6-2019

Approximate Algorithms for Regulatory Motif Discovery in DNA

Hasnaa Imad Al-Shaikhli
Western Michigan University, hasnaaimad@gmail.com

APPROXIMATE ALGORITHMS FOR REGULATORY MOTIF DISCOVERY IN DNA

by

Hasnaa Imad Al-Shaikhli

A dissertation submitted to the Graduate College
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
Computer Science
Western Michigan University
June 2019

Doctoral Committee:

Dr. Elise de Doncker, Chair
Dr. Alvis Fong
Dr. Todd Barkman
Dr. Nancy Deng

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my committee Chair, Professor Dr. Elise de Doncker. Without her guidance and persistent help, this dissertation would not have been possible. I learned from her how to define a research problem, find a solution to it, and finally publish the results.

Besides my advisor, I sincerely thank the other members of my dissertation committee members, Dr. Alvis Fong, Dr. Todd Barkman, and Dr. Nancy Deng for their support and advice on problems I encountered during my research.

In addition, my gratitude goes to the Department of Computer Science Chair, Dr. Steven Carr, and staff for the last minute favors.

I also thank my sponsor, the Higher Committee for Education Development in Iraq (HCED), for their financial support, without I which would not have been able to study at Western Michigan University.

Last but not least, I want to express my deepest gratitude to my family and friends. This dissertation would not have been possible without their warm love, continued patience and endless support.

Hasnaa Imad Al-Shaikhli

APPROXIMATE ALGORITHMS FOR REGULATORY MOTIF DISCOVERY IN DNA

Hasnaa Imad Al-Shaikhli, Ph.D.

Western Michigan University, 2019

Motif discovery is the problem of finding common substrings within a set of biological strings. Therefore it can be applied to finding Transcription Factor Binding Sites (TFBS) that have common patterns (motifs). A transcription factor molecule can bind to multiple binding sites in the promoter region of different genes to make these genes co-regulating. The Planted (l, d) Motif Problem (PMP) is a classic version of motif discovery where l is the motif length and d represents the maximum allowed mutation distance. The quorum Planted (l, d, q) Motif Problem (qPMP) is a version of PMP where the motif of length l occurs in at least q percent of the sequences with up to d mismatches. In this thesis we develop the *Strong Motif Finder (SMF)* and *quorum Strong Motif Finder (qSMF)* algorithms and evaluate their performance.

The *Strong Motif Finder (SMF)* returns a list of its highest ranked (strongest) motifs. The performance of SMF is compared with the APMotif and MEME algorithms with respect to execution time and prediction accuracy. Several performance metrics are used at both the nucleotide and the site level. The algorithms are tested on simulated datasets. The time comparisons show that SMF is faster than the APMotif and the MEME (ANR) and similar in speed to the MEME (ZOOPS). The MEME algorithm with choice OOPS is the fastest but is not practical if no prior knowledge is available. The prediction accuracy results reveal that the SMF outperforms the APMotif, and performs at the level of the best prediction accuracy of the MEME (with OOPS choice), notwithstanding that the SMF is not given

a-priori information. In addition, the SMF is tested on real DNA datasets of orthologous regularity regions from multiple species, without using their related phylogenetic tree. The experiments indicate that the SMF results agree with published motifs.

The *quorum Strong Motif Finder (qSMF)* returns a list of highest ranked (strongest) motifs occurring in at least q percent of the data sequences. The algorithm is tested on ChIP-Seq (large) data that was sampled using the SamSelect algorithm. In comparison with the FMotif algorithm, the experimental results show that qSMF is faster and returns predicted motifs similar to results in the literature and to motifs discovered by the ENCODE project tool which uses the established motif finding algorithms of AlignACE, MEME, MDscan, Trawler, and Weeder.

In order to determine the strength or the significance of the predicted motifs, a scoring function, the *Motif Strength Score (MSS)*, is proposed for ranking the discovered motifs in both algorithms. In future work, this score can be combined with other statistical scores, such as the complexity score, P-value and information content, to better determine the motif significance.

TABLE OF CONTENTS

Table of Contents—Continued

Table of Contents—Continued

Table of Contents—Continued

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A	**A**denine
AlignACE	**Align**s Nucleic **A**cid **C**onserved **E**lements
ANR	**A**ny **N**umber of **R**epetitions
AP	**A**ffinity **P**ropagation
ASP	**A**verage **S**ite **P**erformance
bp	**b**ase **p**air
C	**C**ytosine
CC	**C**orrelation **C**oefficient
ChIP	**Ch**romatine **I**mmuno**P**recipitation
CS	**C**omplexity **S**core
ChIP-Seq	**Ch**romatine **I**mmuno**P**recipitation **S**equencing
CS	**C**omplexity **S**core
DNA	**D**eoxyribo**N**ucleic **A**cid
EdMP	**Ed**ited **M**otif **P**roblem
EM	**E**xpectation **M**aximization
ExMP	**Ex**tended (l, d) **M**otif **P**roblem
FN	**F**alse **N**egative
FP	**F**alse **P**ositive
G	**G**uanine
IC	**I**nformation **C**ontent
i.i.d.	**i**ndependent and **i**dentically **d**istributed
IUPAC	**I**nternational **U**nion of **P**ure and **A**pplied **C**hemistry

MCAT	**M**otif **C**ombining **A**ssociation **T**ool
MEME	**M**ultiple **E**M for **M**otif **E**licitation
MDscan	**M**otif **D**iscovery scan
MFMD	**M**emetic **F**ramework **M**otif **D**iscovery
MoD	**Mo**tif **D**iscovery
mRNA	**m**essenger **R**ibo**N**ucleic **A**cid
MSS	**M**otif **S**trength **S**core
NP	**N**ondeterministic **P**olynomial
OOPS	**O**ne **O**ccurrence **P**er **S**equence
PC	**P**erformance **C**oefficient
PFM	**P**osition **F**requency **M**atrix
PMP	**P**lanted (l, d) **M**otif **P**roblem
PPV	**P**ositive **P**redictive **V**alue
PSSM	**P**osition **S**pecific **S**coring **M**atrix
PWM	**P**osition **W**eight **M**atrix
qPMP	**q**uorum **P**lanted (l, d) **M**otif **P**roblem
qSMF	**q**uorum **S**trong **M**otif **F**inder
R	pu**R**ine
RNA	**R**ibo**N**ucleic **A**cid
S	**S**trong
Sn	**Sn**sitivity
SMC	**S**imple **M**atching **C**oefficient
SMF	**S**trong **M**otif **F**inder
SMP	**S**imple **M**otif **P**roblem
SP	**SP**ecifficity

SS	**S**elected **S**equence
T	**T**hymine
TBFS	**T**ranscription **F**actor **B**inding **S**ite
TCM	**T**wo **C**omponent **M**ixture
TF	**T**ranscription **F**actor
Tmod	**T**oolbox of **mo**tif **d**iscovery
TN	**T**rue **N**egative
TP	**T**rue **P**ositive
tRNA	**t**ransfer **R**ibo**N**ucleic **A**cid
TSS	**T**ranscription **S**tart **S**ite
W	**W**eak
Y	p**Y**rimidine
YMF	**Y**east **M**otif **F**inder
ZOOPS	**Z**ero or **O**ne **O**ccurrence **P**er **S**equence

CHAPTER 1

INTRODUCTION

This chapter gives a brief overview as biological background of DNA and other related information, followed by a general description of the motif discovery problem, its various versions and categories. The importance of the problem and its applications are also highlighted.

1.1. Biological Background

This section presents a brief description of biological molecules, genome, genes, DNA structure, and the central dogma of molecular biology, as these underlie the work on motif discovery.

1.1.1. What is the Basis of Life?

In 1665, Robert Hooke discovered that organisms are composed of cells. Cell theory was further advanced by Matthias Schleiden and Theodor Schwann in 1830 when the study of life became the study of cells [1]. Living organisms can be classified into classes, prokaryotes and eukaryotes, on the basis of cell structure. The cells of prokaryotes do not have a nucleus; therefore, DNA floats loosely in the liquid center of the cell, while the cells of eukaryote organisms have a nucleus that encapsulates the DNA [1], [2].

The three primary types of molecule upon which life depends are DNA, RNA, and proteins. DNA is considered a huge library describing how cells work. RNA enables the transferral of short pieces of DNA to various places in the cell. Those small pieces of information

are used as templates to synthesize proteins. Proteins form enzymes that perform biochemical reactions, send signals to other cells, form major body components such as skin keratin, and control the real work of the cells. DNA, RNA, and proteins are instances of long strings written either in the four-letter alphabet {A, C, G, T or U} for DNA and RNA or the twenty-letter alphabet {A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V} for proteins [1].

1.1.2. What is a Genome?

The German botanist Hans Winkler introduced the term *genome* in 1920 to refer to the complete genetic material of an organism. It contains all the information of heredity encoded in the DNA and packaged in the chromosomes. The genome involves both genes as coding sequences and non-coding sequences in the DNA. More specifically, the genome of an organism is a complete DNA sequence of one set of chromosomes. The total length of the human nuclear genome is 3×10^9 base pairs (bp) [3]. In short, genomes are comprised genes that hold the DNA coding information.

1.1.3. What are Genes?

Genes are discrete pieces located on DNA chromosomes that code for proteins. A current estimate for the number of genes in a human is around 25,000 that are encoded on 23 chromosomes. A eukaryotic gene consists of protein-coding segments called exons separated by non-coding segments called introns. The size and number of introns vary in different genes [3]. Genes are important pieces located on DNA.

1.1.4. What is DNA?

The DeoxyriboNucleic Acid (DNA) molecule consists of two complementary chains that are twisted together to form a double helix [3]. All of the Earth's living cells store their heredity information in the form of double-stranded molecules of DNA, which can be described as a long linear structure that encodes genetic information [2].

DNA chemically consists with a sugar, a phosphate group, and one of four nitrogenous bases of adenine (A), thymine (T), guanine (G), or cytosine (C) [1]. Cytosine always pairs with guanine while adenine always pairs with thymine. Triple hydrogen bonds exist between every C:G pair while two hydrogen bonds occur between each A:T pair [3].

1.1.5. DNA Discovery Story

In 1869, Johann Friedrich Miecher discovered DNA when he separated a substance he named "nuclein" from the nuclei of white blood cells. By the early 1900s, DNA or nuclein was considered as a repetitive sequence of the four bases, A, T, G, C that formed a long molecule. At the beginning of the 1950s, the modern DNA era started. In 1950, Erwin Chargaff discovered the one-to-one ratio of the A-to-T and G-to-C content in DNA (known as Chargaff's first rule). In 1951, Maurice Wilkins and Rosalind Franklin obtained X-ray pictures of DNA, which suggested that DNA is a helical molecule. In 1953, James Watson and Francis Crick determined the double helix structure of DNA [1]. These initial discoveries lead to more questions and investigations.

1.1.6. Central Dogma

The central dogma of molecular biology was first phrased by Francis Crick in 1958, and represents a framework for how the sequence information transfers from DNA to the

generation of proteins [1], [3]. Figure 1.1 shows the flow of information beginning with a DNA template strand. The information on this strand is transferred to RNA during the transcription stage with the help of the RNA polymerase enzyme that synthesizes messenger RNA (mRNA). The single stranded RNA is translated by large molecular complexes called ribosomes to synthesize a poly-peptide chain, which folds into a protein [3].

In the translation phase, the ribosomes read consecutive codons (triplets of nucleotides) and locate their corresponding amino acids, which will be included in the poly-peptide with the help of transfer RNA (tRNA). The tRNA has a triplet base segment (anticodon) that is complementary and binds to the codon on the RNA. Each of the (20) amino acids binds to one of the (20) types of tRNA. The amino acid is then added to the poly-peptide [1]. The central dogma is the general process for gene expression.

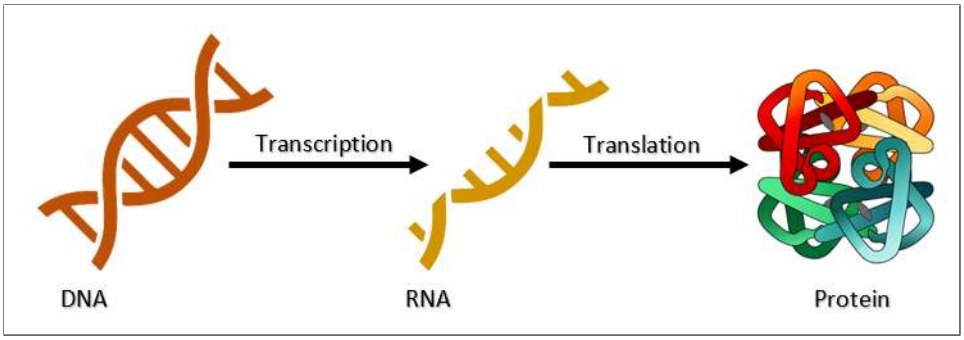


Figure 1.1: Central Dogma Process Illustration

1.1.7. What is Gene Expression?

The primary unit of inherited information in DNA is the gene, consisting of a segment that is used as a template for the transcription (copying) process. Every gene contains the necessary information to produce a protein. Gene expression starts when multiple protein factors, known as transcription factors, are bound to enhancer and promoter sequences.

Transcription factors regulate the gene expression by activating or inhibiting the transcription machinery [4].

The regulatory region, where the motifs are located, is the promoter region and is located upstream of the coding sequence or gene as illustrated in Figure 1.2. The promoter region contains information about the cell status. The transcription level is adjusted according to this information [5].

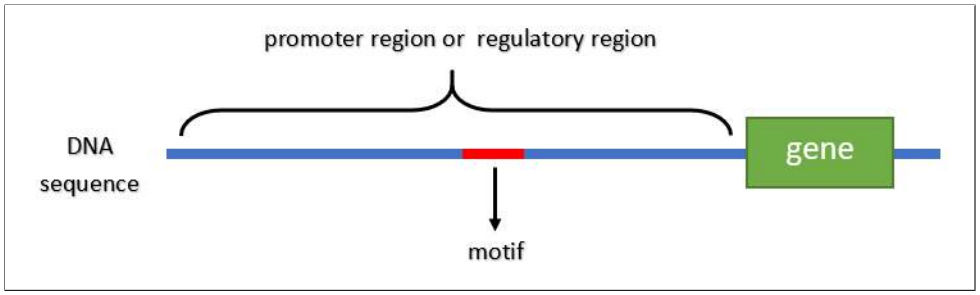


Figure 1.2: Motif Location with its Corresponding Gene in the Promoter Region of a DNA Sequence

1.1.8. What are Motifs?

Motifs are short (in the range 5-25 bp [6], 5-20 bp [4], 8-20 bp [7]), recurring substrings that are presumed to have a biological function. Usually, they indicate sequence specific binding sites for proteins such as Transcription Factors (TFs). These binding sites are called Transcription Factor Binding Sites (TFBS) [4], [5], [8]. In other words, motifs are short conserved regions in the non-coding parts of DNA sequences. The interaction between Transcription Factors (TF) and their binding site is considered an initial step in the transcription initiation of genes [9]. The gene expression process starts when a transcription factor molecule starts to bind to a short substring in the promoter region of the gene. A transcription factor can bind to multiple binding sites in the promoter region of different genes to make these genes co-regulate. These binding sites should have common patterns

(motifs), and the goal of the motif discovery problem is to find these patterns [10]. Motifs can occur on both DNA strands, and sequences may have zero, one, or multiple instances of a motif [4].

There are several ways to represent a motif [9], such as consensus strings, degenerate consensus strings in International Union of Pure and Applied Chemistry (IUPAC) notations, position frequency matrices (PFMs), position weight matrices (PWMs), or position specific scoring matrices (PSSMs). (Further information about PSSM can be found in [11].) After motifs are collected and aligned through experimental or computational procedures, a consensus IUPAC string can be generated by selecting a degeneracy base pair symbol for each position in the alignment [12]. In addition, this alignment can be modeled as a PFM by counting the frequency of each base pair at each position. This matrix is also called a profile matrix. Usually, a PFM is converted into a PWM or PSSM representation [12], [13]. Further, PWM can be displayed as logos [14], with color and height proportional to the base pair frequency and information content for each position. Locating motifs is the goal of the motif discovery problem.

1.1.9. What are Mutations?

During both the storing and copying of genetic information processes, random accidents and errors occur, which create mutations that alter the nucleotide sequence. For this reason, when a cell divides, its two daughters are often not identical to one another or to their parent [2].

Section 1.1 reviews the simple biological background that is necessary for understanding the motif discovery problem because the motif discovery must take into consideration where motifs can be located in DNA sequences. In addition, there are elements such as mutations which can complicate motif discovery.

1.2. Motif Discovery Problem Description

A pattern discovery problem can be simply formulated as follows: given a set of sequences, find an unknown pattern that occurs frequently. If a pattern m of length l is known to appear exactly in each sequence, then the problem can be solved by enumerating all $l-$letter patterns. However, working with DNA sequences complicates the problem since patterns may include mutations, insertions or deletions of nucleotides [4].

The problem of finding meaningful patterns (i.e. motifs) from biological data has been studied widely in view of its significance [15]. The DNA motif discovery problem is simply finding short conserved sites in genome DNA sequences [16], which is considered a major problem in computer science and molecular biology [17] [18]. Tracking and exploring motifs, not just in DNA but also in other biological sequences such as RNA and proteins, help biologists understand, learn, and discover the functions of these sequences. Despite research efforts, this problem is challenging for computer scientists as the general version is considered NP-hard [17].

1.3. Motif Discovery Problem Versions

Several versions of the motif finding problem can be found in the literature [15], [17]:

1. **Planted (l, d) Motif Problem (PMP)**

   Given are a set $S = \{S_1, S_2, ..., S_t\}$ of t sequences each of length n over an alphabet Σ, $|S_1| = |S_2| = ... = |S_t| = n$, and two integers l and d, $0 \leq d < l < n$, where each of the t sequences is assumed to contain an implanted variant of a consensus motif M of length l. A variant of M is a string at Hamming distance $\leq d$ from M. The Hamming distance between two strings is the number of locations where they differ. The PMP problem attempts to find one or more consensus motifs within distance d.

PMP emerges in molecular biology from the search for the transcription factor binding sites in genomic sequences. Studies of the planted motif problem include those by Sagot [19] in 1998, Pevzner and Sze [20] in 2000, and Buhler and Tompa [16] in 2002.

