

10 Fragments	Occurrence frequency and gap sizes in 6 sequences.					
1) ATTAGC-TGGCAG --	1:0	2:0	3:2 {20,12}	4:0	5:3 {17,5,8}	6:1 {22}
2) GCCTGG-TTTCT --	1:1 {18}	2:1 {16}	3:3 {12,4,16}	4:1 {16}	5:1 {9}	6:1 {14}
3) TGATAT-TTTCTG --	1:0	2:0	3:2 {20,12}	4:0	5:2 {17,17}	6:1 {22}
4) TATTAG-TTCTGG --	1:0	2:1 {22}	3:3 {18,10,22}	4:1 {22}	5:1 {15}	6:2 {20,5}
5) TAGCCT-TTTCTG --	1:1 {20}	2:1 {18}	3:3 {14,6,18}	4:1 {18}	5:1 {11}	6:1 {16}
6) GCCTGG-TTTCTG --	1:1 {18}	2:0	3:3 {12,4,16}	4:2 {16,7}	5:2 {9,12}	6:1 {14}
7) ACACGA-CAGAAA --	1:1 {18}	2:0	3:3 {19,13,16}	4:1 {21}	5:1 {10}	6:3 {14,12,7}
8) AGCCTG-TTTCT --	1:1 {19}	2:1 {17}	3:3 {13,5,17}	4:0	5:1 {10}	6:2 {15,8}
9) CAGAAA-TAATAT --	1:0	2:1 {22}	3:3 {22,10,18}	4:1 {22}	5:1 {15}	6:1 {20}
10) AGCCTG-TTTCTG --	1:1 {19}	2:1 {17}	3:3 {13,5,17}	4:1 {17}	5:1 {10}	6:0
Occurrence Agreement(%)	60%	60%	80%	60%	70%	60%

Fig. 6.5: Two-box fragments *occurrence data* output, from modified Exmotif, for the top 10 fragments in the first 6 sequences. The occurrence agreement for each sequences indicates the percentage of fragments that have the same number of occurrences in that sequence.

Our heuristic is to use the occurrence data to infer the actual gap constraints for each fragment. The idea is to find the input sequence that contains the most agreement in the number of occurrences for all fragments. For example, sequence 1 in Figure 6.5 has 60% agreement (i.e., 6 out of the 10 fragments occure the same number of times in sequence 1). This results in choosing sequence 3 as the sequence with the most agreements (80%). Therefore, the gap sizes listed in sequence 3 are the ones used to represent the actual gap sizes for each fragment (e.g., fragment no.4 has actual gap sizes of {10,18,22}). The rationale behind our heuristic is that if a set of fragments repeat in the same sequence the same number of times, then most likely they belong to the same structured motif that was repeated in that sequence.

Now we can use these new discrete gap constraints to find the best multiple alignment to uncover the unknown pattern. SMExtract can handle discrete gap constraints because they are handled internally by SMAlign (Section 4.4). Using the actual gap sizes enhances the resulting multiple alignment as we have eliminated gap sizes that have not actually occurred together. Figure 6.6 shows the difference between two runs of SMExtract using the same input sequences and the same template, but once using the occurrence data (Figure 6.5) and once without. It is clear from the result that using our heuristic improved the results in this case.

Finally, if the sequence with the highest occurrence agreement is missing some fragments (occurrence of zero), then these fragments use their original continuous gaps as given in $\mathcal{T}$. Also, if SMExtract does not find a sequence with occurrence agreement above a user defined threshold (default to 70%), then it falls back to all fragments having the same continous gap as given in $\mathcal{T}$.

```
 7) ACACGA[1,22]CAGAAA
 1) ATTAGC[1,22]TGGCAG
 4) TATTAG[1,22]TTCTGG
 9) CAGAAA[1,22]TAATAT
 3) TGATAT[1,22]TTTCTG
 5) TAGCCT[1,22]TTTCTG
10) AGCCTG[1,22]TTTCTG
 8) AGCCTG[1,22]TTTCT
 6) GCCTGG[1,22]TTTCTG
 2) GCCTGG[1,22]TTTCT
```

(a) The top 10 fragments reported by Exmotif. Notice that we did not use the occurrence data in this example. Each fragment gap constraint is the same as given in the template $\mathcal{T}$.

```
Implanted:  TGATATTAGCCTGG----TTTCTGGCAGA----TCGTGTCCC
            ||||||||||||||    |||||||         ||||||
  Output:   TGATATTAGCCTGGcag-TTTCTGG--------TCGTGT---

r    7)     ------------------TTTCTG---------TCGTGT---
     1)     ----ATTAGC-TGGcag-------------------------
     4)     ---TATTAG----------TTCTGG-----------------
r    9)     --ATATTA----------TTTCTG------------------
     3)     TGATAT------------TTTCTG------------------
     5)     ------TAGCCT------TTTCTG------------------
    10)     -------AGCCTG-----TTTCTG------------------
     8)     -------AGCCTG-----TTTCT-------------------
     6)     --------GCCTGG----TTTCTG------------------
     2)     --------GCCTGG----TTTCT-------------------
```

(b) The Result of SMCluster multi-aligning the top 10 fragments. 'r' indicates that SMCluster used the reverse complement of the reported fragment in the multiple alignment.

```
 7) ACACGA{13,16,19}CAGAAA
 1) ATTAGC{12,20}TGGCAG
 4) TATTAG{10,18,22}TTCTGG
 9) CAGAAA{10,18,22}TAATAT
 3) TGATAT{12,20}TTTCTG
 5) TAGCCT{6,14,18}TTTCTG
 8) AGCCTG{5,13,17}TTTCT
10) AGCCTG{5,13,17}TTTCTG
 6) GCCTGG{4,12,16}TTTCTG
 2) GCCTGG{4,12,16}TTTCT
```

(c) The top 10 fragments reported by Exmotif. We use the occurrence data in this example. Each fragment actual gap constraint is obtained from sequence 3 from the occurrence data (Figure 6.5), as described in this section.

```
Implanted:  TGATATTAGCCTGG----TTTCTGGCAGA--------TCGTGTCCC
            ||||||||||||||    ||||||||||          ||||||
  Output:   TGATATTAGCCTGG----TTTCTGGCAG---------TCGTGT---

r    7)     ------------------TTTCTG-------------TCGTGT---
     1)     ----ATTAGC------------TGGCAG--------------------
     4)     ---TATTAG----------TTCTGG-----------------------
r    9)     --ATATTA----------TTTCTG------------------------
     3)     TGATAT------------TTTCTG------------------------
     5)     ------TAGCCT------TTTCTG------------------------
     8)     -------AGCCTG-----TTTCT-------------------------
    10)     -------AGCCTG-----TTTCTG------------------------
     6)     --------GCCTGG----TTTCTG------------------------
     2)     --------GCCTGG----TTTCT-------------------------
```

(d) The Result of SMCluster multi-aligning the top 10 fragments. 'r' indicates that SMCluster used the reverse complement of the reported fragment in the multiple alignment.