A specific version of PMP is the quorum Planted (l, d, q) Motif Problem (qPMP), where the motif must occur in at least the quorum constraint (q) sequences where $1 \leq q \leq t$ [19] (or, alternatively, in $q\%$ of the given sequences). Given a set of t sequences, and motif length l with the allowed mutation d, the algorithm is tasked with finding all the patterns of M of length l that occur in at least q sequences out of t (or $q\%$ of the sequences). Thus the qPMP problem is the same as PMP when $q = t$ [21].

2. **Edited Motif Problem (EdMP)**

   Consider the set $S = \{S_1, S_2, ..., S_t\}$ of sequences of average length L over the alphabet σ, and three integers l, d and q. The problem consists of finding all substrings, or edited motifs, of length l in S such that each substring has at least q edited variants in at least q different sequences of S. A substring u is considered an edited variant of a substring v if the edit distance $D_{\sigma,\gamma,\delta}(u, v) \leq d$. The edit distance $D_{\sigma,\gamma,\delta}$ is the minimum number of edit operations, i.e. change (σ), insert (γ), and delete (δ) that change u to v.

3. **Simple Motif Problem (SMP)**

   A simple motif pattern consists of a string of symbols from an alphabet $\Sigma \cup \{?\}$, where the pattern cannot begin or end with ?. Here "?" refers to a wild-card character and can be replaced with any character from $\sum$, for example, AB?D and EB??DS?R. The length of the simple motif is the number of symbols including ?. A class of simple motifs of length p with wild card character(s) (q) is denoted as a $(p, q)-$class. The input for this problem version is a set of sequences $S = \{S_1, S_2, ..., S_t\}$ over a given

alphabet and an integer $l > 0$. The output contains all simple motifs of length that are $\leq l$ with a number (ranging from 0 to $\lfloor l/2 \rfloor$) of wild card characters q, including how many times each simple motif occurs in the sequences of S.

4. **Extended (l, d) Motif Problem (ExMP)**

   When given the set $S = \{S_1, S_2, ..., S_t\}$ of sequences over alphabet Σ, $|S_1| = |S_2| = ... = |S_t| = n$, with two integers l and d, $0 \leq d < l < n$, the goal of this problem version is to find a string M such that there exist at least k substrings $M_1, M_2, ..., M_k$, $|M_i| = |M|$, $1 \leq i \leq k$ in the sequences of S. If any substring M_i differs from M in at the most d positions over any window of l characters where $l \leq |M|$, the substring M is called an extended (l, d) motif, and the substrings M_i are the extended (l, d) variants of M. This version is defined to address two defects in the PMP version. First, it is rare to get a set of sequences where each sequence contains a variant of the motif. Second, the exact length of the motif is not known by biologists. At best a range of lengths is known [17].

Even though there are four versions of the motif discovery problem, each version takes into consideration different aspects of the problem. The PMP version is the most standard and investigated version.

1.4. Motif Search Computational Algorithms Categories

In the literature, the algorithms that focus on solving the motif finding problem can be categorized as follows:

1. **Exact or Approximate Algorithms**

   Algorithms that always output the correct solutions (implanted motifs) are called exact algorithms. These coincide with exhaustive algorithms, while algorithms that may not

always output the correct solutions are approximate algorithms which, in view of the use of heuristics, also fall under heuristic algorithms. There are many exact algorithms in literature, but they have become impractical due to increasing problem sizes, which render them extremely time consuming [15].

2. **Knowledge-Based or De Novo Algorithms**

   Motif discovery algorithms that use prior information about binding sites or nucleotide patterns, are situated within a knowledge-based category, while de novo algorithms discover motifs purely from a set of sequences without any prior knowledge [21].

3. **Profile-Based or Pattern-Based Algorithms**

   Algorithms can also be categorized depending on their profile-based (alignment-based) or pattern-based (consensus-based) approach [15], [17], [22]. The profile-based approach predicts the starting positions of motifs in each sequence, while the pattern-based approach predicts the motif itself as a sequence of residues [15], [17].

4. **Word-Based or Probability-Based Algorithms**

   Word-based (or string-based) techniques rely on counting and comparing oligonucleotide frequencies, whereas probability-based methods estimate the model parameters using a maximum-likelihood principle or Bayesian inference. Methods belonging to the word-based class are fast and guarantee global optimality but suffer from generating too many spurious (false) motifs. The probability-based methods represent the motif instance as a position weight matrix. Probability-based methods require fewer search parameters, but they suffer from regularity region sensitivity [4].

5. **Single Species, Many Genes; Single Gene, Many Species; or Both**

This classification of motif finding algorithms is based on the type of DNA information utilized by the algorithm. In the single species and many gene cases, a set of regularity region sequences (i.e., promoters) from co-regulated genes of a single genome are analyzed by looking for over-represented motifs or TFBS occurrences that are responsible for co-regulating these genes. In the single gene and many species case, which is known as phylogenetic footprinting, a single gene is investigated while multiple non-coding (promoter) sequences are compared to their homologous promoters in other species. The two approaches can be merged, and each set of co-regulated genes can be compared to its homologous genes from multiple species and to the other genes from the same species [4], [23].

1.5. Importance of the Motif Finding Problem

Understanding the mechanisms that regulate gene expression is a fundamental challenge in biology. Identifying regulatory elements, especially the binding sites for transcription factors in DNA, is a major task in this challenge. DNA pattern discovery is one of the most challenging problems in molecular biology and computer science [4] [24].

The motif finding problem goal is to identify substrings that are more or less conserved in the given data. This problem is fundamental for both biologists and computer scientists. Extracted motifs help biologists track and explore challenging questions involving the functions of biological sequences and the mechanisms where these sequences play a role. Although much research has been done, this problem is still a challenge for computer scientists since its general version is NP-hard. Furthermore, incomplete knowledge of the biological mechanisms hinders computer scientists in the development of efficient models to solve such problems [17].

Determining motifs through direct biological experiments is not cost-effective and is impractical for many biological systems. Thus, further development of computational motif discovery techniques is necessary and essential in studies involving gene regulation [25].

1.6. Summary

In bioinformatics, the classic version of the motif finding problem is PMP due to its importance in identifying meaningful patterns in biological sequences [26]. In the literature and current research, various algorithms are presented to solve the PMP problem. In this dissertation, two approximate algorithms are proposed, SMF and qSMF. The SMF (Strong Motif Finder) algorithm addresses PMP applied to relatively small DNA datasets, while the qSMF (quorum Strong Motif Finder) algorithm is intended to solve qPMP for much larger DNA datasets. Our main goals with both algorithms are to reduce the execution time while achieving equal or higher prediction accuracy.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

Gene expression is controlled or regulated by transcription factors (TFs) when they start binding to specific transcription factor binding sites (TFBSs) within regulatory regions associated with the genes. The identification of the binding site locations is an initial step for understanding gene regulation. Experimental identification and verification of these segments are challenging. Therefore, many efforts have gone into developing computational approaches to solve such a problem. Good computational methods can potentially achieve high quality prediction accuracy of the binding sites and decrease the time needed for verification. However, the computational approaches become as challenging as the experimental verification. Therefore, many different methods have been developed [5].

The problem of motif discovery has been studied widely. The various methods with different approaches introduced so far differ primarily in three points. The first is how similar motif instances (oligos) that form a candidate motif are selected. The second is how the over representation (statistical significance) of these motifs is measured. Finally, the background or random model employed must be determined [7].

In 1977, Korn et al. [27] attempted to solve the motif discovery problem in DNA sequences. They were able to discover sequence similarities in regions that are immediately upstream from the transcription start site (TSS) by considering mismatches and gaps. Improvements [28] followed in 1982 when multiple sequences were compared simultaneously. Exact requirements of motifs were defined clearly with quorum constraints on sequence

support, maximum number of mismatches in motif occurrences, and maximum distances between occurrence positions in the sequences. In the past 30 years, a large number of algorithms and tools have been developed to solve the motif discovery problem with varying success.

This chapter gives a quick overview of the complexity of the motif finding problem, the main scoring functions used to assess motif significance, the most recent algorithms that can be applied on large datasets, some online tools that can be useful for algorithm developers, the online databases that can be used to assess algorithm performance, the recent research direction towards ensemble methods, and how algorithm performance is assessed. Finally, clarification of the limitations and challenges in the computational methods is examined. All information included may help researchers when working on the motif finding problem.

2.2. Motif Finding Problem Complexity

A canonical representation of the motif problem was given by Li et al. in [29], and it was demonstrated to be NP-hard even with simplified assumptions [30]. Therefore, the current algorithms in the literature concentrate on enhancing the average performance to solve challenging instances within an acceptable time [26].

When given t sequences of length n and motif size l, and assuming that a motif instance should appear in each sequence, there are $(n - l + 1)^t$ candidate solutions. Therefore, exhaustive enumeration of the solution space requires exponential time and is computationally impractical. As a result, employing heuristic techniques is a must to solve this problem when using profile-based algorithms [7].

In consensus-based algorithms, the problem is formulated in a different way because for each of 4^l substrings of length l, the algorithm must collect all its approximate occurrences with up to d mismatches from the input sequences. From the occurrences, the consensus

string is generated and its significance is calculated. The problem here becomes an exhaustive approximate pattern matching problem. The methods under this category consume too much time, especially with longer motifs. By applying indexing structures to input sequences, the approach of [7] reduces the theoretical complexity from 4^l to 4^d (exponential in the number of mismatches).

2.3. Mostly Used Scoring Functions

Computational motif discovery is possible because the motifs are statistically over-represented. Many methods the over-representation criteria into account when evaluating the significance of a discovered motif [5]. A statistically over represented motif is a pattern that occurs more often than one would expect by chance [4]. Mentioned below are several computational (statistical) measures that have been used to calculate the significance or the quality of these motifs.

2.3.1. Information Content (IC)

Information content (IC) or relative entropy is a statistical measure that was defined in [31]. It measures the statistical difference between a motif from a specific probabilistic model and a motif from a probabilistic background model. In other words, the information content can be used to calculate the overall motif conservation and its distance from a background random distribution, if it is assumed that the sequence nucleotides are independent. IC can be calculated using [7], [11], [30]

$$IC = \sum_{i=1}^{4} \sum_{j=1}^{l} m_{i,j} log_2 \frac{m_{i,j}}{b_i} \tag{2.1}$$

where $m_{i,j}$ is the entry at row index i and column index j of the profile matrix, l is the motif length, and b_i is the expected frequency of nucleotide i in the input sequences. Therefore, the b_i can be derived from the genomic sequence of the organism studied or from the input sequences themselves [7].

Relative entropy is used to design sequence logos [14]. The height of each nucleotide in each position is proportional to its entropy value. For example, most conserved nucleotides have an entropy of 2 bits [7].

2.3.2. Complexity Score (CS)

The complexity score (CS) penalizes the sequences with low complexity. In other words, the sequences with low entropy, since these sequences may interfere with the search and should be considered as noise [30], [32]. Wootton and Federhen [33] define the complexity score as

$$CS_1 = \frac{1}{l} \log_N \left(\frac{l!}{\prod_{i=1}^{N} n_i!} \right) \tag{2.2}$$

where $N = 4$ for DNA and n_i denotes the total number of nucleotides present in a string of length l of type $i \in \{A, C, G, T\}$. They introduce another informational measure of complexity; CS_2 in Eq. (2.3). This measure is expressed in bits since the logarithm is taken to base 2.

$$CS_2 = -\sum_{i=1}^{N} \frac{n_i}{l} \log_2 \left(\frac{n_i}{l} \right) \tag{2.3}$$

2.3.3. P-Value

The most direct approach to determine over-representation is by comparing the discovered motif score with the expected scores from a background model. The p-value [5] can be defined as the probability that an event occurs by chance. The range of p value is between 0 and

1 [34]. If the p-value is close to zero, then the word or motif is highly significant. If it is close to one, then the word or motif is rarely significant [34], [35]. See [36] for further information.

2.3.4. Z-Score

Any calculated score distribution can be assumed to be Guassian. Therefore, a score can be normalized and transformed into a z-score using

$$Z = \frac{x - \mu}{\sigma} \tag{2.4}$$

where μ and σ are the mean and standard deviation of a score x [30].

This measure reveals the difference between the observed count and the expected count [35], and can be expressed as the number of standard deviations by which the observed score exceeds its expectation [24], [37]. Higher z-scores are better, since the farther the real score is from the mean, the more significant the motif is [34].

2.4. Algorithms for Solving Planted Motif Problem

Selecting from the large pool of algorithms for a deeper review is difficult. Thus, only algorithms that have been used to compare SMF or qSMF are treated in this section. Appendix A lists the most well-known algorithms over the last three decades. The table provides an initial look at how many algorithms have been designed to solve the motif discovery problem.

2.4.1. MEME

MEME (rhymes with "team" [38]) is one of the most powerful and widely used algorithms for searching for novel signals in sets of biological sequences such as DNA, RNA, or protein, and applies the developed Expectation Maximization (EM) algorithm to find the maximum

likelihood of a motif estimation based on the Bayesian statistical model [25]. MEME is a probabilistic method that tries to maximize the relative entropy obtained from the construction of the position specific score matrix (PSSM) and has a run time that is fast but falls within local optima [30]. MEME takes time $O(n^2)$ where n is the size of a dataset in characters [39].

In further detail, MEME uses the technique of EM to fit a two component finite mixture model to a set of DNA or protein sequences. One component describes a set of similar sequences, or motifs, while the other component describes all other positions in the sequences or background. The input of the algorithm is only a set of unaligned sequences and a motif width. The algorithm estimates how many times each motif occurs in each sequence in the dataset and outputs an alignment of the occurrences of the motif. MEME is capable of discovering several different motifs with differing numbers of occurrences in a single dataset [40].

The name MEME has several explanations: first as an acronym for Multiple EM for Motif Elicitation; second as an English word *meme* meaning a theme or motif whose propagation through cultural evolution is similar to the propagation of genes in biological evolution; and third as a greedy algorithm (me! me! algorithm) [39] because at each step, it takes the best solution.

MEME has been developed over the past three decades. Therefore, identifying the original published algorithm from the many published papers associated with MEME has caused some confusion. In 1990, Charles E. Lawrence and Andrew A. Reilly, who introduced EM for motif finding, described a statistical method for identification and characterization of protein binding sites in a set of unaligned DNA fragments [41]. They developed the EM algorithm to generate estimates of the probabilities that the sites are located in each possible position in each sequence. As a result, the most likely binding sites with maximum

likelihood estimates were predicted. In 1993, Timothy L. Bailey and Charles Elkan wrote a technical report, CS93-302, where they first mentioned the MEME algorithm with its full details. This report was published in 1995 as [39], but in 1994, the same authors published a paper [40] that described an MM algorithm with its context of MEME. The MM algorithm is an extension of the EM technique for fitting finite mixture models developed by Aitkin and Rubin in 1985 [42], and was also related to the algorithm described by Charles E. Lawrence and Andrew A. Reilly in 1990. The paper [40] published in 1994 is considered the main MEME algorithm paper. In 1995, MEME's authors developed the algorithm further with several extensions in [43] as its third version. In this paper, they first introduced the OOPS (One occurrence Per Sequence), ZOOPS (Zero or One Occurrence Per Sequence), and TCM (Two-Component Mixture) models. By 2006, the MEME performance was significantly improved in [38] with the introduction of a higher-order background model where the first web server for MEME was published. The MEME Suite was published in 2009 [44] and further developed in 2015 [45] as mentioned in the section Useful Online Tools. There is also a useful Google group at `https://groups.google.com/forum/#!forum/meme-suite` for frequently asked question regarding installing and using the MEME suite.

In order to understand the MEME algorithm, a simple background about the EM algorithm is given. The EM algorithm described in [46] clarifies the analysis of a problem with missing information by iteratively solving a sequence of problems in which expected information is substituted for missing information. This expected information is used at each step to solve the more straightforward problem associated with having complete information by maximizing the likelihood. The MEME algorithm employs the EM algorithm developed by Lawrence and Reilly in [41] and providing further extensions to solve the motif finding problem since the locations of the sites are considered the missing information. The name of the EM algorithm is derived from its two iterative steps of the Expectation (E) step and

Maximization (M) step. Both steps are repeated alternately until a convergence condition is met [41].

2.4.2. APMotif

The authors of [47] utilized Affinity Propagation (AP) clustering and the EM algorithm for their motif finding problem. They proposed the APMotif algorithm which consists of 4 stages: constructing clusters, extracting clusters, refining clusters, and verifying motif instances. The clustering process gives rise to long execution times. The algorithm selects a sequence X_1 as a reference sequence, then for each $l-$mer x_k (where $k = 1, 2, ..., n - l + 1$) in X_1, a cluster $C(x_k, X)$ is constructed as the set of all $l-$mers $x^{'}$ in $X - \{X_1\}$ such that $d_H(x_k, x^{'}) \leq 2d$ (where X is the given set of sequences and d_H is the Hamming distance). Thus, for each l-mer of the sequence X_1, all the neighbors at a distance $\leq 2d$ are collected. As a result, there are $n - l + 1$ clusters. This wide distance ($2d$) allows for a lot of neighbors to be in each cluster. The conserved clusters are extracted using AP clustering and refined using expectation maximization.

2.4.3. YMF

The Yeast Motif Finder (YMF) algorithm was first introduced in [24] and is a statistical method which uses the z-score as a measure for motif significance over the DNA alphabet $\{A, C, G, T\} \cup$ the degenerate symbols $\{R, Y, S, W\} \cup$ the spacer character $\{N\}$. The algorithm first makes a pass over the input sequences and calculates the number of occurrences of each motif in either direction, then calculates their z-scores. The output is the sorted motifs with the highest z-score values. YMF has been implemented and tested on 17 well studied co-regulated sets of genes in yeast S. cerevisiae [48] to find the transcription factor

binding sites which are of length 6-10 bp (N is not included). The input is the corresponding upstream sequences of length 800 bp. YMF was also tested on eight datasets of co-expressed gene clusters. In addition, the authors ran the algorithm through several independent sets of simulated data generated by employing the Markov chain of order 3 to model the background genomic distribution. The running time to calculate the z-score for a single motif is $O(c^2k)$, where k is the number of non spacer characters in the motif and c is the number of possible instantiations of R (purine), Y (pyrimidine), S (strong), and W (weak) symbols.