Fig. 6.6: The Result of SMExtract using the template $\mathcal{T} = [4,6] : 0 - [1,22] - [4,6] : 0,\ q_{\mathcal{T}} = 70\%$. Notice that in alignment (d), fragment no.1 was correctly aligned due to the more restrictive actual gap sizes inferred from the data without affecting the correctness of the other fragments' alignment.

6.2.5 SMExtract vs. Exmotif

We now consider the alternative of solving the same example problem using solely Exmotif instead of SMExtract. First, we have to guess a template $\mathcal{T}$ to use with Exmotif. Suppose we use the same default two-box template ($\mathcal{T} = [4,6] : 0 - [1,22] - [4,6] : 0, q_{\mathcal{T}} = 70\%$) as we did with SMExtract, then Exmotif's result would be the one shown in Figure 6.3a which is not very useful as an end result. Therefore, we rerun Exmotif with another blind guess for $\mathcal{T}$ by increasing the number of boxes from two to three ($\mathcal{T} = [4,6] : 0 - [1,22] - [4,6] : 0 - [1,22] - [4,6] : 0$). Exmotif would not return any results this time because our implanted motif has one arbitrary mismatch per box and we only updated the number of boxes for our template and left the template mismatches at zero. We then continue with a third blind run increasing the number of mismatches to 1 mismatch per box (i.e., $\mathcal{T} = [4,6] : 1 - [1,22] - [4,6] : 1 - [1,22] - [4,6] : 1$). This time Exmotif would return part

of the implanted structured motif, as the sizes of the boxes and gaps in this template do not conform to the implanted structured motif. However, even for obtaining this partial result, the time required is much longer than using the template with zero mismatches. This is clearly a frustrating process for a user of Exmotif. Also, the algorithms discussed in Chapter 2 result in similar user experiences. This is in contrast with using SMExtract where the default template gave meaningful results or at least partial results based on which we can make informed updates to our initial template. A qualitative comparison between SMExtract and solely using Exmotif is presented below in Section 6.4.

6.3 Time Complexity of SMExtract

The time complexity of SMExtract is the sum of Exmotif's time complexity (for a two-box template) and SMCluster's time complexity. Recall from Section 2.2.1 that Exmotif's time complexity is $O(kN|\Sigma_{\text{DNA}}|^{kb})$, where $|\Sigma_{\text{DNA}}| = 4$ is the size of the DNA alphabet, $k = 2$ is the number of boxes (SMExtract uses two-box templates regardless of the implanted structured motif), $N = nt$ is the total length of the input sequences, and $b = b^{\text{max}}$ is the size of the larger box between the two boxes in the template. Also, recall from Section 5.3 that SMCluster2's worst time complexity is $O(n^3m^2)$, where n is the number of top fragments to be clustered that are reported by Exmotif and $m = 2b + g^{\text{min}}$ is the length of these fragments. Therefore, SMExtract's worst time complexity is $O(2N|\Sigma_{\text{DNA}}|^{2b} + n^3(2b + g^{\text{min}})^2)$.

If we reasonably assume $(n \leq \sqrt[3]{2N})$ and $(g^{\text{min}} \leq |\Sigma_{\text{DNA}}|^b - 2b)$, then SMExtract's time complexity becomes $O(2N|\Sigma_{\text{DNA}}|^{2b})$. These assumptions are valid in many cases. For instance, in the example in Section 6.2.2 we had $((n = 40) \leq (\sqrt[3]{2N} = 58))$ and $((g^{\text{min}} = 1) \leq (|\Sigma_{\text{DNA}}|^b - 2b = 4^6 - 2 \times 6 = 4084))$. This clearly shows the efficiency of SMExtract over solely using Exmotif for $k > 2$. Also, for $1 \leq k \leq 2$ Exmotif requires more user knowledge than SMExtract about the unknown pattern to formulate an effective template that produces meaningful results (as we have seen in Section 6.2.2).

6.4 Results and Analysis

We implemented both SMAlign and SMCluster using MATLAB and executed the following experiments on a 64-bit Windows 7 machine with Intel Core i5-430UM processor having 1.7GHz max speed with 4GB of RAM. Exmotif was also run on the same machine as a 32-bit process using Cygwin v1.7. We will compare the result of our proposed algorithm SMExtract (Exmotif+SMCluster2) against solely using Exmotif [9] in terms of efficiency, results' quality, and practicality.

6.4.1 Synthetic Datasets

We used SMGenerate to generate biologically relevant synthetic datasets as described in Chapter 3. Table 6.3 shows the characteristics of the generated problems, and comparison of the results provided by SMExtract and Exmotif. Each generated problem consists of $t = 100$ sequences each of size $n = 1000$bp. However, to detect patterns on the complementary strand, we append each sequence with its reverse complement. So, the effective size of each sequence is $n = 2000$bp. Also, we evaluate the quality of the extracted pattern against the implanted pattern using the nucleotide performance coefficient (*nPC*) defined in Equation (6.7). For each generated test case in the table, we used the default template ($\mathcal{T} = [4, 6] : 0 - [1, 22] - [4, 6] : 0, q_{\mathcal{T}} = 70\%$) to show the versatility of SMExtract using one template under various conditions.

Table 6.3: SMExtract($\mathcal{S}, \mathcal{T}$) vs. Exmotif($\mathcal{S}, \mathcal{T}^*$) (Synthatic Datasets)

	Implanted Pattern Parameters							SMExtract($\mathcal{S}, \mathcal{T}$)[1]			Exmotif($\mathcal{S}, \mathcal{T}^*$)[3]		
No.	**Boxes**	**Pattern (M)**	**Instances**	**Per Box Mismatches**	**Quorum$_M$(%)**	**Repeats(%)**	**Complementary Strand(%)**	**Top Clustered**[2]	**nPC(%)**	**Time(s)**	**Rank/Total**[4]	**nPC(%)**	**Time(s)**
1	1	CCTGAAGACTTGG	105	0	79	25	36	30	**100**	**222**	2/2	100	13
2	1	AACACGTACGCC	117	1	82	30	0	18	**100**	**190**	2/2	100	836
3	2	CATCCGCCTT[4,18]AGTTTCATGA	82	0	82	0	0	30	**100**	**232**	2/2	100	13
4	2	AGAGCGACGAGC[4,18]TTCCAAAGGAC	122	0	86	30	47	40	**100**	**273**	1/1	100	13
5	2	GTGAATGTTATGTC[4,18]TTCTTGGCGT	112	1	78	30	0	35	**92**	**236**	1/3	92	275[5]
6	2	GGTGCCCGTGT[4,18]TTCCCAGTG	160	1	100	38	34	35	**79**	**234**	1/1	100	194
7	2	GATAGCGAGTCCAT[4,18]AGCCACACAGCA	111	2	83	25	41	26	**81**	**202**	1/8	85	1065[5]
8	2	AACAGGTGAA[4,18]TGCTCACACTT	140	2	100	29	39	12	**81**	**183**	-	-	>4h[6,7]
9	3	ACCCTCAGTC[4,18]TTCTCCTAGTCTC[4,18]ACCCCCAGCC	84	0	84	0	36	35	**91**	**265**	1/2	100	13
10	3	GCTATCCGAT[4,18]AGTTGTACCGGGAG[4,18]AAGTGCATGTT	100	0	100	0	0	35	**97**	**279**	1/1	100	13
11	3	TTATGACCGGA[4,18]CTTCGCAGTCC[4,18]TCTCAATTCGGT	147	0	100	32	37	35	**100**	**284**	1/1	100	13
12	3	CGTAGGACGCGC[4,18]TCTTGCCAGTA[4,18]TATTCCCAT	88	1	88	0	0	40	**100**	**254**	1/10	84	163[5]
13	3	CGGTAGTCACTCTA[4,18]GTCCCACCCTC[4,18]CTCCCGTCGGGCG	99	1	73	26	41	30	**85**	**227**	1/2	87	182[5]
14	3	CAAAGCACC[4,18]TGAACCTTAGCG[4,18]GGCCGTCTTCC	100	1	100	0	44	30	**80**	**225**	1/22	84	159[5]
15	3	ACGACGGCCT[4,18]TGCTCCTAGTGG[4,18]AAAGAGAGCCGGTA	136	1	100	26	46	40	**81**	**262**	3/34	83	372[5]
16	3	TTTCAAAGGAG[4,18]CCGGTTATTC[4,18]ATGGAGGAT	148	2	100	32	0	22	**77**	**194**	-	-	>4h[6,7]
		Average[7]							**92**	**242**		94	237

[1] SMExtract($\mathcal{T}$)=Exmotif($\mathcal{T}$)+SMCluster2(Top Clustered), where $\mathcal{T} = [4,6] : 0 - [1,22] - [4,6] : 0, q_{\mathcal{T}} = 70\%$ for all test cases.

[2] Tob Clustered=min(X,Total Fragments) where $X \in [30, 40]$.

[3] Exmotif($\mathcal{T}^*$) uses an ideal template $\mathcal{T}^*$, formulated from complete knowledge of M, for each case. For example, for test case no.2, $\mathcal{T}^* = [12,12] : 1, q_{\mathcal{T}^*} = 70\%$, and for test case no.6, $\mathcal{T}^* = [11,11] : 1 - [4,18] - [9,9] : 1, q_{\mathcal{T}^*} = 70\%$, etc.

[4] This is the rank of M in the list of total reported patterns.

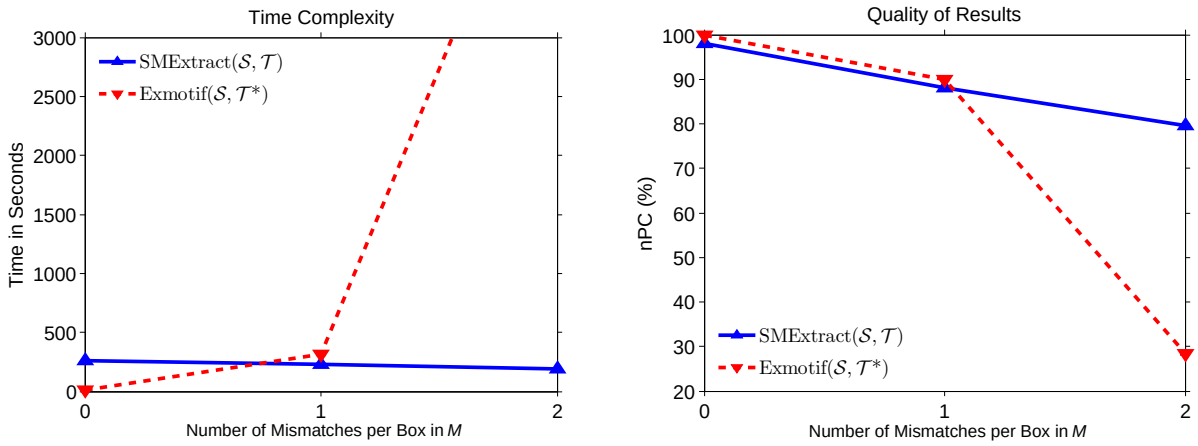
[5] $\mathcal{T}^*$ box size reduced from ideal due to failure of our Exmotif's setup to complete its run using templates with box sizes > 12 (or box range > 0) having mismatches > 0.