The YMF algorithm usually returns a long list of sorted motifs based on their statistical significance [49]. For this reason, YMF was further discussed and developed in [37] with a focus on its application to classes of yeast genes. Finally, a web interface for YMF was presented in [50] and is unavailable online, but the algorithm is available through the Tmod software [25]. Recently, the authors considered YMF as a retired algorithm.

2.4.4. MFMD

The Memetic Framework for Motif Discovery (MFMD) [30] is an algorithm published in 2018 to find and classify over-represented patterns in DNA sequences and predict their positions using semi-greedy heuristics and the hybridization of genetic algorithms. MEMD focuses on solving the de novo motif problem. The de novo motif discovery means the localization of the motifs without any prior knowledge. It was tested on several datasets such as ChIP-Seq data retrieved from the JASPAR database (see Table 2.1), promoter sequences extracted from the ABS database (see Table 2.1), and artificially generated datasets. The motif quality is measured using the information content and complexity score. Its performance was assessed using precision, recall, and the f-score when it was compared with well known approaches such as the MEME and the Gibbs Motif Sampler. The algorithm and the used datasets can be downloaded through `https://github.com/jadermcg/mfmd`. The

algorithm's complexity is $O(n^t)$ or $O(tn^2)$.

2.4.5. Weeder

The basic version of the Weeder algorithm was first presented in [18] in 2001. Its input is a set of t sequences on alphabet $\sum = \{A, C, G, T\}$ and an error ratio $\epsilon < 1$. Weeder's job has been to find all patterns that occur in at least q sequences of the input set with at most ϵl mutations, denoted by d, where l is the pattern length. It utilizes the suffix tree to represent the set of sequences and begins by searching for valid pattern occurrences. Instead of reducing the number of patterns to be searched, Weeder reduces the execution time by narrowing the set of valid occurrences for each pattern. In other words, the number of paths that have to be searched is restricted. The idea behind the algorithm's name is *weeding out* all the paths that are unlikely to happen. Therefore, it is almost an exact algorithm (approximate algorithm). In addition, to narrow the paths further, the Weeder's authors imposed a restriction on the mismatch locations.

The algorithm complexity is $O(|\sum|^d L^d tN)$ where L is the length of the longest pattern, $l < L$ and N is the total length of the t sequences. It is exponential in the number of allowed mutations but not in a pattern length. Thus, the algorithm works well only for small values of d. Then, the algorithm's complexity was reduced to $O(\lceil 1/\epsilon \rceil^d |\sum|^d tN)$ where $d = \lceil \epsilon L \rceil$. There is no need to provide the algorithm with the pattern length and the maximum allowed mutations since they are determined dynamically. The authors used three statistical measures to sort the output and highlight the pattern significance. One of these measures is relative entropy. The Weeder algorithm seems to work better on a very large set of short sequences (up to 600 nucleotides) rather than a small set of large sequences. Pavesi et at. tested their algorithm on a single challenge problem where 20 sequences were generated based on the independent and identically distributed (i.i.d.) model, each containing an

unknown pattern of length 15 with 4 mismatches where sequence lengths varied from 100 to 1000 nucleotides. They did not test it on real datasets. In 2005, comparative assessment in [51] of the performance of 13 motif discovery tools showed that the Weeder algorithm had a more satisfactory performance with respect to the other tools. The Weeder output is only a consensus motif. In order to locate its instances in a set of sequences, another program or tool called the Motif Locator has to be used in the post-processing stage. The Motif Locator is available online at `http://159.149.160.88/modtools/`. In [22], a WeederWeb was introduced in 2004 which is the web interface for the Weeder algorithm. Unfortunately, it is unavailable online.

In 2006, the MOtif Discovery (MoD) Tools web server was introduced in [52]. The MoD tools can be browsed though the link `http://159.149.160.88/modtools/`, and includes a set of tools dedicated to the novel conserved sequence and structure motif discovery such as Weeder, WeederH, and RNAProfile. The user can install the Weeder version 1.4.2 through the link `http://159.149.160.51/modtools/`. This version is for motif discovery in sequences from co-regulated genes of a single specie. Weeder 2.0, a newer version, was released after being rewritten to improve speed and optimize larger ChIP-Seq data.

WeederH in [23] is an algorithm for the discovery of conserved TFBSs and distal regulatory modules in sequences from homologous genes published in 2007. Using a reference sequence with any number $k \geq 1$ of homologous sequences, motif size, and maximum number of substitution as an input, the algorithm starts by matching each oligo of a suitable size of the reference sequence with the homologous sequences. The matches that are within the allowed substitution threshold are scored with a measure that takes into account sequence and position conservation. High scored matches are kept, while scored oligos are scored again with a relative score. Finally, high scored oligos are merged when possible to obtain longer motifs and regions. The algorithm was tested on both simulated and real datasets. The

simulated datasets were generated using the Dawg program [53] that allows simulation of sequences with insertion and deletion operations. The authors used the ABS database [54] (See Table 2.1). The results showed that WeederH outperformed both the FootPrinter [55] and phasCons [56] algorithms.

2.4.6. Trawler

Trawler [57] is a de novo computational pipeline dedicated to discover over-represented motifs in ChIP experiment data with their instances. A suffix tree is used to index large datasets. The significance of motifs is assessed by using the z-score. The Trawler algorithm was tested on several datasets such as Tompa's et al. benchmark datasets, S. cervisiae yeast datasets, and mammal datasets. Compared to AlignACE [58], MEME [38], Motifcut [59] and Weeder [22] algorithms, Trawler's results showed superior performance regarding speed and accuracy [57].

2.4.7. MDscan

The Motif Dicovery scan (MDscan) [60] is a computational method which tests the ChIP-array sequences to search for DNA motifs that represent the protein-DNA interaction sites. MDscan combines the advantages of word-enumeration and position-specific weight matrix updating strategies to speed up the search and enhance results. When tested on simulated and real datasets such as ChIP-array experiments of yeast datasets, MDscan performance was faster than the BioProspector [61], CONSENSUS [62], and AlignACE [63] in locating the published motifs.

2.4.8. FMotif

The FMotif algorithm [64] is an exhaustive method for finding long (l, d) motifs in DNA sequences. The algorithm's authors submitted a simple sampling strategy for ChIP-enriched data. They tested their algorithm on synthetic samples and several ChIP datasets such as 16 ChIP-seq datasets and 5 ChIP-exo datasets. The results demonstrated that the FMotif algorithm was able to find these motifs with high efficiency and accuracy.

2.4.9. AlignACE

The Aligns Nucleic Acid Conserved Elements (AlignACE) algorithm [63] is a Gibbs sampling algorithm for identifying motifs that are over-represented in a set of DNA sequences. The AlignACE is an extension of the exhaustive Gibbs Motif Sampling algorithm [65] with several distinctions. AlignACE has been optimized for finding multiple motifs and DNA sequence alignments of both strands via an iterative masking procedure. The algorithm implements two proposed scoring methods, MAP and general specified, to evaluate the alignment significance by their frequency of occurrence. In [63], AlignACE was used to find transcriptional regulatory DNA motifs in Saccharomyces cerevisiae yeast using groups of genes derived from genome-wide mRNA expression data. AlignACE was able to return many more motifs that were not known in the literature. In addition, the experimental results showed that AlignACE has lower sensitivity to transcripts of low abundance as compared with previously published S. cerevisiae expression studies.

AlignACE was studied further in [58] when applied on 248 variety groups of genes for same yeast type. The algorithm's authors proposed two statistical measures, group specificity and positional bias, for motif significance to refine the long list of 3311 of returned motifs. The results illustarted that AlignACE was able to return many known cis-regulatory motifs

as well as novel motifs.

2.5. Algorithms for Large Datasets

The recent introduction of technologies such as chromatine immunoprecipitation (ChIP [66]) coupled with tiling arrays (ChIP on Chip [67]) or next-generation sequencing (ChIP-Seq [68]) has allowed the genome-wide identification of the regions bound by a given transcription factor (TF). In other words, technologies allow for the identification a set of genomic regions (sequences) whose binding sites are bound by the same transcription factor. ChIP-Seq has quickly become the actual standard in this field, which poses new challenges for the developers of algorithms and tools [7].

The regions derived from ChIP experiments are perfect case study data for finding binding sites. Typically, the output of ChIP experiments is a list of thousands of sequences of a size seldom exceeding a few hundred base pairs. This has led to better results with new challenges for the algorithm developers. Applying ChIP experiment sequences makes motif discovery methods more reliable than when using promoter sequences. The frequency of binding sites is much higher in regions coming from a ChIP, while in promoter regions of co-expressed genes, there is no guarantee for even a single occurrence in a single sequence. Further, ChIP experiments submit clearer sequences with more redundancy because thousands of sequences are expected to find several instances of binding sites that are highly similar to each other, whereas with gene promoters, the sets of sequences are less clean with much smaller numbers of longer sequences. As a result, different binding sites are returned with greater differences from one another. Still, regular motif discovery methods have a reputation of low performance when applied on ChIP data. The main reason is that the input size becomes significantly larger [7].

Many former and current algorithms are being developed to deal practically with ChIP

sequences. Some are profile-based methods such as MEME-ChIP [69], Gibbs Sampler [70], and STEME [71], which is a faster version of MEME where the sequences are indexed with a suffix tree, with feasible time requirements for ChIP-Seq data. STEME is developed further in [72]. There are also consensus-based methods like MDscan [60], Trawler [57] with its web-based release at [73], and Amadeus [74] (for ChIP on chip) with online access at `http://acgt.cs.tau.ac.il/amadeus/`. Amadeus is now integrated with Allegro [75], which is also accessable online through `http://acgt.cs.tau.ac.il/allegro/`. Other available tools are DREME [76], CisFinder [77], cERMIT [78], and RSAT Peak-motifs (for ChIP-Seq) [79] with online access at `https://rsat01.biologie.ens.fr/rsat/peak-motifs_form.cgi`.

SamSelect [80] is an algorithm for sampling large DNA sequences for the quorum planted motif search problem. DeepFinder [81] is an integration of a feature-based and deep learning approach, and employs an initial subset of ChIP input sequences to predict initial motifs that are employed for site detection in the remaining input sequences.

Genomic-wide ChIP experiments for TFs can be a source for building great feasible benchmark sequence sets for testing the motif finding algorithms, like the Harbinson dataset [82] and the metazoan dataset introduced in [74], which are composed of several promoter sets mostly generated from genome-wide ChIP on Chip. Both datasets can be considered as hybrid benchmarks since they are composed of promoter sequences with TF binding that have been identified through ChIP [7].

The introduction of ChIP technologies helped develop other tools referenced as peak-calling, which are considered a hot topic in research today. These tools identify the enriching regions in the ChIP-Seq experiments, and are presumed to support the algorithm results [7].

2.6. Useful Online Tools

There are many online tools that motif discovery method developers may use for different purposes. This section covers the main tools that are currently available.

- **Tmod:** The Toolbox of Motif Discovery (Tmod) is software for Windows operating system users since the majority of motif finding algorithms were made to run on the Linux operating systems. Tmod provides a unified interface to simplify the use of these programs and help users understand the tuning of their parameters. The reason behind developing such a toolbox is because there are many de novo motif finding tools that have been developed by researchers, but most of these tools do not have a user friendly interface so their results are not easily comparable. The current version of Tmod integrates 12 widely used motif discovery algorithms and tools [25]: MDscan [60], BioProspector [61], AlignACE [63], Gibbs Motif Sampler [83], MEME [40], CONSENSUS [84], MotifRegressor [85], GLAM [86], MotifSampler [87], SeSiMCMC [88], Weeder [18] and YMF [50]. Tmod utilizes BioOptimizer [89] (available at `https://sites.fas.harvard.edu/~junliu/BioOptimizer/`), which combines and compares the returned motifs using a score function based on a Bayesian model. Tmod is available for download at `http://www.fas.harvard.edu/~junliu/Tmod/`.

- **MEME Suite:** The MEME Suite is a web server that provides a unified portal for online discovery and analysis of sequence motifs [44], and includes the popular MEME motif discovery algorithm with other useful tools. The Suite was first introduced in 2009 and updated in 2015 in [45]. The MEME suite is freely available for academic use at `http://meme-suite.org/index.html`. In addition, the source code is available for download and local installation. The MEME Suite supports motif-based analysis of DNA, RNA and protein sequences. MEME Suite also allows for the discovery of

motifs with arbitrary insertions and deletions through a tool called GLAM2 [90]. In addition to motif discovery, the MEME Suite provides tools for scanning sequences to match motifs using FIMO [91], MAST [36] and GLAM2Scan [90] tools, and can scan for clusters of motifs using MCAST [92], compare motifs to known motifs using Tomtom [93], find preferred spacings between motifs using SpaMo [94], predict the biological roles of motifs using GOMo [95], measure the positional enrichment of sequences for known motifs using CentriMo [96], and analyze ChIP-seq and other large datasets using MEME-ChIP [69].

- **Dust:** Dust is a tool that was created by R. Tatusov and D. J. Lipman (unpublished work) with a goal to remove subsequences with low complexity from a dataset. Dust is available on `ftp://ftp.ncbi.nlm.nih.gov/pub/agarwala/dustmasker/` and through `http://meme-suite.org/doc/dust.html`. Dust was used by the authors of [30] to normalize raw datasets.

- **WebLogo:** Designed as a web-based application, WebLogo makes the generation of sequence logos easy and painless. WebLogo has been featured in over 4000 scientific publications [97] and is available at `https://weblogo.berkeley.edu/`. Its source code can be downloaded. The newer release WebLogo3 is available at `http://weblogo.threeplusone.com/`. Another web-based logo generator is called enoLOGOS, which generates sequence logos from a variety of input data including energy measurements, probability matrices, alignment matrices, count matrices and aligned sequences [98]. The enoLOGOs tool can be accessed at `http://www.benoslab.pitt.edu/cgi-bin/enologos/enologos.cgi`.

- **STAMP:** STAMP [99] is a free online tool that can be used to check the motif prediction results against experimentally validated TFBS from dedicated databases such

as TRANSFAC and JASPAR (See Table 2.1). STAMP is accessable online through http://www.benoslab.pitt.edu/stamp/.

- **Peak Calling Tools:** Peak calling is a computational method that can be used to identify motif enriched areas in ChIP data such as [100], and can predict the motif regions as input for motif finding algorithms that search for motifs in ChIP-Seq datasets [101]. Another peak calling tool is RSAT that can be accessed at https://rsat01.biologie.ens.fr/rsat/peak-motifs_form.cgi, and https://academic.oup.com/nar/article/46/W1/W209/4990780.

- **Melina II:** In order to show potential DNA motifs in promoter regions, Melina II [102] is combined with several available programs such as Consensus [31], MEME [40], Gibbs sampler [103], MDscan [60] and Weeder [18] with several parameter settings. Melina II enables the running of a maximum of four programs simultaneously and the comparing of their results with graphical representations. In addition, users can build a weight matrix from a predicted motif and apply it to upstream sequences of several typical genomes (human, mouse, S. cerevisiae, E. coli, B. subtilis or A. thaliana) or to the public motif databases of JASPAR or DBTBS in order to find similar motifs. Melina II is accessible over the web at http://melina.hgc.jp.

2.7. Useful Databases

Most algorithms have been tested on real datasets, but some of them are no longer available, while other ones are well-maintained and updated. Table 2.1 lists 12 useful currently available online databases that can be used to test the performance of motif discovery methods. Most of these are free.

Table 2.1: Useful Databases

Database	Database Description	URL
ABS [54]	Annotated regulatory Binding Sites	http://genome.crg.es/datasets/abs2005/index.html
EPD [104],	Eukaryotic Promoter Database	https://epd.vital-it.ch/index.php
GTRD [105] [106]	Gene Transcription Regulation Database	http://gtrd.biouml.org/
JASPAR [107],	transcription factor binding profile	http://jaspar2016.genereg.net/
PRODORIC [108]	Gene Regulation and Expression in Prokaryotes	http://www.prodoric.de/
RegulonDB [109]	Escherichia coli K-12 gene regulation database	http://regulondb.ccg.unam.mx/
SCPD [48]	Saccharomyces Cerevisiae Promoter Database	http://rulai.cshl.org/SCPD/index.html
SGD [110]	Saccharomyces Genome Database	https://www.yeastgenome.org/
TRANSFAC [111]	Transcription Factors binding sites of eukaryotic	http://genexplain.com/transfac/
YEASTRACT [112]	transcription regulatory networks in S. cerevisiae	http://www.yeastract.com/
UniPROBE [113]	Universal PBM Resource for Oligonucleotide Binding Evaluation	http://thebrain.bwh.harvard.edu/uniprobe/

2.8. Ensemble Algorithms

Historically, motif finding algorithms have suffered from low performance issues. In early studies such as [51], when comparing 13 algorithms, the results showed low performance even for the best algorithm. A major reason is that all computational-based algorithms suffer from a high predication rate of false positives. Increasingly, many studies encourage researchers to combine the results of various algorithms as ensemble tools to improve prediction accuracy [101].

Ensemble motif finding algorithms combine several individual algorithms to solve the motif finding problem and produce a solution agreed upon by most of the algorithms. This technique attains better prediction accuracy. In the past decade, several ensemble tools have been developed such as Scope [114], MotifVoter [115], GimmeMotifs [116] [117], EMD [118], WebMOTIFS [119], CompleteMOTIFS [120] and DynaMIT [121]. The most recent tool, the Motif Combining and Association Tool (MCAT) [34] was published in 2019. MCAT combines the state-of-art motif discovery tools of MEME [44], BioProspector [61], DECOD [122], XXmotif [123], Weeder [22], and CMF [124]. The challenge after applying the individual algorithms is how to combine the results and rank them [34]. A good review of recent ensemble methods for de novo motif discovery before and after the ChIP-Seq data era can be found in [101].

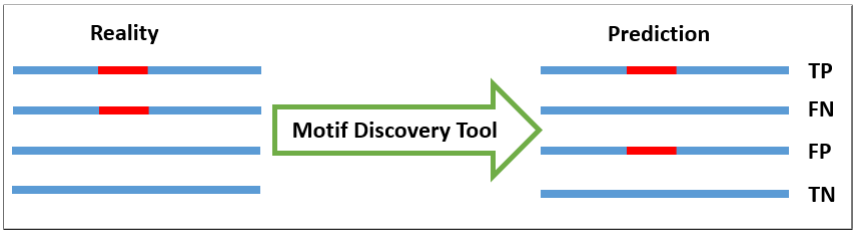


Figure 2.1: Visual Description of TP, FN, FP, and TN

2.9. Prediction Accuracy Evaluation

Assessing the performance of different motif finding algorithms has always been far from straightforward [7]. Algorithms were often tested on synthetic datasets in which simulated binding sites were implanted into simulated sequences [20] [16]. Some benchmark datasets derived from experimental data have been introduced over the past few years by Tompa et al. in [51], and 13 well known tools were compared. The results reported low performance of such tools as MEME and Weeder. Later, an improved benchmark data set by Tompa was presented [125], based on a machine learning perspective.