[6] The Hamming distance between two instances ≤ 4. Thus, a template with per box mismatches set to 2 completed its run in 2 hours with no results and a template with per box mismatches set to 4 was running beyond 4.5 hours!

[7] Test cases 8 and 16 are not included in the average calculations to give the best chance for Exmotif($\mathcal{T}^*$).

The results clearly show the versatility and efficiency of SMExtract especially when we introduce mismatches in the implanted pattern. Using only the default template $\mathcal{T}$, SMExtract found all structured motifs in the 16 test cases with average quality of 92% and average time of 242s. This is comparable to soley using Exmotif with an ideal template $\mathcal{T}^*$, i.e., using best possible template for each test case, which found 14 of the 16 motifs with average quality of 94% and average time of 237s. Running Exmotif with an ideal template $\mathcal{T}^*$, formulated using our complete knowledge of M, takes longer, in most cases, to extract patterns with mismatches. Furthermore, it is practically unreasonable to assume ideal template formulation for various problems with unknown M. Also, our setup of Exmotif failed to produce results when we used large templates with mismatches (box size > 12 with mismatches > 0), e.g., test cases 8 and 16 in Table 6.3. We suspect that this happens due to the high space complexity of Exmotif to enumerate neighboring patterns when mismatches are specified [9].

SMExtract sacrifices a little efficiency, in the basic non-mismatched cases, for increased versatility and practicality. For the test cases in Table 6.3, SMExtract spent on average 81% of its time executing Exmotif to extract and rank the two-box fragments, and 19% of the time executing SMCluster. Figure 6.7 shows the time complexity and quality of result for SMExtract versus sole Exmotif as we varied the number of mismatches-per-box in the implanted motif M. It is clear to see that SMExtract's processing time is not sensitive to the number of mismatches in the implanted motif as we where successful in extracting M, having mismatches, without specifying any mismatches in our two-box template $\mathcal{T}$. On the other hand, Exmotif requires its ideal template $\mathcal{T}^*$ to have the same number of mismatches as in the implanted motif for successful extraction, which is exponentialy expensive. The same observation is evident in the quality measure *nPC*. SMExtract's *nPC* matches the ideal Exmotif's quality when mismatches at $d \leq 1$, but Exmotif's *nPC* degrades rapidly as the number of mismatches increase $d > 1$. This is because Exmotif did not find two out of three motifs with mismatches-per-box $d > 1$. The ideal Exmotif run took more than four hours without producing any results for the two cases (cases 8 and 16 in Table 6.3), while SMExtract took 183s (*nPC* $= 81\%$) and 194s (*nPC* $= 77\%$) respectively. From this empirical



(a) The time complexity of SMExtract vs. sole Exmotif varying the number of mismatches-per-box of the implanted motif M.

(b) The quality of results for SMExtract vs. sole Exmotif varying the number of mismatches-per-box of the implanted motif M.

Fig. 6.7: The empirical time complexity and quality of result (*nPC*) for SMExtract vs. sole Exmotif, varying the number of mismatches-per-box in the implanted motif M. SMExtract uses one default template $\mathcal{T}$ for all test cases, on the other hand, sole Exmotif uses different ideal template $\mathcal{T}^*$ for each test case.

results, we observed that Exmotif's time and space complexities are practically more sensitive to the number of mismatches in the target pattern than the pattern's number of boxes k. On the other hand, SMExtract novel approach of indirectly extracting the target pattern through its fragments greatly reduces its sensitivity to the number of mismatches in the target pattern.

We have also examined our choice of the number of top fragments to cluster (x) and its effect on the quality of the result (*nPC*). In Section 6.3, we argued that a good choice would be $x \leq \sqrt[3]{2nt}$, where t is the number of input sequences and n is the length of each sequence, to maintain a good balance of time and quality. Figure 6.8 shows the average quality (*nPC*) and time of six test cases (per data point) when we varied the number of top clustered fragments. The figure shows that when $x > \sqrt[3]{2nt}$, the quality of the result does not improve much more to justify the corresponding increase in processing time.

Finally, thanks to the reduced number of user specified parameters, it is easier to tweak SMExtract's parameters to further improve the results in some of the cases presented in Table 6.3 than guessing the full parameters of an ideal template for Exmotif, as discussed in Section 6.2.3.

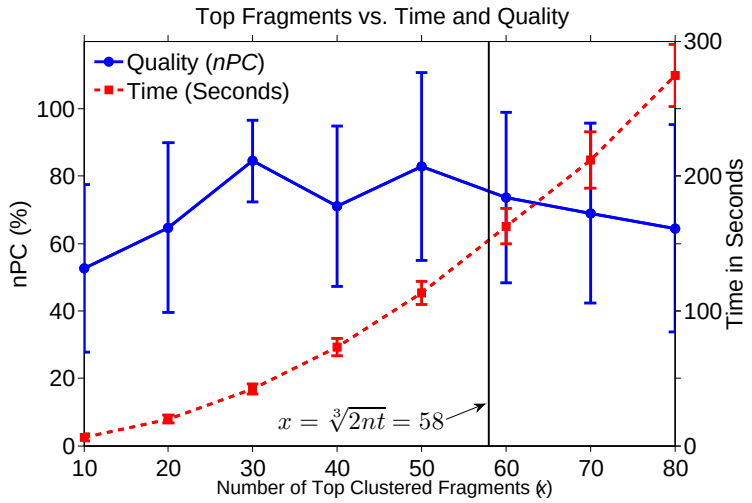


Fig. 6.8: The effect of varying the number of top clustered fragments x on the quality of the extracted motif (*nPC*) and processing time. Choosing $x > \sqrt[3]{2nt}$ did not improve the results quality on average. Each data point is the average of six test cases.