In order to assess algorithm performance, we use nine metrics that are given in the literature to measure the prediction accuracy. It is an easy process to calculate the prediction accuracy of the (predicted) sites if the known sites are given. Two levels of prediction to be tested are the nucleotide level and the site level. The metrics definitions and other pertinent definitions related to their computation are found in [51] [126]. In the list below, n and s indicate nucleotide and site respectively. T means True, F means False, P means Positive, and N means Negative.

1. **nTP:** is the number of nucleotide positions in both the known and predicted sites. The prediction suggests that the nucleotide positions are positively part of a motif, and in reality, it is a true prediction.

2. **nFN:** is the number of nucleotide positions in the known sites but not in the predicted sites. The prediction suggests that the nucleotide positions are negatively part of a motif, and in reality, it is a false prediction.

3. **nFP:** is the number of nucleotide positions not in the known sites but in the predicted sites. The prediction suggests that the nucleotide positions are positively part of a motif, and in reality, it is a false prediction.

4. **nTN:** is the number of nucleotide positions in neither the known sites nor the predicted sites. The prediction suggests that the nucleotide positions are negatively part of a motif, and in reality, it is a true prediction.

5. **sTP:** is the number of known sites overlapped by the predicted sites. The prediction suggests that the sites are positively part of motifs, and in reality, it is a true prediction.

6. **sFN:** is the number of known sites not overlapped by the predicted sites. The prediction suggests that the sites are negatively part of motifs, and in reality, it is a false prediction.

7. **sFP:** is the number of predicted sites not overlapped by the known sites. The prediction suggests that the sites are positively part of motifs, and in reality, it is a false prediction.

It can be inferred that $n = nTP + nFN + nFP + nTN$. In the definitions of the metrics below, note that x indicates n (nucleotide level) or s (site level). These metrics are used to assess the motif position prediction accuracy of profile-based algorithms. Figure 2.1 gives a visual representation for TP, FN, FP, and TN.

1. **Performance Coefficient nPC:** gives an indication about how many motif positions have been predicted correctly [15]. This metric can be used to assess the prediction

accuracy of profile-based algorithms. Pevzner and Sze in [20] define the nucleotide level performance coefficient as $|K \cap P| \,/\, |K \cup P|$, where K is the set of known signal positions and P is the set of predicted positions. Their formula is equivalent to Eq. (2.5) used in [51],

$$nPC = \frac{nTP}{(nTP + nFN + nFP)} \tag{2.5}$$

2. **Sensitivity xSn:** gives the fraction of known sites or site nucleotides that are predicted,

$$xSn = \frac{xTP}{(xTP + xFN)} \tag{2.6}$$

3. **Positive Predictive Value xPPV:** gives the fraction of predicted sites or site nucleotides that are known,

$$xPPV = \frac{xTP}{(xTP + xFP)} \tag{2.7}$$

4. **Specificity nSP:** gives the fraction of non-site nucleotides that are predicted as non-site nucleotides. However, the number of non-coding nucleotides in DNA sequences is much larger than the number of coding nucleotides. For this reason, TN tends to be much larger than FP,

$$nSP = \frac{nTN}{(nTN + nFP)} \tag{2.8}$$

5. **Correlation Coefficient nCC:** is defined by Burset and Guigo in [126]. The value of nCC ranges from -1 (indicating perfect anti-correlation) to +1 (indicating perfect correlation),

$$nCC = \frac{(nTP)(nTN) - (nFN)(nFP)}{\sqrt{(nTP + nFN)(nTN + nFP)(nTP + nFP)(nTN + nFN)}} \tag{2.9}$$

6. **Average Site Performance sASP:**

$$sASP = \frac{sSn + sPPV}{2} \tag{2.10}$$

7. **Simple Matching Coefficient nSMC:** is the probability of a correct prediction,

$$nSMC = \frac{nTP + nTN}{(nTP + nFN + nFP + nTN)} \tag{2.11}$$

2.10. Computational Methods: Limitations and Challenges

The motif discovery problem is a challenging problem. The computational-based methods to find motifs within molecular biology data suffer from several limitations. This section takes a general look at these limitations and challenges.

Finding regulatory elements can be formulated as the problem of discovering the over-represented motifs within regulatory regions. Finding motifs seems simple at first sight, since these motifs occur multiple times in the same genome. But in reality, this approach is complicated because most of these motifs or binding sites are short with possible variations. In other words, variations of these motifs can be found at random throughout the genome. The challenge is how to distinguish between real and random motifs [5]. This leads to another challenge, involving a long list of predicted binding sites that includes the real motifs (actual corresponding binding sites) along with a large number of random variations of the real ones [49].

During the past 30 years, more than a hundred methods that have been proposed for motif discovery. This implies large variations in underlying algorithmic approaches and models, and how these algorithms are described and tested. As a result, it is difficult to get

a good overview of the field to see which are the best algorithms and which to choose for comparisons [5].

In [51], when comparing thirteen motif discovery tools, the authors mentioned that it is very difficult to compare the performance of these tools, especially in complex genomes like humans. One method may have had a better performance on one type of genome compared to other genomes, making it more difficult to assess the performance in general. As a result, there is a continuing need for more standardized routines for testing and comparing alternative approaches [5].

Defining optimal test sets to be used as benchmarks is difficult. Even when comparing method performance using real biological sequences with experimentally verified binding sites, the real sequences may contain additional binding sites that are still not identified. These undefined binding sites lower any method's performance since they are considered as false positives. This limitation can be addressed by using synthetic background sequences with implanted motifs, while creating a new problem with realistic background sequences and motif distribution. In [51], Tompa et al. created benchmark datasets and applied 13 motif discovery tools. Some of the established methods such as MEME, AlignACE, and ANN-Spec performed reasonably well, at least on simple data such as yeast. However, the latest method, Weeder, was the best algorithm of all tested datasets [5].

Another challenge that motif discovery method developers may face when comparing methods is the difficulty to know whether the test results reflect the assumed methodological differences between approaches. Many methods need various levels of parameter tuning to allow for motif length, the expected number of motif occurrences and inter-motif distances. Some methods need additional data besides the actual sequences to enhance their performance, such as the methods that use related organisms for phylogenetic footprinting. These challenges may produce biased test results and make automatic testing difficult [5].

2.11. Summary

Motif finding is one of the most widely studied problems in bioinformatics. A common case study for motif discovery has been the analysis of sequences such as promoters from genes that show similar expression patterns and probability to be bound by the same set of transcription factors. Different algorithms have been developed over the past 30 years to solve this type of problem. Many significant measures have been applied to discover the most over-represented motifs. However, in spite of these efforts, the problem has been solved with limited success, especially when dealing with complex organisms, such as humans [7].

Despite all significant efforts to date, the motif discovery problem remains a challenge for biologists and computer scientists. Researchers have used different approaches in developing different motif finding algorithms and tools. Performance comparisons between these algorithms and tools and identification of the best ones have been proven to be a difficult task since the underlying algorithms and motif models are diverse and complex. In addition, our incomplete knowledge about regulatory mechanisms is not always adequate for the evaluation of these algorithms [4].

All these challenges make it difficult to compare the performance of different methods. However, the choice of test data, performance metrics and tuning parameters all have a great influence on the performance of these methods in attaining good results [5].

CHAPTER 3

PROPOSED ALGORITHMS

Chapter 3 presents research methodologies related to the SMF and qSMF algorithms, in Sections 3.1 and 3.2, respectively. The motif discovery versions are described in further detail, and the algorithms are outlined step by step. The scoring functions used to evaluate the significance of the motifs are defined. Finally, the algorithm time complexity is addressed.

3.1. SMF Algorithm

This section details the **Strong Motif Finder (SMF)** algorithm [127], starting from the basic ideas and the hypotheses on which the SMF is based. The inputs and outputs are explained first. The algorithm steps are described and detailed further in the algorithm pseudo-code. The scoring function is specified, as well as other important elements of the algorithm and their relationships with some of the inputs. The algorithm time complexity is derived based on the pseudo-code.

3.1.1. Planted (l, d) Motif Problem

The planted (l, d) motif finding problem was first introduced by Sagot in [19]. Pevzner and Sze in [20] presented a specific challenge instance of the problem with respect to *(15,4)*. The planted (l, d) motif search problem was described formally in [16] [10] [15] [128], and can be specified as follows:

Input:

- A set of t sequences $\{S_1, S_2, ..., S_t\}$ of length n over the alphabet $\{A, C, G, T\}$.

- A length l of the motif to be searched for with an allowed mutation d where $0 \leq d < l << n$.

Output:

- Consensus motif M of length l, which is the original motif without mutations.
- All motifs, substrings or neighbors, of length l such that, for each sequence S_i, $1 \leq i \leq t$, there exists at least one substring of length l at the Hamming distance $\leq d$ from M.
- Locations of the motifs.

3.1.2. Proposed Algorithm

If multiple DNA sequences from the same species or from different species share certain characteristics, i.e. that have common binding sites, it is reasonable to assume that they share one or more conserved motifs. Furthermore, if multiple DNA sequences have a common conserved motif, one or more sequences will likely have a motif instance with at most a small number of mutations.

The proposed algorithm attempts to select a sequence with a binding site with or without minimal mutations, which will lead the algorithm to the discovery of binding sites in other sequences.

3.1.3. Algorithm Input and Output

Input: The input for the algorithm includes $t \times n$ DNA sequences over the alphabet $\{A, C, G, T\}$ where t is the number of sequences and n is the length of each sequence. Also given are the length l of the motif, the maximum allowed mutation distance d, and the number of consensus motifs to be returned by the algorithm, i.e. the top k.

Output: The algorithm returns up to k consensus motifs sorted by their scores from higher to lower. Each consensus motif M is presented with its score, found binding site(s) called neighbor(s) in each sequence, and starting position(s).

3.1.4. Algorithm Description

The pseudo-code of the algorithm is listed as Algorithm 1. The algorithm starts by initializing the parameter r in **step 1**. This value is the allowed distance between each substring of length l that the algorithm is checking and its collected neighbors. Specifying its value is important in narrowing or widening the search space. The r value can range between 1 and $2d$. **Step 2** initializes the list *Tried Sequences*. This list contains the sequences processed by the algorithm that failed to return a solution.

The algorithm picks one of the t sequences randomly as the *Selected Sequence* SS in **step 3**. Possible membership of the SS in the *Tried Sequences* list is checked in **step 4**, and the SS is added to the *Tried Sequences* list in **step 5** if not already present. The core work of the algorithm is performed through **steps 6-21**, where each substring s_i of length l in the SS, $1 \leq i \leq n - l + 1$, is checked in **step 9** to see whether it is found in the other $t - 1$ sequences at the Hamming distance $\leq r$. If the condition is not satisfied, the substring s_i is not common among these sequences, so it is discarded at **step 17**. If the condition is satisfied, then s_i is considered a strong candidate to be a true motif and kept with its found neighbors in the *Neighbors* list at **step 10**.

The total number of substrings in *Neighbors* list is denoted by N. The *Neighbors* list contains the substring s_i with all similar substrings (neighbors) found in other sequences within a distance of at most r. The Hamming distance d_H is used as the distance measure, which is the total number of mismatches between two substrings.

The collected neighbors may vary from the substring s_i by the allowed distance r. If the

substring s_i is the original motif or a motif with small number of mutations, the retrieved neighbors are also close to the motif. This leads to a highly scored consensus motif and an accurate prediction of binding site locations. If the substring s_i is a motif with a larger number of mutations, the neighbors are also farther from the motif. This generates a consensus motif with a lower score and weaker prediction of binding sites. Smaller r values tend to reduce the number of collected neighbors and the running time, but can lead the algorithm in a direction where no solution may be found if r is too small. Larger r values allow for more sequences, substrings, and neighbors to participate in solutions. This increases the running time, and multiple solutions may arise due to sequences that have motif instances with large mutations.

After collecting neighbors, they are refined in **step 11** by keeping the nearest neighbors in each sequence. For example, if a sequence has four neighbors for substring s_i, two of which are at distance $d_H = 2$ and two at $d_H = 1$, the algorithm will ignore the two more distant neighbors and keep the nearest ones.

The algorithm calculates the *Profile matrix* P for the collected neighbors in **step 12**. P is a $4 \times l$ matrix that represents the frequencies of each letter $\{A, C, G, T\}$ at each location for the *Neighbors* list. A consensus motif M is generated in **step 13** from P by choosing the highest frequency at each letter position. In **step 14**, the consensus motif M is scored using the Motif Strength Score (MSS) as proposed in the next subsection where the algorithm adds M to the *Nominated Motifs* list with its score, collected neighbors and starting locations in **step 15**. This list contains all the consensus motifs that are nominated to be binding sites.

When all $(n - l + 1)$ substrings of the SS are considered, the algorithm checks whether the *Nominated Motifs* list is empty in **step 20**. If it is empty, the current SS did not return a solution, and the algorithm selects another sequence in **step 21** until a solution is reached or

no more sequences are available. If it is not empty **(step 22)**, the algorithm has a solution.

Finally, if the algorithm found a solution, then the *Nominated Motifs* list is sorted in **step 27** in descending order of the MSS score values. The SMF returns up to k of the top-ranked nominated motifs in a final output list named *Final Motifs* at **step 29**. If all sequences fail to return a solution, the algorithm is unable to reach a solution. Thus, the SMF does not guarantee finding a solution. The best scenario occurs when the SMF selects a sequence SS where a substring s_i has no mutations at all. This helps the algorithm collect true neighbors at the distance r.

3.1.5. Proposed Scoring Function

The consensus motifs M derived by the algorithm are scored in order to determine their relative strength. Various statistical scoring functions have been proposed in the literature, such information content, p-value, z-score, and sequence specificity. In [51], Tompa et al. performed an assessment of 13 computational tools for the discovery of the TFBS, which was followed by further analysis in [129]. Using three of the most used objective functions, researchers concluded that none of the available scoring functions were satisfactory to retrieve the true binding sites from the background, and there appears to be a lack of knowledge of binding site characteristics for the scoring functions to work well in general. Here we will utilize a simple statistical measure, *Motif Strength Score (MSS)*, which measures the consensus motif strength based on the generated *Profile Matrix* P according to

$$MSS = \frac{\sum_{i=1}^{l} P[M[i]][i]}{l \times N} \tag{3.1}$$

where l is the length of the motif, N is the number of neighbors, and M is the consensus string.

Algorithm 1 Strong Motif Finder Algorithm (SMF)

Input: DNA, t, n, l, d, k
Output: *Final Motifs*

Set bound for Hamming distance (r)
Tried Sequences $\leftarrow$ [ ]
Select SS randomly
while SS not in *Tried Sequences* **and** not all sequences are exhausted **do**
    Add SS to *Tried Sequences*
    for each substring s_i in SS, $1 \leq i \leq n - l + 1$ **do**
        Neighbors $\leftarrow$ [ ]
        Add s_i to *Neighbors*
        if s_i found in the other $t - 1$ sequences at $d_H \leq r$ **then**
            Add all found similar substrings to $Neighbors$
            Refine *Neighbors*
            Calculate *Profile Matrix* P for *Neighbors*
            Generate M (consensus motif)
            Compute Motif Strength Score MSS of M
            Add M to the *Nominated Motifs*
        else
            Discard s_i
        end if
    end for
    if *Nominated Motifs* is empty **then**
        Select SS randomly
    else
        break ▷ out of while loop
    end if
end while
if *Nominated Motifs* is not empty **then**
    Sort *Nominated Motifs* list
    Final Motifs $\leftarrow$ Top motifs (up to k) of the *Nominated Motifs*
    return *Final Motifs*
else
    print No Solution Found
end if

The MSS sums up the frequencies of the *Profile Matrix P* for the corresponding letters of M, and divides the sum by $l \times N$. This is a modification of the scoring function of Jones and Pevzner [1], which only uses the sum in the numerator. In the SMF algorithm, the value of $l \times N$ is the maximum frequency sum if all neighbors are exact (so $MSS = 1$). If the MSS value is near 1, that means that the substrings in the *Neighbors* list have high similarity. As such, strong motifs are motifs that are common through all sequences, but not necessarily with the most occurrences. On the contrary, motifs may be the common ones with fewer occurrences as pointed out by Sagot [19]. By the $l \times N$ scaling, the MSS measure can be used to evaluate motifs of different lengths and different numbers of neighbors.

3.1.6. Algorithm Time Complexity

Within an iteration of the **while** loop started at step 4, the SMF algorithm (Algorithm 1) consumes most of its time through steps 6-19 (the **for** loop). Each substring from the *Selected Sequence SS* is compared to all $(n - l + 1)$ substrings of the other sequences. Searching $O(t)$ sequences for a string s_i requires $O((n - l + 1)tl)$ time for step 9, and steps 10-15 take $O(lN)$ time leading to $O((n - l + 1)tl) + O(lN)$ where N is the maximum number of neighbors retrieved, and its bound is $O((n-l+1)t)$. Thus, through the **for** and **while** loops, $O((n - l + 1)t\ [(n - l + 1)tl + O(lN)])$ time may be needed, which is $O((n - l + 1)^2t^2l) + O((n - l + 1)tlN) = O((n - l + 1)^2t^2l)$.

The sort in step 23 consumes $O(N'logN')$ time, where N' is the number of motifs in the *Nominated Motifs* list. Since each s_i can lead to one consensus motif, this can yield $O(n-l+1)$ nominated motifs per *Selected Sequence SS*. Thus, $N' = O((n - l + 1)t)$ over $O(t)$ selected sequences. The final algorithm complexity is $O((n - l + 1)^2t^2l) + O(N'logN') = O(n^2t^2l)$.

Designing the SMF algorithm was an integral part of the research core. Understanding how it functions in locating motifs in smaller DNA datasets led the researchers to the

designing of another algorithm that may be more effective when examining larger datasets.

3.2. qSMF Algorithm

This section explains the proposed **quorum Strong Motif Finder (qSMF)** algorithm, which is an extension of the SMF algorithm described in the previous section. Whereas SMF focuses on solving the planted (l, d) motif finding problem (PMP) in promoter regions, qSMF targets the quorum planted (l, d, q) motif finding problem (qPMP) in ChIP-seq data. Several considerations have been taken into account in the development of the qSMF algorithm to overcome some drawbacks of the SMF.

3.2.1. Quorum Planted (l, d, q) Motif Problem

The quorum Planted (l, d, q) Motif Problem (qPMP) is a version of the PMP where the motif must occur in at least q_s sequences, $1 \leq q_s \leq t$, where q_s is called the quorum constraint [19] and t is the total number of sequences. Thus, the qPMP problem coincides with the PMP when $q_s = t$ [21], and qPMP is NP-hard [80]. This can be specified as follows:

Input:

- A Set of t sequences $\{S_1, S_2, ..., S_t\}$ of various lengths over the alphabet $\{A, C, G, T\}$.
- A length l of the motif to be searched for with an allowed mutation d where $0 \leq d < l << n$.
- Minimum number of sequences q_s where a motif has to occur.

Output:

- Consensus motif M of the length l, which is the original motif without mutations.
- All motifs, instances or neighbors, of length l such that there exists at least one substring of length l at the Hamming distance $\leq d$ from M in at least q_s sequences.

- Locations of the motifs.

3.2.2. Proposed Algorithm

The qSMF algorithm [130] is built based on the same ideas and hypotheses as the SMF algorithm with further extensions. The SMF algorithm has to be applied on sequences of the same length, while the qSMF can be applied on sequences of different lengths. The SMF targets small datasets such as sequences of promoter regions, while the qSMF is designed to work on samples of large ChIP-seq datasets. Motifs returned by the SMF occur in all input sequences, unlike those returned by the qSMF that have to occur in at least q_s sequences.

3.2.3. Algorithm Input and Output

Input: The input for the algorithm includes t' sequences of a sampled DNA dataset D' over the alphabet $\{A, C, G, T\}$, and t sequences of the target DNA dataset D where D' is a subset of D ($t' \leq t$); the percentage q' of the sequences in D' where a motif appears; the percentage q of the sequences of D where a motif must appear ($q' \leq q$); the length l of the motif; the maximum allowed mutation distance d; and the number of consensus motifs to be returned by the algorithm (i.e., the top k).

Note that the notation q (or q') is used here as a ratio of the total number of sequences, so that the number of sequences in the subset is $q_s = qt$ (or $q'_s = q't'$).

Output: The algorithm returns up to k consensus motifs that are sorted in descending order according to their total number of instances. Each consensus motif M is presented with its score, as well as located binding site(s) called neighbor(s) that occur in at least q_s sequences in the dataset D, and with their starting position(s).

3.2.4. Algorithm Description

The qSMF algorithm is designed to collect all motifs that appear in the sampled dataset D' and then use these motifs to search for their instances in the target dataset D. Thus, qSMF is a two-pass algorithm. During the first pass, the algorithm looks for substrings (motifs) that are in at least $q't'$ sequences in D' then derives the substrings' consensus and prepares a *FirstPassNominatedMotifs* list for the next pass without saving the substrings, their locations or even calculating their scores. The algorithm then uses this list to search for occurrences of each nominated motif in D during the second pass. The nominated motif must appear in at least qt sequences. The output of the second pass is a *FinalMotifs* list that is sorted in descending order with respect to the number of neighbors (occurrences).

The pseudo-code of the algorithm is listed as Algorithm 2. The algorithm starts by initializing a parameter r in **step 1**. This value is the allowed distance between each substring of length l to be considered and its collected neighbors. Specifying the r value is important for narrowing or widening the search space. The r value can range from 1 to $2d$ by default $r = d$. **Step 2** creates a list called the *Shuffled Seqs*. This list contains all sequence indices from 1 to t' that are then shuffled in a random manner.

The algorithm loops in **step 3** over the SS sequences in the randomly *Shuffled Seqs* list. The loop of **step 4** moves over the substrings s_i of length l in SS, $1 \leq i \leq n - l + 1$, where n is the length of SS. At **step 6**, the complexity score CS is calculated for s_i according to (3.3) below and compared to $CS_{threshold}$ using (3.4). Thus, the algorithm only chooses the substrings with high complexity scores and will discard those with low scores in **step 15**. Substring s_i is checked in **step 7** to see whether it occurs in at least $q't'$ sequences within the Hamming distance $d_H \leq r$. If the condition is not satisfied, this means that the substring s_i is not common among these sequences, so it will be discarded at **step 13**. If the condition

is satisfied, then s_i is nominated as a candidate to be a true motif and kept with its found neighbors in the *Neighbors* list at **step 8**. The *Neighbors* list contains the substring s_i with all similar substrings (neighbors) found in at least $q't'$ sequences within a distance of at most r. The Hamming distance d_H is used as the distance measure, which is the total number of mismatches between two substrings.

The algorithm calculates the *Position Frequency Matrix PFM* for the collected neighbors in **step 9**. The *PFM* is a $4 \times l$ matrix that represents the frequencies of each letter $\{A, C, G, T\}$ at each location on the *Neighbors* list. The *PFM* is also called a *Profile Matrix*. A consensus motif M is generated in **step 10** from the *PFM* by choosing the highest frequency at each letter position. The algorithm adds M to the *FirstPassNominatedMotifs* list in **step 11**. This list contains all the consensus motifs that are nominated to be binding sites.

Next, the algorithm begins a second pass if the *FirstPassNominatedMotifs* list is not empty (**step 16**). Each *consensus* (string) in this list is checked in **step 19** to see whether it is found in at least qt sequences at $d_H \leq r$. If the condition is not satisfied, this means that this *consensus* is not common among the target dataset and is only common in the sampled dataset; therefore, it is discarded at **step 30**. If the condition is satisfied, the *consensus* is considered as a nominated candidate to be a true motif and kept with all its found neighbors in the *NeighborsInfo* list at **step 20**.

In the first pass, the collected motif instances are not filtered, while in the second pass, the motif instances (neighbors) are filtered twice. At **step 21**, the *NeighborsInfo* list is filtered based on the distance from their *consensus*, and keeps only the nearest neighbors. For example, if a sequence has four neighbors for substring s_i, two of which are at $d_H = 2$ and two at $d_H = 1$, the algorithm will ignore the two more distant neighbors and keep the nearest ones, which is done the same way as in the SMF algorithm. Then, the refined

NeighborsInfo is filtered further with the qSMF algorithm by keeping only the neighbors that achieved the highest *Match Scores* in **step 24**. This can be done by re-calculating the *PFW* for the refined *NeighborsInfo* (**step 22**). The *PFM* is converted into its corresponding Position Weight Matrix (PWM) using [34],

$$PWM[i][j] = \log_2 \left(\frac{p(i,j)}{p(i)} \right) \tag{3.2}$$

where $p(i,j)$ is the frequency for a nucleotide i at position j, and $p(i)$ is the background frequency of nucleotide i. The background frequencies are considered uniform when $p(A) = p(C) = p(G) = p(T) = 0.25$. The *Match Score* is calculated using (3.6).

The total number of refined neighbors in the *NeighborsInfo* list is denoted by N. A consensus motif M is generated again in **step 26**. Here M represents a motif that is found in the target dataset D. In **step 27**, the *Motif Strength Score* (*MSS*) of M is calculated according to (3.1) below. Then the algorithm adds M with its information, such as its *MSS* score, refined neighbors and their locations, to the *SecondPassNominatedMotifs* list in **step 28**. This list contains all the consensus motifs that are nominated as final binding sites.

At **step 31**, the algorithm checks the emptiness of the *SecondPassNominatedMotifs* list, and sorts it (if not empty) in descending order according to their total number of neighbors N (**step 32**). The algorithm designates up to k consensus motifs as its *Final Motifs* list at **steps 33-34**. This is the case where the algorithm reaches a solution and terminates successfully at **step 35**. Otherwise, the algorithm terminates without a solution at **step 36**. In the latter case, it is advised to change the algorithm parameters, in particular q' and q.

3.2.5. Scoring Functions

The researchers designed the scoring function for the SMF, but for qSMF, multiple scoring functions from the literature are used. These included Complexity Score (CS), Motif Strength Score (MSS), and Match Score.

3.2.5.1. Complexity Score (CS). Wootton and Federhen [33] used complexity vectors to express the compositional complexity of a sequence. For DNA, n_i denotes the total number of nucleotides present in the string of type $i \in \{A, C, G, T\}$. Thus, a complexity state vector is of the form (n_1, n_2, n_3, n_4). The number of associated sequences of length l per composition characteristic of the state vector (not distinguishing between the letter names) is given as

$$CS = \frac{l!}{\prod_{i=1}^{4} n_i!} \tag{3.3}$$

Table 3.1 illustrates sequences of length $l = 10$ in increasing order of complexity, starting from a mono-nucleotide sequence and introducing changes of N_{type} types of nucleotide in $N_{\#pos}$ positions.

A threshold CS was defined in 3.4 such that strings with lower complexity will be considered as noise, corresponding to a change of one nucleotide in the mono-nucleotide sequence. Thus, the complexity vector is $(l - N_{\#pos}, N_{\#pos}, 0, 0)$ if $l - N_{\#pos} \geq N_{\#pos}$, otherwise, $(N_{\#pos}, l - N_{\#pos}, 0, 0)$.

$$CS_{threshold} = \frac{l!}{(l - N_{\#pos})! N_{\#pos}!} = \frac{\prod_{i=0}^{N_{\#pos}-1}(l - i)}{N_{\#pos}!} \tag{3.4}$$

Table 3.1: Complexity Score and Vectors for Strings of Length $l = 10$ Derived from a Mono-Nucleotide String for Different Values of $N_{\#pos}$ and N_{type}

String	Vector	$N_{\#pos}$	N_{type}	CS
AAAAAAAAAA	$(10, 0, 0, 0)$	0	0	1
ACAAAAAAAA	$(9, 1, 0, 0)$	1	1	l
ACAAACAAAA	$(8, 2, 0, 0)$	2	1	$l(l-1)/2$
ACAAATAAAA	$(8, 1, 1, 0)$	2	2	$l(l-1)$
ACAAACAAAC	$(7, 3, 0, 0)$	3	1	$l(l-1)(l-2)/6$
ACAAATAAAC	$(7, 2, 1, 0)$	3	2	$l(l-1)(l-2)/2$
ACAAATAAAG	$(7, 1, 1, 1)$	3	3	$l(l-1)(l-2)$

3.2.5.2. Motif Strength Score (*MSS*). This measure was first introduced in [127]. It measures the consensus motif strength based on the generated *Position Frequency Matrix PFM* according to

$$MSS = \frac{\sum_{i=1}^{l} PFM[M[i]][i]}{l \times N} \tag{3.5}$$

where l is the length of the motif, N is the number of neighbors, and M is the consensus string.

The *MSS* computes the frequencies of the *PFM* for the corresponding letters of M, and divides the sum by $l \times N$. If the *MSS* value is near 1, the neighbors in the *NeighborsInfo* list have a higher degree of similarity.

3.2.5.3. Match Score .

This score has often been referenced in literature, and is sometimes just called the *Score* when it is used with the *PFM* as in [1], or *Sequence Score* as in [34], or *Match Score* as in [36]. The match score of a neighbor or substring w satisfies

$$Match\ Score(w) = \sum_{i=w[j], j=1}^{l} PWM[i][j] \tag{3.6}$$

where l is the length of w and PWM is the Position Weight Matrix for a collection of neighbors. This score calculates how much a substring is matched with a PWM matrix.

3.2.6. Algorithm Time Complexity

When searching for the neighbors of a substring s_i in at least q' or q sequences, the qSMF applies a technique that can considerably reduce the execution time. The qSMF algorithm keeps track of not only the sequences where s_i is found, but also the number of sequences where s_i is NOT found. For instance, suppose a data sample contains 100 sequences, and q' is set at 80%. Then, the qSMF records whether or not the substring s_i is located in a sequence. If s_i is not found in 21 sequences, the algorithm stops its search and ignores this substring. Indeed, the q' constraint will not be satisfied even if s_i is found in the remaining 79 sequences. As a result, the qSMF search time will be reduced significantly. The same technique is applied when searching for a consensus motif in the entire dataset during the second pass.

The qSMF algorithm (Algorithm 2) consumes most of its time through the **for** loop (steps 4-18) of the first pass and the **for** loop (steps 20-35) of the second pass. During the first pass and within an iteration of the **for** loop starting at step 3, each substring s_i of the *Selected Sequence SS* is compared to all $(n-l+1)$ substrings of at most $q't'$ sequences of the dataset sample. Searching $O(q't')$ sequences of D' for a string s_i requires $O((n-l+1)q't'l)$ time for step 7; steps 8-11 take $O(lN)$ time leading to $O((n-l+1)q't'l)+O(lN)$ where N is the maximum number of neighbors retrieved, and is bounded as $O((n-l+1)t)$. Thus, the first pass consumes $O((n-l+1)q't'l)+O(lN)$ time. The *FirstPassNominatedMotifs* list is generated from the qSMF first pass with length P. Searching $O(qt)$ sequences of D for a *consensus* requires $O((n-l+1)qtl)$ time in step 22; steps 23-31 take $O(lP)$ time leading to $O((n-l+1)qtl)+O(lP)$ where P is the maximum number of nominated motifs generated

from the first pass, and is bounded as $O((n-l+1)q't')$. Thus, the second pass consumes $(n-l+1)qtl + O(lP)$ time.

As a result, for both passes, $O((n-l+1)q't'\ [(n-l+1)q't'l + lN + (n-l+1)qtl + lP])$ time may be needed, which is $O((n-l+1)^2(q't')^2l) + O((n-l+1)q't'lN) + O((n-l+1)^2q'qt'tl) + O((n-l+1)q't'lP) = O((n-l+1)^2q'qt'tl)$.

The sort in line 37 takes $O(N'logN')$ time, where N' is the number of motifs in the *SecondPassNominatedMotifs* list. Since each s_i can lead to one consensus motif, this may yield $O(n-l+1)$ nominated motifs per *Selected Sequence (SS)*. Thus, N' is bounded as $O((n-l+1)q't')$ over $O(q't')$ selected sequences. The final algorithm complexity is $O((n-l+1)^2q'qt'tl) + O(N'logN') = O(n^2q'qt'tl)$.

Designing both the SMF and qSMF algorithms will enable the researchers to locate motifs in smaller and larger DNA datasets. The results will be presented and discussed in the next chapter.

Algorithm 2 Quorum Strong Motif Finder Algorithm (qSMF)

Input: D, D', l, d, q, q', k
Output: *Final Motifs*

Set bound for Hamming distance (r)
Generate a *Shuffled Seqs* list of D'
for each SS in *Shuffled Seqs* **do** ▷ 1^{st} Pass
    for each substring s_i in SS, $1 \leq i \leq n - l + 1$ **do**
        $Neighbors \leftarrow [\,]$
        if $CS(s_i) > CS_{threshold}$ **then**
            if s_i found in at least $q't'$ sequences at $d_H \leq r$ **then**
                Add all found substrings similar to $Neighbors$
                Calculate *PFM* for *Neighbors*
                Generate M (consensus motif) from $Neighbors$
                Add M to the *FirstPassNominatedMotifs* list
            else
                Discard s_i
            end if
        else
            Discard s_i
        end if
    end for
    if *FirstPassNominatedMotifs* is not empty **then** ▷ 2^{nd} Pass
        for each *consensus* in *FirstPassNominatedMotifs* **do**
            NeighborsInfo $\leftarrow [\,]$
            if *consensus* found in at least qt seqs at $d_H \leq r$ **then**
                Add all found similar substrings to *NeighborsInfo*
                Refine *NeighborsInfo* based on distance
                Calculate *PFM* for *NeighborsInfo*
                Calculate *PWM* for *NeighborsInfo*
                Refine *NeighborsInfo* based on Match Scores
                Calculate *PFM* for *NeighborsInfo*
                Generate M from *NeighborsInfo*
                Compute Motif Strength Score *MSS* of M
                Add M to the *SecondPassNominatedMotifs* list
            else
                Discard *consensus*
            end if
        end for
        if *SecondPassNominatedMotifs* is not empty **then**
            Sort *SecondPassNominatedMotifs* list based on N
            Final Motifs $\leftarrow$ Top motifs (up to k) of the
                SecondPassNominatedMotifs
            return *Final Motifs*
        end if
    end if
end for
return *Final Motifs* $\leftarrow [\,]$

CHAPTER 4

EXPERIMENTAL RESULTS

Chapter 4 presents experimental results of the algorithms SMF and qSMF in Sections 4.1 and 4.2, respectively. The SMF algorithm is tested on both simulated and real datasets, while the qSMF is tested only on real datasets. Experimental results of both algorithms were obtained on a DELL laptop with Intel(R) Core(TM) i7-7500U CPU @ 2.70 GHz/2.90 GHz and 12.0 GB RAM.

4.1. Experimental Results of SMF

The SMF algorithm is compared with the recent APMotif (approximate) algorithm [47] (2015), and with the three distribution choices of the MEME algorithm [40]. The distribution choices are One Occurrence Per Sequence (OOPS), Zero or One Occurrence Per Sequence (ZOOPS), and Any Number of Repetition (ANR). The OOPS assumes that each sequence has exactly one occurrence of a motif, the ZOOPS assumes that each sequence has at most one motif, and the ANR searches for any number of motif occurrences. Therefore, the ANR is considered the most practical choice since it does not take prior knowledge about the data into account. The comparison focuses on algorithm execution time and prediction accuracy.