6.4.2 Real Datasets

To predict the transcription factor protein binding sites, we examined the promoter regions for each set of co-regulated genes affected by the same transcription factor in different organisms. We chose data from *Saccharomyces cerevisiae* (SCPD [42]), *Streptomyces coelicolor* (DBSCR [6]), and *Bacillus subtilis* (DBTBS [7]). Table 6.4 shows the transcription factors' (TF) known binding sites, the template parameters used for each case, and SMExtract's predictions. We quantified the quality of the predicted motif compared to the known motif using the nucleotide performance coefficient (*nPC*, Equation (6.7)), the nucleotide positive predictive value (*nPPV*), and the nucleotide sensitivity measure (*nSn*) [8]. The *nPPV* value measures the fraction of the predicted motif that matches the actual known motif, and is defined as:

$$nPPV = \frac{nTP}{nTP + nFP}. \tag{6.8}$$

The *nSn* value measures the fraction of the known motif that was correctly predicted, and is defined as:

$$nSn = \frac{nTP}{nTP + nFN}. \tag{6.9}$$

where *nTP*, *nFP* and *nFN* are the nucleotide true positive, false positive and false negative respectively. Intuitively, an algorithm with high *nPPV* value indicates its ability to predict at least part of the known motif with few or no false positive positions. On the other hand, an algorithm with high *nSn* value indicates its ability to predict large parts of the known motif but may have other false positive positions. The *nPC* measure encapsulates both ideas in one value, but for real datasets, it is more informative to look at all three measures to assess the performance of our algorithm.

The results show promising prediction capabilities even when the template parameters differ significantly from those of the actual known binding site. In some cases, SMExtract picks up some patterns surrounding the actual binding site, but only if these patterns are also overrepresented in the set of promoter sequences. The high *nSn* value indicates that SMExtract is sensitive in picking up large parts of the known motif, but also some other patterns are picked up as well. At the same time, SMExtract's *nPPV* value is not too low indicating that the false positive parts are on average not prominent over the true positive parts.

From both results, synthetic and real, we conclude that SMExtract is more prominent in extracting longer patterns in large sets of sequences, which other algorithms take exponentially longer time to extract.

Table 6.4: Predicting transcription factor protein binding sites in different organisms.

Organism	No.	TF Name	Known Binding Site	Template $\mathcal{T}$ Parameters[1] $[L_M^{\min}, L_M^{\max}]$	$[C_M^{\min}, C_M^{\max}]$	$q_{\mathcal{T}}$	$d_{\mathcal{T}}$	Top[2]	SMExtract Prediction	nPC[3] (%)	$nPPV$[4] (%)	nSn[5] (%)
Saccharomyces cerevisiae	1	GAL4	CGG[11,11]CCG	[10,20]	[3,5]	100	0	20	CCGG[11,11]CCGA	75	75	100
	2	PDR3	TCCGYGGA	[5,10]	[3,5]	70	0	20	TATTTCT[1,1]TCTTTTTCCGCGGA	38	38	100
	3	UASH-URS1H[6]	TTTTGGAGT[10,185]GGCGGCTAA	[10,200]	[4,6]	70	0	20	ATATTTTGAC[1,186]GCGGCT[1,190]CTATTT	38	50	61
Streptomyces coelicolor	4	HrdB	TTGAC[16,18]TAGART	[20,30]	[2,4]	70	0	20	TGACAG[6,16]GGGTACGCA	37	47	64
	5	SigR	GGAAT[17,17]NGTTG	[20,30]	[3,4]	70	0	30	GGAAT[7,7]CCGGC[1,1]CGT[2,2]GTTG[21,21]ACC	75	75	100
Bacillus subtilis	6	SigA	TTGACA[13,21]TATAAT	[20,30]	[2,4]	70	0	30	TTTGACTC[6,6]ATATA	56	69	75
	7	SigB	AGGTTT[12,17]GGGTAT	[20,30]	[2,4]	70	0	30	GTTT[8,23]GGGAT[6,2]GATAA[7,23]CCTA	30	39	58
	8	SigW	TGAAACN[12,13]CGTA	[20,30]	[3,4]	70	0	30	TGAAAC[17,17]CGTAT[13,13]TACG	67	67	100
				Mean						**52**	**57.5**	**82.3**

[1]Template parameters intentionally chosen not to strictly conform to the known binding site to simulate real world test cases.
[2]The number of Exmotif's top reported fragments that were clustered.
[3]*nPC*: The nucleotide performance coefficient, Equation (6.7).
[4]*nPPV*: The positive predictive value, Equation (6.8).
[5]*nSn*: The nucleotide sensitivity measure, Equation (6.9).
[6]Composite site: 2 binding sites for UASH and URS1H TFs that cooperatively regulate 11 yeast genes.

6.5 Discussion

In this chapter, we presented SMExtract, a novel, efficient and versatile structured motif extraction algorithm. Given a set of promoter sequences of co-regulated genes, SMExtract uses an existing algorithm (e.g., Exmotif) to extract overrepresented two-box fragments. Then, SMExtract uses a new clustering method SMCluster (Chapter 5) to assemble together these fragments to construct the target (unknown) pattern. The advantages over other methods, including solely using Exmotif, are in the reduction of the user specified parameters and the flexibility of specifying mismatches while improving the versatility of extracting a wide range of patterns from simple motifs to multi-box structured motifs implicitly. The results clearly show that SMExtract outperforms Exmotif in extracting longer patterns with mismatches in less time and with little to no prior knowledge from the user about the pattern.

Furthermore, we introduced a heuristic to infer the actual gap sizes for the extracted two-box fragments from the data instead of assuming the same gap range for all, as given in the template. This further restricted the number of possible correct multiple alignments which in turn further improved SMExtracts's result quality.

A MATLAB implementation and an online version of SMExtract can be accessed from our website at *http://bioproject.syr.edu/smtools*.

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Summary of Contributions

Motif extraction is a difficult problem. Even after decades of research on simple motif extraction, there is no general agreement on one or two *good* algorithms. Extraction of structured motifs is considerably more difficult due to the lack of knowledge about the motif's number of boxes, their sizes, and the intra-gap sizes. Furthermore, there is a lack of good datasets of annotated real structured motifs, covering many biologically possible cases, to test the performance of a proposed algorithm.

First, we presented an algorithm called SMGenerate that can generate a dataset mimicking real promoter regions of co-regulated genes having common structured motifs. Flexibility in the parameter choices allow a researcher to insert structured motifs with desired specifications to generate specific and realistic test cases. We compared our algorithm with available data generation tools, and we showed that SMGenerate is more flexible and easier to use.