4.1.1. Experimental Results on Simulated Data

In order to assess the SMF performance, 23 test datasets were generated for 23 challenging problem instances (l, d), where a problem is called *challenging* according to Buhler and Tompa's [16] classification into solvable and unsolvable problems, while considering

the expected number of spurious motifs that can occur by chance in random DNA sequences. The last solvable problem instance was considered a challenge problem for this research. The 23 problem instances are: *(8,1), (9,1), (10,2), (11,2), (12,3), (13,3), (14,4), (15,4), (16,5), (17,5), (18,6), (19,6), (20,7), (21,7), (22,8), (23,8), (24,9), (25,10), (26,10), (27,11), (28,11), (29,12)* and *(30,12)*. For further details about how these problem instances were selected, see Appendix B. For each of these, a random dataset of 10 sample files was generated over the DNA characters $\{A, C, G, T\}$ with equal probability. A sample file contains $t = 20$ sequences each of length $n = 600$. For each file, a consensus motif M of length l was also generated at random, and twenty instances of M were created by randomly mutating the M with variations $\leq d$ at random positions. The instances of M were implanted in the different sequences at random locations. The main goal of the algorithms is to return the consensus motif M from the implanted neighbors with their locations. The results over these files for every problem instance were averaged. This dataset creation configuration has been used by many authors when testing their algorithms. Figure 4.1 depicts the average execution times for the MEME, APMotif and SMF algorithms. The execution time is averaged over all 23 problem instances for MEME and SMF. The APMotif algorithm's average time is 2.5 hours over 14 problem instances. It is expected to be higher if the remaining problem instances are timed. The MEME algorithm's average execution times for OOPS, ZOOPS, and ANR are 1.7, 3.4, and 7.6 seconds, respectively. The SMF algorithm achieves 3.5 seconds. In Figure 4.2, the average over all problem instances for each performance metric is given for the three algorithms. The MEME algorithm with the OOPS choice achieves higher prediction accuracy than ZOOPS, and ZOOPS scores higher than ANR. The SMF algorithm achieves performance results at the level of the MEME (OOPS), whereas the APMotif scores close to the MEME (ZOOPS). The APMotif performance tests covered 14 problem instances.

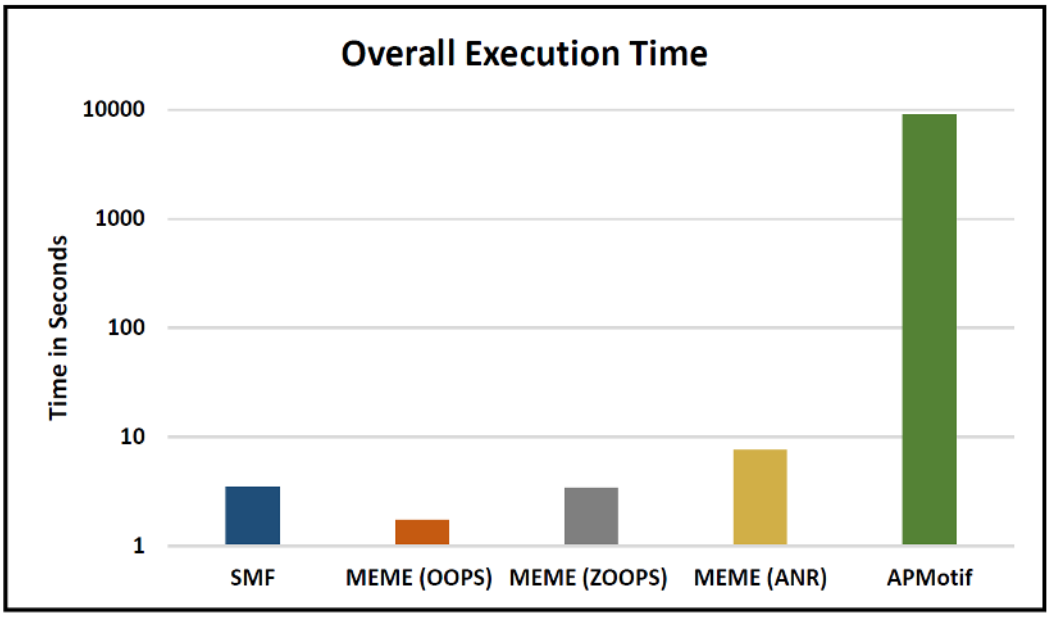


Figure 4.1: Overall Averaged Execution Time

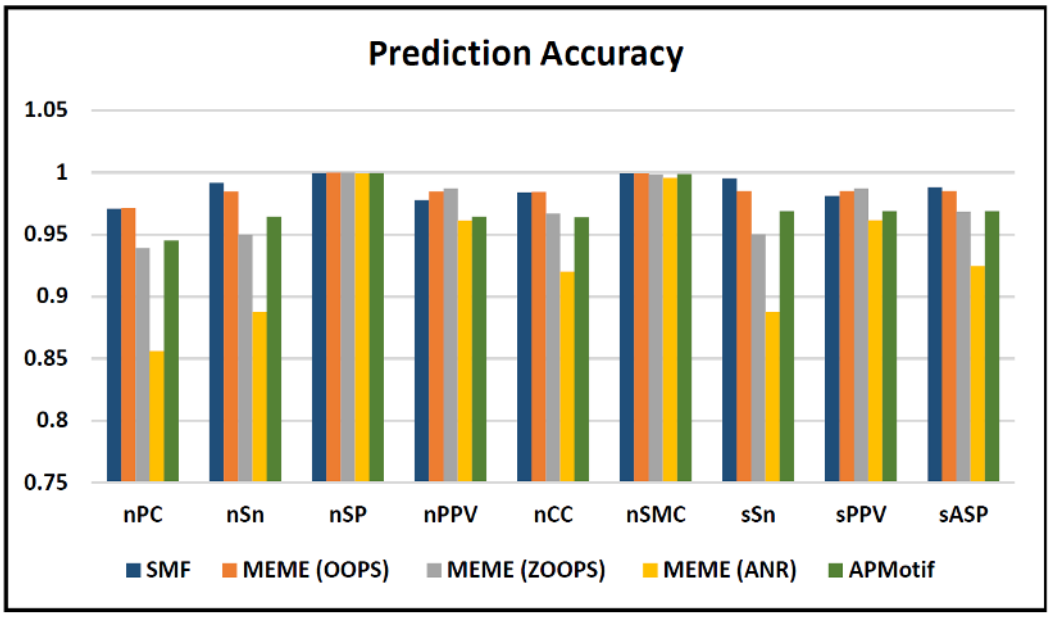


Figure 4.2: Overall Averaged Execution Time

4.1.2. Experimental Results on Real Data

The SMF algorithm was tested on real datasets that were used by Blanchette and Tompa in [131], when they tested their FootPrinter algorithm. We used seven datasets: *c-fos*, *c-myc*, *growth hormone*, *histone H1*, *insulin*, *interleukin-3*, and *metallothionein*. These are freely available at `http://bio.cs.washington.edu/supplements/FootPrinter`. The website contains nine datasets. We did not consider *c-myc second intron* and *c-fos first intron* because their sequences are of different lengths. While the proposed SMF algorithm can currently process only DNA sequences of the same length, so a modification was planned to address this.

The purpose of using the same datasets is to compare the top k motifs returned by the SMF with published motifs. The FootPrinter algorithm also returns multiple motifs instead of just one. Other papers that have used some of these datasets are [132], [16] and [6]. In [132], the detected motifs were compared with only one motif that was listed in [131]. In [6], only insulin, metallothionein, and c-fos were used.

Figures 4.3-4.9 show the common substrings in each dataset when searching for the (l, d) instance. For most datasets, a large number of common substrings were found, especially for *interleukin-3*, whereas no common substrings were found for the *growth hormone* and *metallothionein* datasets using the default SMF settings of $r = d$. If a dataset does not contain any common DNA pieces, it is advised to increase the value of r, which will allow more neighbors to be collected and more sequences to be contributed, although less powerful motifs may be returned. The results reported for *growth hormone* and *metallothionein* were obtained with $r = d + 1$. Table 4.1 lists the final SMF motifs for each dataset, their MSS score and the number of neighbors (N) (left column of Table 4.1). The performance metrics cannot be computed since the exact locations of the real motifs were unknown. The SMF

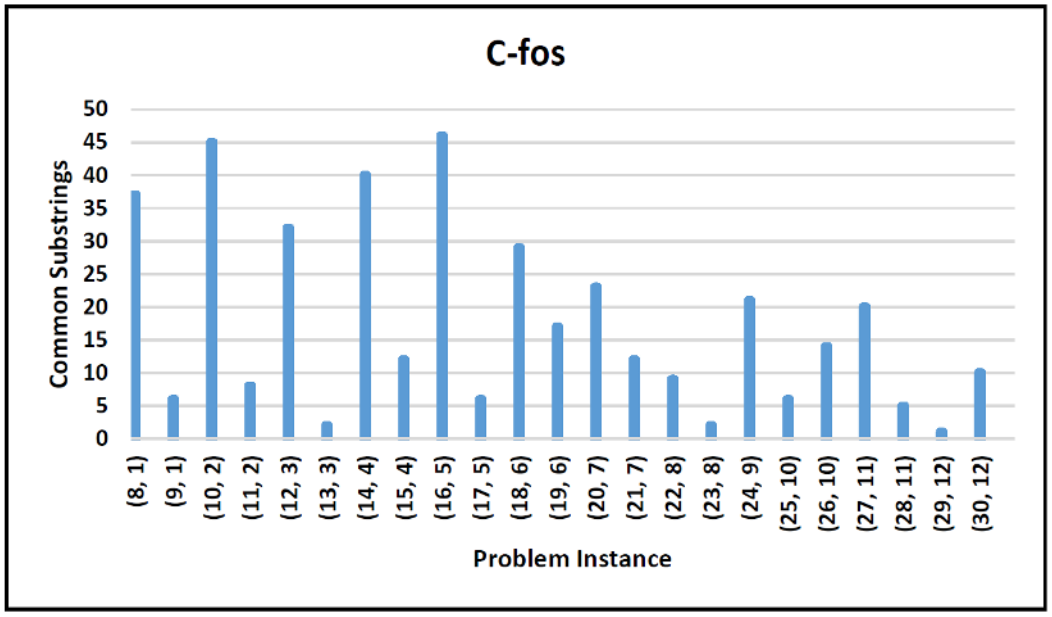


Figure 4.3: Number of Common Substrings for the C-fos Dataset

discovers multiple motifs for each set, and most of which are similar to the published motifs (See the right column of Table 4.1). The (id) references the motif number in [131].

The erratic behavior of the number of common substrings in the Figures 4.3 to 4.9 is related to Table 3 in Appendix B. When examining the values of $E_t(l, d)$, there are irregular jumps between the values. For example, the value of $E_t(8, 1)$ is 8.8E-10, while the value of $E_t(9, 1)$ is 1.46E-19. The jump in the values indicates the larger number of common substrings for problem instance (8,1), and the smaller number of common substrings of the problem instance (9, 1). A substring of length 8 with an allowed mutation 1 is more likely to occur by chance than a substring of length 9 with the same allowed mutation. Knowing the relationship between the motif length and the allowed mutation is important when deciding the maximum allowed mutation for each motif length.

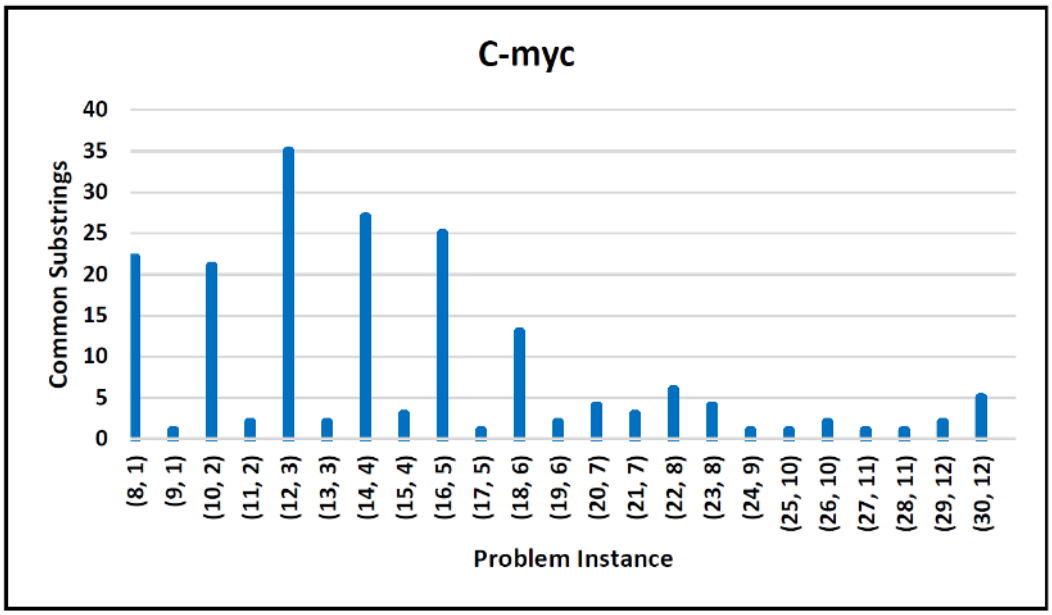


Figure 4.4: Number of Common Substrings for the C-myc Dataset

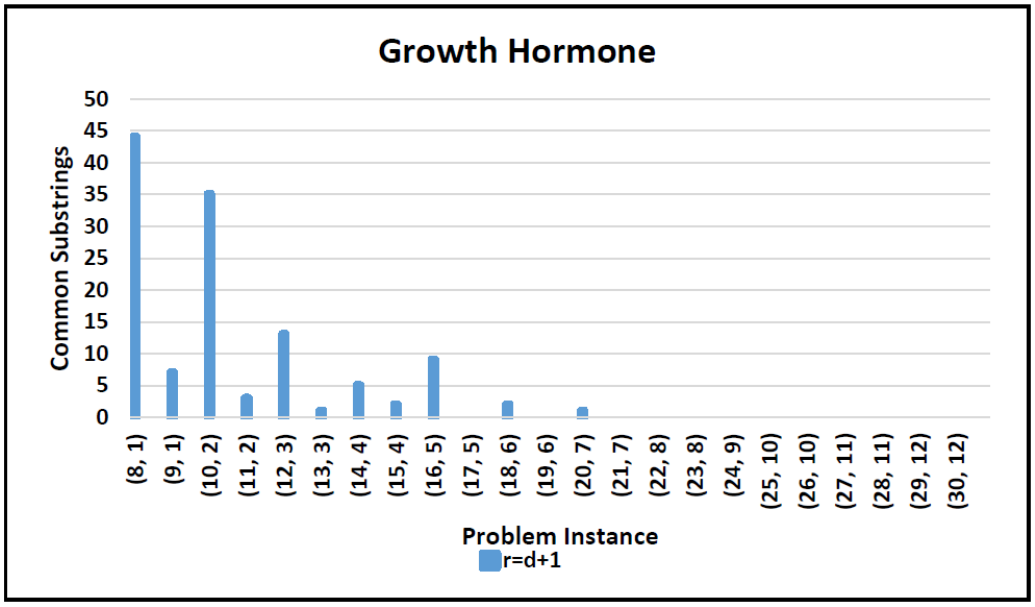


Figure 4.5: Number of Common Substrings for the Growth Hormone Dataset

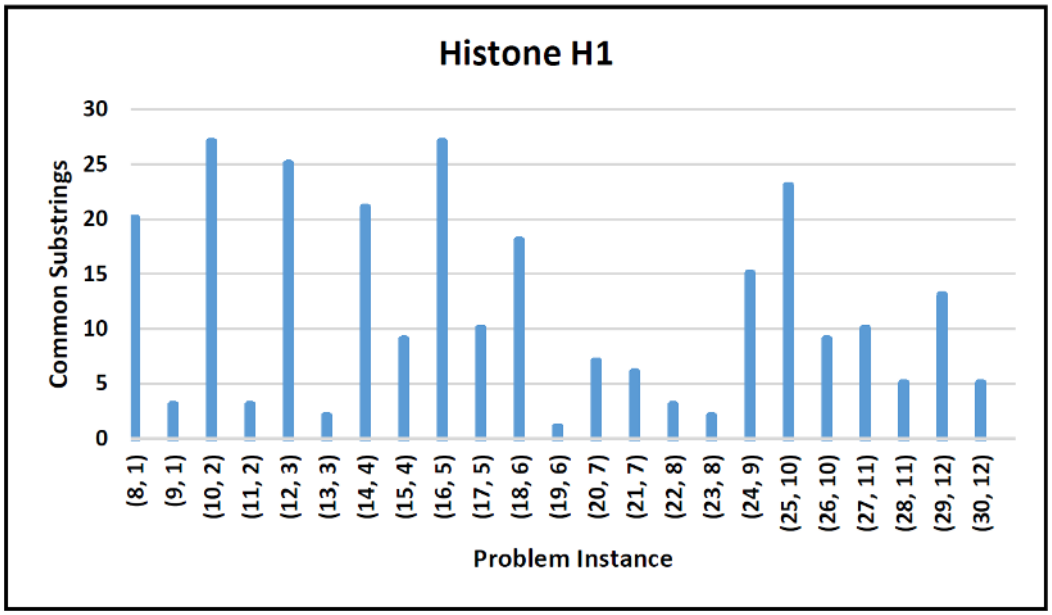


Figure 4.6: Number of Common Substrings for the Histone H1 Dataset

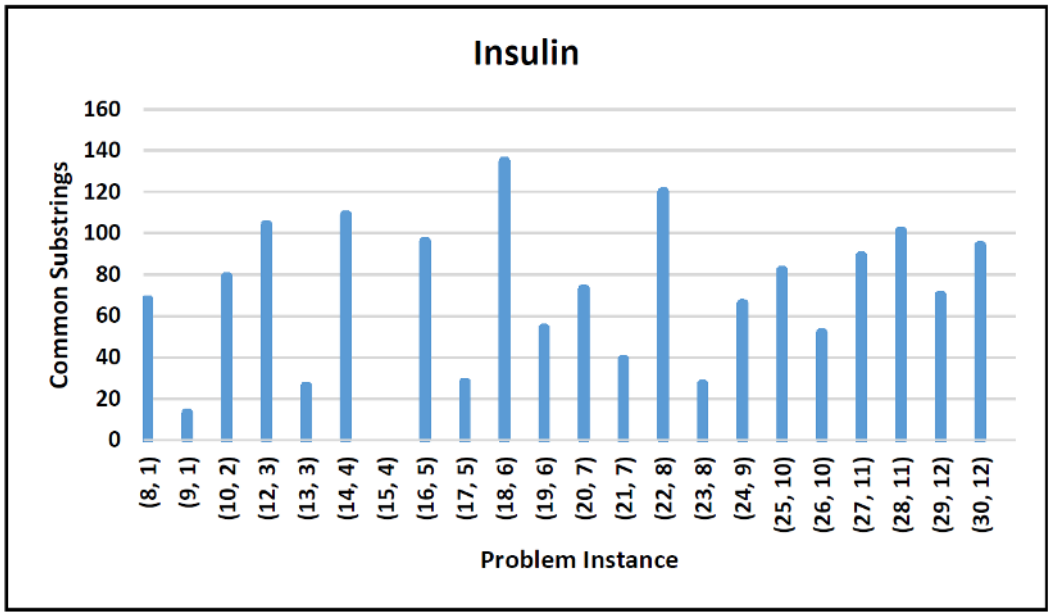


Figure 4.7: Number of Common Substrings for the Insulin Dataset

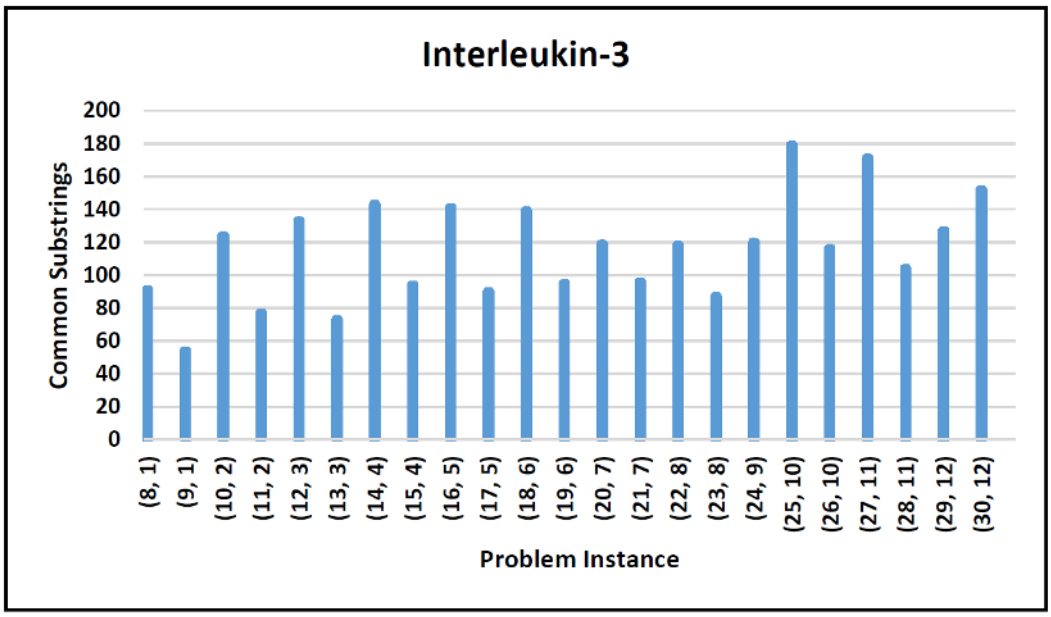


Figure 4.8: Number of Common Substrings for the Interleukin-3 Dataset

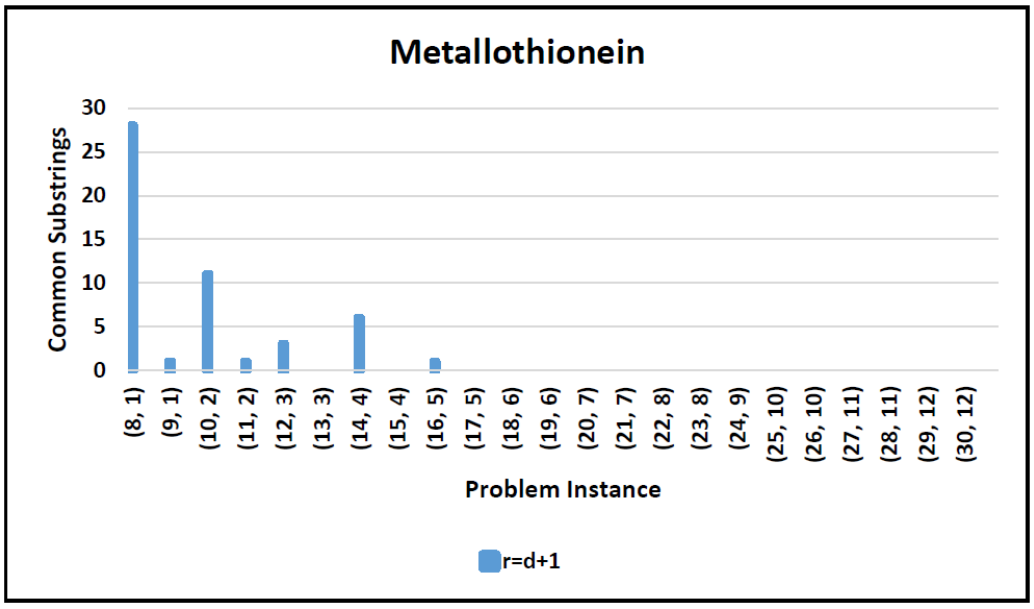


Figure 4.9: Number of Common Substrings for the Metallothionein Dataset

Table 4.1: Top Motifs Returned by SMF When Applied on Real Datasets

Dataset$_{t\times n}$	Final SMF Motifs $(l)(N)(MSS)$	Motif $(l)(id$ in [131])
c-fos$_{6\times 702}$	ACAGGATG (8)(6)(1)	CACAGGATGTcc (10)(5)
	CCATATTAGGA (11)(6)(0.97)	
	CCGCGGCCCC (10)(6)(0.95)	
c-myc$_{7\times 1000}$	CCCTCCCC (8)(8)(0.97)	ccaCCCTCCCC (8)(6)
	GTTTATTC (8)(9)(0.93)	aGTTTATTC (8)(1)
	CAGCAAAT (8)(9)(0.92)	
	GCAGCAAA (8)(8)(0.92)	
growth hormone$_{16\times 380}$	TATAAAAA (8)(25)(0.89)$_{(r=d+1)}$	cagggTATAAAAAGggc (9)(7)
	ATGCATTA (8)(31)(0.81)$_{(r=d+1)}$	
histon H1$_{4\times 650}$	ACAAAAGT (8)(4)(1)	gAAACAAAAGTtt (10)(2)
	CAATCACC (8)(4)(0.97)	CAATCACCAC (10)(1)
	GCGGCTCCTCTC (12)(4)(0.9)	
insulin$_{8\times 500}$	CTATAAAG (8)(8)(1)	CTATAAAGcc (8)(3)
	TCAGCCCC (8)(8)(1)	tcagccccaGCCATCTGCC (10)(2)
	GCCATCTGC (9)(8)(1)	tcagccccaGCCATCTGCC (10)(2)
	TTAAGACTCTAAT (13)(8)(0.97)	gttAAGACTCTAAtgacc (10)(1)
interleukin-3$_{6\times 490}$	ACTAGAAA (8)(6)(1)	TTGAGTACTagaaagt (8)(1)
	CTATGGAGGTTCCATGTCAGATAAAG (26)(6)(1)	GTCTGTGGTTTtCTATGGAGGTTCCATGT CAGATAAAG (8)(3)
	TGTGGTTTGCTATGGAGGTTCCATGTCAGA(30)(6)(0.99)	GTCTGTGGTTTtCTATGGAGGTTCCATGT CAGATAAAG (8)(3)
metallothionein$_{26\times 590}$	GTGTGCAG (8)(52)(0.84)$_{(r=d+1)}$	CGTGTGCAggc (8)(3)
	TGTGTGCA (8)(46)(0.84)$_{(r=d+1)}$	CGTGTGCAggc (8)(3)
	TTGCACCC (8)(50)(0.82)$_{(r=d+1)}$	tGCGCCCGG (8)(5)

4.2. Experimental Results of qSMF

4.2.1. Experimental Results on Real Data

The proposed algorithm was tested using eight Homo Sapiens datasets selected from the ENCODE TF ChIP-seq data as shown in Table 4.2. These datasets are part of a larger group of datasets that were investigated in [133]. The authors of [133] performed a systematic motif analysis of 427 ChIP-Seq datasets grouped into 84 factor groups using five established motif discovery tools that included the AlignACE [58], MDscan [60], MEME [40], Trawler [57], and Weeder [18]). They provide a web interface for browsing the discovered results and their motifs that are known or published in the literature along with their enrichments at `http://compbio.mit.edu/encode-motifs/`

These eight datasets have been also used in [80] to test their proposed sampling algorithm, SamSelect. The goal of their algorithm was to sample a large dataset into much smaller sample(s). The output of the SamSelect algorithm is a collection of samples of these datasets, and analyze the performance of their algorithm by applying the FMotif algorithm [64] to search for motifs in these samples. The SamSelect authors illustrated which samples enable the FMotif algorithm to return known motifs.

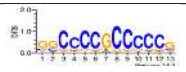
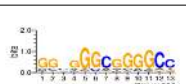
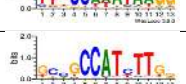
Our algorithm does not involve any pre-sampling, but we have utilized their sample datasets to compare the results of the qSMF algorithm with corresponding results in both [80] and [133].

Table 4.2 lists the real datasets used, with the number of sequences in each one (t). Each dataset is referenced by its corresponding transcription factor. The third column (TS) represents the total number of samples generated from the SamSelect algorithm by its authors of each dataset. The next column (S#) is the sample number that we used in our experiments. The qSMF algorithm uses these samples to nominate motifs during the first

pass, then applies its retrieved motifs to search for their instances in the entire dataset. The fifth column (MR) is the motif rank. This index was used in [133] and [80] to order the obtained motifs. The T_{FMotif} in the sixth column is the total time taken by the FMotif on all dataset samples to find motifs. The seventh column shows the time taken by the FMotif for a single sample, which is an average time generated by dividing the total time for all samples in the column 6 by the number of samples in the column 3, assuming that all samples are of a similar size. In the last column, the sequence logo of the predicted motif is drawn based on the substrings similar to the motif in the entire dataset at the Hamming distance within $d/2$ from the motif. Note that the T_{FMotif} does not include the time consumed for searching the entire dataset for a motif found from a sample dataset. The FMotif was tested on the entire sample set (not just a single sample) for reasons of testing the sampling performance. In column 8, T_{qSMF} is the total time taken by the qSMF for both passes. In particular, the time taken for searching the whole dataset and its sample is generated by taking the average of five runs. The predicted motif by the qSMF is illustrated in column 9. The tenth column lists the published motif known in the literature followed by the discovered motif (in column 11) through the five motif discovery methods in [133]. The results show that the qSMF algorithm is able to predict most of the known motifs in a short time.

It should be mentioned that the qSMF and FMotif algorithms are executed on different computer platforms. The experimental results of the qSMF were obtained on a DELL laptop with Intel(R) Core(TM) i7-7500U CPU @ 2.70 GHz/2.90 GHz and 12.0 GB RAM, while the Fmotif algorithm was tested on a 2.60 GHz 24-core platform with 64 Gbyte of memory (where Samselect and FMotif were executed on a single core).

Table 4.2: Results of Eight Homo Sapiens Real Datasets Selected from the ENCODE TF ChIP-Seq Dataset

Dataset	t	TS	S#	MR	T_{FMotif} for all samples	Time per single sample	T_{qSMF} per single sample	Predicted Motif by qSMF	Published Motif in [133]	Discovered Motif by the five algorithms	Discovered Motif by FMotif in [80]
egr1	15,400	5	1	1	11.7 m	2.34 m	55.70 s				
elf1	8611	9	4	4	32.9 m	3.66 m	41.17 s				
hnf4	11,045	14	3	1	23.0 m	1.64 m	27.37 s				
myc	4542	5	3	1	12.5 m	2.5 m	8.64 s				
nfy	9781	2	2	1	4.6 m	2.3 m	100.52 s				
sp1	14,779	4	2	6	9.0 m	2.25 m	167.67 s				
srf	4903	24	1	1	24.7 m	1.03 m	8.29 s				
yy1	2077	17	1	1	30.0 m	1.76 m	15.40 s				

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

After testing the SMF and qSMF algorithms on dedicated datasets, multiple conclusions have been reached.

1. Both the SMF and qSMF algorithms implement strategies based on collecting neighbors of a substring. The SMF is intended for small datasets (promotor regions), while the qSMF targets large datasets (ChIP-seq data).

2. The performance of the SMF is compared to that of APMotif and MEME with respect to execution time and prediction accuracy.

   (a) The time comparisons show that the SMF is faster than APMotif and MEME (ANR) and similar in speed to MEME (ZOOPS). The MEME algorithm with choice OOPS is the fastest but is not practical if no prior knowledge is available.

   (b) The prediction accuracy results show that the SMF outperforms the APMotif, and performs at the level of the best prediction accuracy of MEME (OOPS), notwithstanding that the SMF is not given a-priori information.

3. Dealing with real datasets is more difficult than dealing with simulated datasets since real datasets may have high similarity or no similarity between sequences, and no prior knowledge is available before running the algorithms. When testingthe SMF on real DNA datasets, the experiments show that the SMF results agree with published motifs.

4. From the experimental results of the SMF when applied to real datasets, many motifs with an MSS score of 1 are obtained. This score means that all collected neighbors are identical, which corresponds to DNA sequences that have common, exact motifs.

5. The time complexity of SMF is quadratic with respect to the length of the sequences and to the number of sequences. In comparison, the MEME algorithm is quadratic with respect to the number of characters and cubic with respect to the number of sequences [134].

6. Two causes of a possible decreased SMF performance are:

   (a) Unreal neighbors that are at the same distance from the original motif as the real neighbors. This produces extra starting locations that affect the performance measures. These measures are computed depending on the accuracy of the motif location predictions.

   (b) Sequences implanted with mutated motifs that are at greater distances from the original motifs yield lower performance, and because the average of five runs for each file was taken, the average may be affected by these lower values.

7. The top k motifs that are returned by the SMF can be part of each other with different values of the MSS score. This is not a concern when applied on simulated data, but is an important issue when applied to real datasets with high similarity between sequences.

8. The expected number of motifs E_t impacts the SMF algorithm execution time, but its effect is reduced when the allowed distance r is decreased to d. If the value of r is set to $2d$, the effects of E_t may be more obvious. The execution time of the proposed

algorithm is affected significantly by the number of sequences that contain zero or a small number of mutations.

9. The r parameter is an important factor in both the SMF and qSMF. It has effects on:

   (a) Sequences that contribute to solutions. Larger r values would admit more sequences even with implanted motifs that are far from the original.

   (b) Scoring value MSS. Larger values of r allow farther neighbors to be added to the $Neighbors$ list. Farther neighbors correspond to a lower MSS value.

   (c) Number of substrings found in all sequences. Larger r values yield more substrings that satisfy the condition to be found in other sequences.

   (d) Algorithm performance and accuracy. Larger r values allow farther final motifs to be returned by the algorithms, which would reduce the accuracy.

   (e) Running time. More sequences, substrings, and neighbors lead to more processing time, which would increase the overall running time.

10. The qSMF algorithm returns a list of highest ranked, strongest, motifs occurring in at least q percent of the data sequences. The qSMF algorithm is capable of predicting many of the published results with motifs of rank 1 when applied to real DNA datasets, and is an approximate algorithm that returns solutions successfully within a reduced execution time.

11. When comparing the qSMF algorithm with FMotif, the experimental results show that the qSMF is faster and returns predicted motifs similar to results in the literature and to motifs discovered by the ENCODE project tool using the established motif finding algorithms of AlignACE, MEME, MDscan, Trawler, and Weeder.

12. The performance of the qSMF depends on the quality of the sampled dataset. The quality of performance increases as the sampled dataset contains more of the target dataset motifs.

13. The proposed simple statistical measure, Motif Strength Score (MSS), can be used to rank motifs of different lengths.

14. There is no perfect algorithm. Each algorithm has its own advantages and drawbacks. In general, the SMF and qSMF algorithms produce good accuracy and are practical with respect to execution time.

5.2. Future Work

After reviewing the findings of the current research, several short- and long-term projects can be done in the future to extend the scope and performance of the algorithms.

1. Make both the SMF and qSMF algorithms available online either through a portal as software or by converting them into a web service (web-based algorithms).

2. Create simulated datasets with a more realistic nucleotide distribution (not uniform random, but with GC biased distribution). Investigate the obtained performance for the biased distribution.

3. Continue comparing the proposed scoring measure MSS with other statistical measures such as p-value, z-score, and information content. This comparison is needed for further effectiveness testing of the suggested score.

4. Implement enhancements to the APMotif algorithm of [47] to address long and erratic execution times. We suggest using our analysis on collecting neighbors, and optimizing the value of r to decrease the cluster sizes in APMotif (instead of using $r = 2d$). We expect this enhancement will decrease the execution time and retrieve more accurate results. Furthermore, our scoring function MSS could be used to score APMotif solutions. The performance of the new algorithm can be evaluated.

5. Parallelize SMF and qSMF by distributing the dataset sequences over multiple nodes and letting each node perform the strategies locally, in order to support very large datasets effectively to reduce execution time.

6. For the qSMF algorithm, add the p-value as a motif scoring function during the filtering process [135]. The K_2 measure of [33] can also be investigated.

7. Investigate new techniques and algorithms for sampling, or utilize existing tools to generate dataset samples.

BIBLIOGRAPHY

[1] N. C. Jones and P. A. Pevzner, *An Introduction to Bioinformatics Algorithms*. Cambridge, Massachusetts: Massachusetts Institute of Technology, 2004, pp. 83–123.

[2] Alberts, Johnson, Lewis, Raff, Roberts, and Walter, *Molecular Biology of The Cell*. 2015, pp. 597–638, ISBN: 9780815341109. DOI: 10.1017/CBO9781107415324.004.

[3] G. B. Singh, *Fundamentals of bioinformatics and computational biology : methods and exercises in MATLAB*. 2015, p. 339, ISBN: 9783319114033.

[4] M. K. Das and H.-K. Dai, "A survey of DNA motif finding algorithms", *BMC bioinformatics*, vol. 8 Suppl 7, no. Suppl 7, S21, 2007, ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-S7-S21.

[5] G. Sandve and F. Drabløs, "A survey of motif discovery methods in an integrated framework", *Biology Direct*, vol. 1, no. 1, p. 11, Apr. 2006, ISSN: 17456150. DOI: 10.1186/1745-6150-1-11. [Online]. Available: http://biologydirect.biomedcentral.com/articles/10.1186/1745-6150-1-11.