The next major contribution is SMExtract, i.e., an algorithm to extract a structured motif from a given set of promoter regions of co-regulated genes. We proposed an algorithm that achieves this goal with very little knowledge required from the user about the structured motif. We achieved high quality results, using less demanding user prior knowledge, that is comparable to the results of solely

using Exmotif with an ideal template. Most significant contribution is that SMExtract requires user to provide *little* information, to formulate a two-box template, to extract motifs regardless of their type. In other words, SMExtract extracts simple motifs as well as multi-box structured motifs indirectly by only using a two-box template to extract overrepresented fragments and then reconstruct the full desired pattern. We have also shown that SMExtract on average is more efficient than solely using Exmotif with an ideal template especially for extracting motifs with mismatches.

The success of SMExtract relies heavily on the unique features of SMAlign and SMCluster, which provided the capability of correctly clustering together two-box fragments to uncover the desired motif. The extension of SMAlign to handle discrete gap constraints, further enhanced the capability of SMExtract to produce better quality results. Finally, thanks to SMExtract's reduced number of user specified parameters, it is easier to tweak the initial template's parameters, to further improve the results, than trying to guess the full parameters of an ideal template for Exmotif.

MATLAB implementations and online versions of the algorithms discussed in this dissertation can be accessed from our website at *http://bioproject.syr.edu/smtools*.

7.2 Future Research

In the near future, we are planning to address the problem of correctly identifying *multiple* distinct structured motifs that are present in the same set of sequences. When the multiple structured motifs are consistently ordered in the input sequences, i.e., motif M always occur upstream from motif N, then SMExtract can correctly extract both motifs using a two-box template with larger gap range. However if motifs M and N are inconsistently ordered in the input sequences, then a more elaborate solution is required that may involve the interpretation and fusion of multiple runs of SMExtract, using different templates, to correctly extract all motifs. Further research can address this problem more concisely.

SMExtract parameters can be estimated using some type of Monte Carlo optimization techniques. In addition, SMExtract can be extended to benefit from using the location information of extracted

fragments in the input sequences to enhance the multiple alignment quality produced by SMCluster. Also, we can extend the gap constraints to be represented as a distribution of weights over the full possible gap range, as given in the template, such that higher weights are given to gap sizes that have actually occurred in the given input sequences. Furthermore, SMExtract can also benefit from using better methods to extract the two-box fragments replacing Exmotif. Therefore, it is of interest to develop or test other methods that are better than Exmotif at extracting and ranking two-box fragments in terms of efficiency and quality. In addition, further improvements of SMCluster's multiple alignment quality can be achieved using iterative methods [43, 44] to augment the current progressive method. For instance, the initially constructed guide tree from the progressive alignment can be improved using an iterative technique called *leave-one-out* approach [45].

The time and space complexities are important concerns in motif extraction algorithms and if we need to iterate the process several times, then the time complexity becomes even more important. To address the time complexity, parallel algorithm, multi-core, and distributed approaches are needed. Parallel GPU techniques for dynamic programming [46, 47, 48] and branch and bound [49] can be adopted and further developed. This will reduce the impact of the time complexity associated with SMAlign and will reduce the overall time to accomplish the task of motif extraction. To address the space complexity of SMAlign, it may be possible to exploit the same techniques used to reduce the space requirement of Smith-Waterman local alignment and Needleman-Wunsch global alignment algorithms [50, 51].

SMExtract techniques can be used to solve other problems in bioinformatics, such as finding motifs in the untranslated region (UTR) of RNAs to which microRNA binds [52, 53, 54]. This is an interesting problem as microRNA affects the protein translation process, thus genes expression levels. Also, SMExtract can be utilized for the protein multiple alignment and finding of protein domains problems. Finally, other areas can benefit as well from the techniques developed in this dissertation. Namely, adopting the solution to the gap constraint alignment problem in SMAlign to solve scheduling, logistics, and resource allocation problems having known constraints.

REFERENCES

[1] S. Gopal, A. Haake, and R. Jones, *BioInformatics: A Computing Perspective*. McGraw-Hill Higher Education, 2008.

[2] P. D'Haeseleer, "What are DNA sequence motifs?" *Nature Biotechnology*, vol. 24, no. 4, pp. 423–425, 2006.

[3] M. Das and H.-K. Dai, "A survey of DNA motif finding algorithms," *BMC Bioinformatics*, vol. 8, no. S7, p. S21, November 2007.

[4] P. D'Haeseleer, "How does DNA sequence motif discovery work?" *Nature Biotechnology*, vol. 24, no. 8, pp. 959–961, 2006.

[5] W. Wei and X.-D. Yu, "Comparative analysis of regulatory motif discovery tools for transcription factor binding sites," *Genomics, Proteomics & Bioinformatics*, vol. 5, no. 2, pp. 131–142, 2007.

[6] Y. Makita and K. Nakai. (2011, June) DBSCR: Database of transcriptional regulation in streptomyces coelicolor and its closest relatives. [Online]. Available: http://dbscr.hgc.jp

[7] N. Sierro, Y. Makita, M. de Hoon, and K. Nakai, "DBTBS: A database of transcriptional regulation in bacillus subtilis containing upstream intergenic conservation information," *Nucleic Acids Research*, vol. 36, no. suppl 1, pp. D93–D96, 2008. [Online]. Available: http://nar.oxfordjournals.org/content/36/suppl_1/D93.abstract

[8] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. D. Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Régnier,

N. Simonis, S. Sinha, G. Thijs, J. v. Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu, "Assessing computational tools for the discovery of transcription factor binding sites," *Nature Biotechnology*, vol. 23, no. 1, pp. 137–144, January 2005.

[9] Y. Zhang and M. J. Zaki, "EXMOTIF: Efficient structured motif extraction," *Algorithms for Molecular Biology*, vol. 1, no. 21, 2006.

[10] M. Federico, P. Valente, M. Leoncini, M. Montangero, and R. Cavicchioli, "An efficient algorithm for planted structured motif extraction," in *Proceedings of the 1st ACM workshop on Breaking frontiers of computational biology*, ser. CompBio '09. New York, NY, USA: ACM, 2009, pp. 1–6.

[11] X. Liu, D. L. Brutlag, and J. S. Liu, "BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes," *Pacific Symposium on Biocomputing*, pp. 127–138, 2001.

[12] M. Kaya, "Automated extraction of extended structured motifs using multi-objective genetic algorithm," *Expert Systems with Applications*, 2009.

[13] P. Nicolas, A.-S. Tocquet, V. Miele, and F. Muri, "A reversible jump markov chain monte carlo algorithm for bacterial promoter motifs discovery," *Journal of Computational Biology*, vol. 13, no. 3, pp. 651–667, 2006.

[14] C. Eng, C. Asthana, B. Aigle, S. Hergalant, J.-F. Mari, and P. Leblond, "A new data mining approach for the detection of bacterial promoters combining stochastic and combinatorial methods," *Journal of Computational Biology*, vol. 16, no. 9, pp. 1211–1225, 2009.

[15] G. Sandve and F. Drablos, "A survey of motif discovery methods in an integrated framework," *Biology Direct*, vol. 1, no. 1, p. 11, April 2006.

[16] N. Li and M. Tompa, "Analysis of computational approaches for motif discovery," *Algorithms for Molecular Biology*, vol. 1, no. 1, p. 8, May 2006.

[17] T. Kaplan, N. Friedman, and H. Margalit, "Ab initio prediction of transcription factor targets using structural knowledge," *PLoS Comput Biol*, vol. 1, no. 1, p. e1, 2005.

[18] N. Pisanti, A. Carvalho, L. Marsan, and M.-F. Sagot, "RISOTTO: Fast extraction of motifs with mismatches," *7th Latin American Theoratical Informatics Symposium*, 2006.

[19] L. Marsan and M.-F. Sagot, "Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification," *Journal of Computational Biology*, vol. 7, no. 3-4, pp. 345–362, August 2000.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyerivan, "A fast and elitist multiobjective genetic algorithm: NSGA II," *IEEE Transaction on Evolutionary Computation*, vol. 6, pp. 182–197, 2002.

[21] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*, 2nd ed. MIT, 2001.

[22] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195–197, 1981.

[23] S. Schbath and M. Hoebeke, "R'MES: a tool to find motifs with a significantly unexpected frequency in biological sequences," in *Advances in genomic sequence analysis and pattern discovery*, ser. Science, Engineering, and Biology Informatics, L. Elnitski, O. Piontkivska, and L. Welch, Eds., vol. 7, 2011.

[24] M. Dsouza, N. Larsen, and R. Overbeek, "Searching for patterns in genomic data," *Trends Genet*, vol. 13, pp. 597–498, 1997.

[25] E. Blanco, D. Farré, M. Albà, X. Messeguer, and R. Guigó, "Abs: a database of annotated regulatory binding sites from orthologous promoters," *Nucleic Acids Research*, vol. 34, pp. D63–D67, 2006. [Online]. Available: http://genome.crg.es/datasets/abs2005/constructor.html

[26] M. Defrance and J. van Helden, "info-gibbs: a motif discovery algorithm that directly optimizes information content during sampling," *Bioinformatics*, vol. 25, no. 20, pp. 2715–2722, 2009. [Online]. Available: http://rsat.ulb.ac.be/rsat/implant-sites_form.cgi

[27] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Pearson Education, Inc., 2010.

[28] Y. Zhang. (2012, July) What is fasta format? [Online]. Available: http://zhanglab.ccmb.med.umich.edu/FASTA/

[29] S. Needleman and C. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *Journal of molecular biology*, vol. 48, no. 3, pp. 443–453, 1970.

[30] S. Mahony, P. E. Auron, and P. V. Benos, "DNA familial binding profiles made easy: Comparison of various motif alignment and clustering strategies," *PLoS Computational Biology*, vol. 3, no. 3, p. e61, 2007.

[31] J. Clausen, "Branch and bound algorithms principles and examples." Department of Computer Science, University of Copenhagen, March 1999.

[32] W. Zhang, "Branch-and-bound search algorithms and their computational complexity." University of Southern California / Information Sciences Institute, Tech. Rep. ISI/RR-96-443, May 1996.

[33] D. E. Knuth, *The Art of Computer Programming*. Addison-Wesley, 2011, vol. 4A.

[34] R. R. Fenichel, "Algorithms: Algorithm 329: Distribution of indistinguishable objects into distinguishable slots," *Commun. ACM*, vol. 11, no. 6, p. 430, June 1968. [Online]. Available: http://doi.acm.org/10.1145/363347.363390

[35] J. D. Opdyke, "A unified approach to algorithms generating unrestricted and restricted integer compositions and integer partitions," *Journal of Mathematical Modelling and Algorithms*, vol. 9, no. 1, pp. 53–97, 2010.

[36] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," *Journal of Computational Biology*, vol. 1, no. 4, pp. 337–348, Winter 1994.

[37] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Research*, vol. 22, no. 22, pp. 4673–4680, 1994. [Online]. Available: http://nar.oxfordjournals.org/content/22/22/4673.abstract

[38] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: a novel method for fast and accurate multiple sequence alignment," *Journal of Molecular Biology*, vol. 302, no. 1, pp. 205 – 217, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022283600940427

[39] I. M. Wallace, O. O'Sullivan, D. G. Higgins, and C. Notredame, "M-coffee: combining multiple sequence alignment methods with T-coffee," *Nucleic Acids Research*, vol. 34, no. 6, pp. 1692–1699, 2000. [Online]. Available: http://nar.oxfordjournals.org/content/34/6/1692.abstract

[40] A. Jain, M. Murty, and P. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, September 1999.

[41] J. Buhler and M. Tompa, "Finding motifs using random projections," *JOURNAL OF COMPUTATIONAL BIOLOGY*, vol. 9, no. 2, pp. 225–242, 2002.

[42] J. Zhu and M. Q. Zhang, "SCPD: a promoter database of the yeast saccharomyces cerevisiae." *Bioinformatics*, vol. 15, no. 7, pp. 607–611, 1999. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/15/7/607.abstract

[43] M. Hirosawa, Y. Totoki, M. Hoshida, and M. Ishikawa, "Comprehensive study on iterative algorithms of multiple sequence alignment." *Comput Appl Biosci.*, vol. 11, no. 1, pp. 13–18, 1995.

[44] D. W. Mount, "Using iterative methods for global multiple sequence alignment," *Cold Spring Harbor Protocols*, vol. 2009, no. 7, p. pdb.top44, 2009. [Online]. Available: http://cshprotocols.cshlp.org/content/2009/7/pdb.top44.abstract

[45] G. J. Barton and M. J. Sternberg, "A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons," *Journal of Molecular Biology*, vol. 198, no. 2, pp. 327 – 337, 1987. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022283687903160

[46] S. Xiao, A. M. Aji, and W.-c. Feng, "On the robust mapping of dynamic programming onto a graphics processing unit," in *Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems*, ser. ICPADS 09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 26–33. [Online]. Available: http://dx.doi.org/10.1109/ICPADS.2009.110

[47] P. Steffen, R. Giegerich, and M. Giraud, "GPU parallelization of algebraic dynamic programming," in *Proceedings of the 8th international conference on Parallel processing and applied mathematics: Part II*, ser. PPAM 09. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 290–299. [Online]. Available: http://dl.acm.org/citation.cfm?id=1893586.1893623

[48] V. Boyer, D. El Baz, and M. Elkihel, "Dense Dynamic Programming on Multi GPU," in *Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on*, Feb. 2011, pp. 545–551.

[49] A. Boukedjar, M. E. Lalami, and D. El-Baz, "Parallel branch and bound on a CPU-GPU system," *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, vol. 0, pp. 392–398, 2012.

[50] D. S. Hirschberg, “A linear space algorithm for computing maximal common subsequences.” *Comm. ACM*, vol. 18, no. 6, pp. 341–343, 1975.

[51] D. R. Powell, L. Allison, and T. I. Dix, “A versatile divide and conquer technique for optimal string alignment,” *Information Processing Letters*, vol. 70, no. 3, pp. 127–139, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019099000538

[52] H. Iwama, T. Masaki, and S. Kuriyama, “Abundance of microRNA target motifs in the 3’-UTRs of 20527 human genes,” *FEBS Letters*, vol. 581, no. 9, pp. 1805–1810, May 2007.

[53] C. Linhart, Y. Halperin, and R. Shamir, “Transcription factor and microRNA motif discovery: the Amadeus platform and a compendium of metazoan target sets.” *Genome Res.*, vol. 18, no. 7, pp. 1180–1189, 2008.

[54] N. J. Martinez, M. C. Ow, M. I. Barrasa, M. Hammell, R. Sequerra, L. Doucette-Stamm, F. P. Roth, V. R. Ambros, and A. J. Walhout, “A C. elegans genome-scale microRNA network contains composite feedback motifs with high flux capacity,” *Genes & Dev.*, vol. 22, pp. 2535–2549, 2008.

VITA

NAME OF AUTHOR: Faisal Abdulmalek Alobaid

PLACE OF BIRTH: The State of Kuwait

DATE OF BIRTH: August 20^{th}, 1977

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

- Syracuse University, Syracuse, NY, USA. 2008-2012.
- University of Southern California, Los Angeles, CA, USA. 2004-2005.
- University of Massachusetts, Dartmouth, MA, USA. 1996-2000.

DEGREES AWARDED:

- MSc in Computer Science, 2005, University of Southern California, Los Angeles, USA.
- BSc in Computer Engineering, 2000, University of Massachusetts, Dartmouth, USA.

PROFESSIONAL EXPERIENCE:

- Faculty Member, Computer Science Dept., Public Authority for Applied Education and Training (PAAET), Kuwait 2004-Present.
- System Analyst, IT Dept., Kuwait National Petroleum Company (KNPC), Kuwait 2000-2004.

PUBLICATIONS:

- F. Alobaid, K. Mehrotra, C. Mohan, and R. Raina. "SMAlign: Alignment of DNA Sequences with Gap constraints." In *The Proceedings of the ISCA 4th International Conference on Bioinformatics and Computational Biology*, BICoB 2012, pages 43-50, Las Vegas, NV, USA, March 2012. ISCA.

AWARDS AND ACHIEVEMENTS:

- 2012: Earned my PhD Degree in Computer Science from Syracuse University.
- 2012: Nunan Research Poster Competition Grand Prize Recipient, Syracuse University.
- 2010: Golden Key International Honor Society for Outstanding Scholarly Achievement.
- 2009: Phi Kappa Phi Honor Society for Outstanding Scholarly Achievement.
- 2009: Phi Beta Delta International Honor Society for Outstanding Scholarly Achievement.
- 2008: PAAET Full Scholarship to Earn my PhD Degree in Computer Science.
- 2002-2006: Organizing Committee Member in HH Sheikh Salem AlSabah Informatics Award.
- 2006: Head of Technical Committee in the 1^{st} Arab Robotics Contest.
- 2006: Invited Guest Speaker - Kuwait Robotics Center.
- 2005: Earned my MSc Degree in Computer Science from USC.
- 2004: PAAET Full Scholarship to Earn my MSc Degree in Computer Science.
- 2004: Awqaf Technological Contest, 1^{st} Prize Recipient, Kuwait.
- 2003: International Robotics Contest - Eurobot 2003, France.
- 2003: Expo Science International - ESI 2003, Moscow, Russia.
- 2003: Invited Guest Speaker - Annual Conference, Creativity Throughout Generations.
- 2002: Kuwait University Award Recipient for Robotics Workshops.
- 2002: Kuwait Ministry of Education Award Recipient for Robotics Workshops.
- 2002: Volunteer Trainer in Kuwait Robotics Center: Seminars on Robotics and Engineering.
- 2002: Advisor and Tutor - Kuwait University Robotics-Related Senior Design Projects.
- 2002: Founder of the 1^{st} Robotics Center in Kuwait and the Region.
- 2001: Invited IT Speaker (Portals and Web-Based Application Development) - KITS.
- 2000-2001: Trinity College Fire-Fighting Robotic Contest, CT, USA.