[6] Chao-Wen Huang, Wun-Shiun Lee, and Sun-Yuan Hsieh, "An Improved Heuristic Algorithm for Finding Motif Signals in DNA Sequences", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 4, pp. 959–975, Jul. 2011, ISSN: 1545-5963. DOI: 10.1109/TCBB.2010.92. [Online]. Available: http://ieeexplore.ieee.org/document/5582077/.

[7] F. Zambelli, G. Pesole, and G. Pavesi, "Motif discovery and transcription factor binding sites before and after the next-generation sequencing era", *Briefings in Bioinformatics*, vol. 14, no. 2, pp. 225–237, Mar. 2012, ISSN: 1477-4054. DOI: 10.1093/bib/bbs016. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22517426http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3603212https://academic.oup.com/bib/article-lookup/doi/10.1093/bib/bbs016.

[8] P. D'Haeseleer, "What are DNA sequence motifs?", *Nat Biotech*, vol. 24, no. 4, pp. 423–425, Apr. 2006, ISSN: 1087-0156. DOI: 10.1038/nbt0406-423.

[9] 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, Jan. 2007, ISSN: 1672-0229. DOI: 10.1016/S1672-0229(07)60023-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1672022907600230.

[10] F. Y. Chin and H. C. Leung, "Voting Algorithms for Discovering Long Motifs", *Proceedings of the 3rd Asia-Pacific Bioinformatics Conference*, pp. 261–271, 2005, ISSN: 17516404. DOI: 10.1142/9781860947322{_}0026.

[11] G. D. Stormo, "DNA binding sites: Representation and discovery", *Bioinformatics*, vol. 16, no. 1, pp. 16–23, 2000, ISSN: 13674803. DOI: 10.1093/bioinformatics/16.1.16. [Online]. Available: https://academic.oup.com/bioinformatics/article-abstract/16/1/16/243066.

[12] W. W. Wasserman and A. Sandelin, "Applied bioinformatics for the identification of regulatory elements", *Nature Reviews Genetics*, vol. 5, no. 4, pp. 276–287, Apr. 2004, ISSN: 1471-0056. DOI: 10.1038/nrg1315. [Online]. Available: http://www.nature.com/articles/nrg1315.

[13] R Staden, "Computer methods to locate signals in nucleic acid sequences.", *Nucleic acids research*, vol. 12, no. 1 Pt 2, pp. 505–19, Jan. 1984, ISSN: 0305-1048. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/6364039http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC321067.

[14] T. D. Schneider and R. M. Stephens, "Sequence logos: a new way to display consensus sequences.", *Nucleic acids research*, vol. 18, no. 20, pp. 6097–100, Oct. 1990, ISSN: 0305-1048. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/2172928http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC332411.

[15] S. Rajasekaran, "Algorithms for motif search", in *Handbook of Computational Molecular Biology*, S. Aluru, Ed., 1st ed., Boca Raton: Chapman and Hall/CRC, 2005, ch. 37, pp. 37–1, ISBN: 0849385970. DOI: 10.1201/9781420036275.ch37.

[16] J. Buhler and M. Tompa, "Finding Motifs Using Random Projections", *JOURNAL OF COMPUTATIONAL BIOLOGY*, vol. 9, no. 2, 2002.

[17] T. El Falah, M. Elloumi, and T. Lecroq, "Motif finding algorithms in biological sequences", in *Algorithms in Computational Molecular Biology : Techniques, Approaches, and Applications*, M. Elloumi and A. Y. Zomaya, Eds., Hoboken, NJ: John Wiley & Sons, 2011, ch. 18, pp. 385–396, ISBN: 0470505192.

[18] G. Pavesi, G. Mauri, and G. Pesole, "An algorithm for finding signals of unknown length in DNA sequences", *BIOINFORMATICS*, vol. 17, no. 1, pp. 207–214, 2001. [Online]. Available: http://www.cs.duke.edu/courses/fall07/cps296.3/pavesi.2001.pdf.

[19] M. F. Sagot, "Spelling approximate repeated or common motifs using a suffix tree", *Lecture notes in computer science*, pp. 374–390, 1998, ISSN: 0302-9743. DOI: 10.1.1.39.3583.

[20] P. A. Pevzner and S. H. Sze, "Combinatorial approaches to finding subtle signals in DNA sequences", *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, vol. 8, pp. 269–278, 2000, ISSN: 1-57735-115-0.

[21] C.-E. Wang, "A Fast and Effective Heuristic Algorithm for Finding Quorum Planted (l, d, q)-Motifs", in *Int'l Conf. Bioinformatics and Computational Biology, BIOCOMP'18*, 2018, pp. 47–51, ISBN: 1601324715. [Online]. Available: https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/BIC3008.pdf.

[22] G. Pavesi, P. Mereghetti, G. Mauri, and G. Pesole, "Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes.", *Nucleic acids research*, vol. 32, no. Web Server issue, pp. 199–203, Jul. 2004, ISSN: 1362-4962. DOI: 10.1093/nar/gkh465. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/15215380http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC441603.

[23] G. Pavesi, F. Zambelli, and G. Pesole, "WeederH: an algorithm for finding conserved regulatory motifs and regions in homologous sequences", 2007. DOI: 10.1186/1471-2105-8-46. [Online]. Available: http://www.biomedcentral.com/1471-2105/8/46.

[24] "A Statistical Method for Finding Transcription Factor Binding Sites", Tech. Rep., 2000. [Online]. Available: www.aaai.org.

[25] H. Sun, Y. Yuan, Y. Wu, H. Liu, J. S. Liu, and H. Xie, "Tmod: toolbox of motif discovery", *BIOINFORMATICS APPLICATIONS NOTE*, vol. 26, no. 3, pp. 405–407, 2010. DOI: 10.1093/bioinformatics/btp681. [Online]. Available: http://www.fas.

[26] Y. Xu, J. Yang, Y. Zhao, and Y. Shang, "An improved voting algorithm for planted (l,d) motif search", *Information Sciences*, vol. 237, pp. 305–312, Jul. 2013, ISSN: 00200255. DOI: 10.1016/j.ins.2013.03.023.

[27] L. J. Korn, C. L. Queen, and M. N. Wegman, "Computer analysis of nucleic acid regulatory sequences.", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 74, no. 10, pp. 4401–5, Oct. 1977, ISSN: 0027-8424. DOI: 10.1073/PNAS.74.10.4401. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/270683http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC431950.

[28] C Queen, M. N. Wegman, and L. J. Korn, "Improvements to a program for DNA analysis: a procedure to find homologies among many sequences.", *Nucleic acids research*, vol. 10, no. 1, pp. 449–56, Jan. 1982, ISSN: 0305-1048. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/6174938http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC326145.

[29] M. Li, B. Ma, and L. Wang, "Finding Similar Regions In Many Strings", Tech. Rep., 1999, pp. 473–482. [Online]. Available: http://delivery.acm.org/10.1145/310000/301376/p473-li.pdf?ip=141.218.147.44&id=301376&acc=ACTIVESERVICE&key=B5D9E165A72B697C.724B5EC19F549CA5.4D4702B0C3E38B35.4D4702B0C3E38B35&__acm__=1546977994_94d5ae198b5048c6b50d40e4847b4360.

[30] J. M. Caldonazzo Garbelini, A. Y. Kashiwabara, and D. S. Sanches, "Sequence motif finder using memetic algorithm", *BMC Bioinformatics*, vol. 19, no. 1, p. 4, Dec. 2018, ISSN: 1471-2105. DOI: 10.1186/s12859-017-2005-1. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/29298679http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5751424https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-2005-1.

[31] G. D. Stormo, G. W. Hartzell, P. O. Brown, and D. Botstein, "Identifying protein-binding sites from unaligned DNA fragments.", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 86, no. 4, pp. 1183–7, Feb. 1989, ISSN: 0027-8424. DOI: 10.1073/pnas.86.4.1183. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/2919167http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC286650.

[32] G. B. Fogel, D. G. Weekes, G. Varga, E. R. Dow, H. B. Harlow, J. E. Onyia, and C. Su, "Discovery of sequence motifs related to coexpression of genes using evolutionary computation", *Nucleic Acid Research*, vol. 32, no. 13, pp. 3826–3835, 2004. DOI: 10.1093/nar/gkh713.

[33] J. C. Wootton and S Federhen, "Analysis of compositionally biased regions in sequence databases.", *Methods in enzymology*, vol. 266, pp. 554–71, 1996, ISSN: 0076-6879. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/8743706.

[34] Y. Yang, J. A. Robertson, Z. Guo, J. Martinez, C. Coghlan, and L. S. Heath, "MCAT: Motif Combining and Association Tool", *Journal of Computational Biology*, vol. 26, no. 1, pp. 1–15, Jan. 2019, ISSN: 1557-8666. DOI: 10.1089/cmb.2018.0113. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/30418034https://www.liebertpub.com/doi/10.1089/cmb.2018.0113.

[35] S. Schbath, "Statistics of motifs", Tech. Rep., 2006. [Online]. Available: https://www.webpages.uidaho.edu/~stevel/565/literature/StatisticsofMotifs.pdf.

[36] T. L. Bailey and M. Gribskov, "Combining evidence using p-values: application to sequence homology searches", *Bioinformatics*, vol. 14, no. 1, pp. 48–54, Feb. 1998, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/14.1.48. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/14.1.48.

[37] S. Sinha and M. Tompa, "Discovery of novel transcription factor binding sites by statistical overrepresentation", *Nucleic Acids Research*, vol. 30, no. 24, pp. 5549–5560, Dec. 2002, ISSN: 13624962. DOI: 10.1093/nar/gkf669.

[38] T. L. Bailey, N. Williams, C. Misleh, and W. W. Li, "MEME: discovering and analyzing DNA and protein sequence motifs.", *Nucleic acids research*, vol. 34, no. Web Server issue, pp. 369–73, Jul. 2006, ISSN: 1362-4962. DOI: 10.1093/nar/gkl198. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16845028http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1538909.

[39] T. L. Bailey and C. Elkan, "Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization", *Machine Learning*, vol. 21, no. 1/2, pp. 51–80, 1995, ISSN: 08856125. DOI: 10.1023/A:1022617714621.

[40] T. L. Bailey and C. Elkan, "Fitting a Mixture Model by Expectation Maximization to Discover Motifs in Bipolymers", *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pp. 28–36, 1994, ISSN: 1553-0833. DOI: citeulike-article-id:878292.

[41] C. E. Lawrence and A. A. Reilly, "An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences", *Proteins: Structure, Function, and Bioinformatics*, vol. 7, no. 1, pp. 41–51, 1990, ISSN: 10970134. DOI: 10.1002/prot.340070105.

[42] M Aitkin and D. B. Rubin, "Estimation and hypothesis testing in mixture models", *Journal of the Royal Statistical Society, Series B*, vol. 47 no. 1, pp. 67–75, 1985. [Online]. Available: https://www.jstor.org/stable/2345545.

[43] T. L. Bailey and C Elkan, "The value of prior knowledge in discovering motifs with MEME.", *Proceedings. International Conference on Intelligent Systems for Molecular Biology*, vol. 3, pp. 21–9, 1995, ISSN: 1553-0833. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/7584439.

[44] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble, "MEME SUITE: tools for motif discovery and searching", *Nucleic Acids Research*, vol. 37, no. Web Server, W202–W208, Jul. 2009, ISSN: 0305-1048. DOI: 10.1093/nar/gkp335.

[45] T. L. Bailey, J. Johnson, C. E. Grant, and W. S. Noble, "The MEME Suite.", *Nucleic acids research*, vol. 43, no. W1, pp. 39–49, Jul. 2015, ISSN: 1362-4962. DOI: 10.1093/nar/gkv416. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/25953851http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4489269.

[46] A. Dempster, N. Laird, D. R.J.o.t.r. s. society, and u. 1977, "Maximum likelihood from incomplete data via the EM algorithm", *JSTOR*, [Online]. Available: https://www.jstor.org/stable/2984875.

[47] C. Sun, H. Huo, Q. Yu, H. Guo, and Z. Sun, "An Affinity Propagation-Based DNA Motif Discovery Algorithm", 2015. DOI: 10.1155/2015/853461.

[48] J. Zhu and M. Q. Zhang, "SCPD: a promoter database of the yeast Saccharomyces cerevisiae", *Bioinformatics*, vol. 15, no. 7, pp. 607–611, Jul. 1999, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/15.7.607.

[49] M. Blanchette and S. Sinha, "Separating real motifs from their artifacts", *Bioinformatics*, vol. 17, no. Suppl 1, S30–S38, Jun. 2001, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/17.suppl{_}1.S30. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/17.suppl_1.S30.

[50] S. Sinha and M. Tompa, "YMF: a program for discovery of novel transcription factor binding sites by statistical overrepresentation", *Nucleic Acids Research*, vol. 31, no. 13, pp. 3586–3588, Jul. 2003, ISSN: 1362-4962. DOI: 10.1093/nar/gkg618.

[51] M Tompa, N Li, T. L. Bailey, G. M. Church, B De 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 Regnier, N Simonis, S Sinha, G Thijs, J van Helden, M Vandenbogaert, Z Weng, C Workman, C Ye, and Z Zhu, "Assessing computational tools for the discovery of transcription factor binding sites", *Nat.Biotechnol.*, vol. 23, no. 1, pp. 137–144, 2005, ISSN: 1087-0156. DOI: 10.1038/nbt1053.

[52] G. Pavesi, P. Mereghetti, F. Zambelli, M. Stefani, G. Mauri, and G. Pesole, "MoD Tools: regulatory motif discovery in nucleotide sequences from co-regulated or homologous genes", *Nucleic Acids Research*, vol. 34, no. Web Server, W566–W570, Jul. 2006, ISSN: 0305-1048. DOI: 10.1093/nar/gkl285. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkl285.

[53] R. A. Cartwright, "DNA assembly with gaps (Dawg): simulating sequence evolution", *Bioinformatics*, vol. 21, no. Suppl 3, pp. iii31–iii38, Nov. 2005, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti1200. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bti1200.

[54] E. Blanco, D. Farré, M. M. Albà, X. Messeguer, and R. Guigó, "ABS: a database of Annotated regulatory Binding Sites from orthologous promoters.", *Nucleic acids research*, vol. 34, no. Database issue, pp. 63–7, Jan. 2006, ISSN: 1362-4962. DOI: 10.1093/nar/gkj116. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16381947http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1347478.

[55] M. Blanchette and M. Tompa, "FootPrinter: A program designed for phylogenetic footprinting.", *Nucleic acids research*, vol. 31, no. 13, pp. 3840–2, Jul. 2003, ISSN: 1362-4962. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/12824433http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC169012.

[56] A. Siepel and D. Haussler, "Combining Phylogenetic and Hidden Markov Models in Biosequence Analysis", *Journal of Computational Biology*, vol. 11, no. 2-3, pp. 413–428, Mar. 2004, ISSN: 1066-5277. DOI: 10.1089/1066527041410472. [Online]. Available: http://www.liebertpub.com/doi/10.1089/1066527041410472.

[57] L. Ettwiller, B. Paten, M. Ramialison, E. Birney, and J. Wittbrodt, "Trawler: de novo regulatory motif discovery pipeline for chromatin immunoprecipitation", *Nature Methods*, vol. 4, no. 7, pp. 563–565, Jul. 2007, ISSN: 1548-7091. DOI: 10.1038/nmeth1061. [Online]. Available: http://www.nature.com/articles/nmeth1061.

[58] J. D. Hughes, P. W. Estep, S. Tavazoie, and G. M. Church, "Computational identification of Cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae", *Journal of Molecular Biology*, vol. 296, no. 5, pp. 1205–1214, Mar. 2000, ISSN: 0022-2836. DOI: 10.1006/JMBI.2000.3519. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022283600935198.

[59] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou, "MotifCut: regulatory motifs finding with maximum density subgraphs", *Bioinformatics*, vol. 22, no. 14, e150–e157, Jul. 2006, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btl243. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btl243.

[60] X. S. Liu, D. L. Brutlag, and J. S. Liu, "An algorithm for finding protein–DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments", *Nature Biotechnology*, vol. 20, no. 8, pp. 835–839, Aug. 2002, ISSN: 1087-0156. DOI: 10.1038/nbt717. [Online]. Available: http://www.nature.com/articles/nbt717.

[61] 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. Pacific Symposium on Biocomputing*, pp. 127–38, 2001, ISSN: 2335-6928. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/11262934.

[62] G. Z. Hertz, G. W. Hartzell, and G. D. Stormo, "Identification of consensus patterns in unaligned DNA sequences known to be functionally related", *Bioinformatics*, vol. 6, no. 2, pp. 81–92, Apr. 1990, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/6.2.81. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/6.2.81.

[63] F. P. Roth, J. D. Hughes, P. W. Estep, and G. M. Church, "Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation", *Nature Biotechnology*, vol. 16, no. 10, pp. 939–945, Oct. 1998, ISSN: 1087-0156. DOI: 10.1038/nbt1098-939. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/9788350http://www.nature.com/articles/nbt1098-939.

[64] C. Jia, M. B. Carson, Y. Wang, Y. Lin, and H. Lu, "A New Exhaustive Method and Strategy for Finding Motifs in ChIP-Enriched Regions", *PLoS ONE*, vol. 9, no. 1, Y. Xu, Ed., e86044, Jan. 2014, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0086044. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24475069http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3901781https://dx.plos.org/10.1371/journal.pone.0086044.

[65] A. F. Neuwald, J. S. Liu, and C. E. Lawrence, "Gibbs motif sampling: Detection of bacterial outer membrane protein repeats", *Protein Science*, vol. 4, no. 8, pp. 1618–1632, Aug. 1995, ISSN: 09618368. DOI: 10.1002/pro.5560040820. [Online]. Available: http://doi.wiley.com/10.1002/pro.5560040820.

[66] P. Collas and J. A. Dahl, "Chop it, ChIP it, check it: the current status of chromatin immunoprecipitation.", *Frontiers in bioscience : a journal and virtual library*, vol. 13, pp. 929–43, Jan. 2008, ISSN: 1093-9946. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17981601.

[67] S. Pillai and S. P. Chellappan, "ChIP on Chip Assays: Genome-Wide Analysis of Transcription Factor Binding and Histone Modifications", in *Methods in molecular biology (Clifton, N.J.)* Vol. 523, 2009, pp. 341–366. DOI: 10.1007/978-1-59745-190-1{_}23. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19381927http://link.springer.com/10.1007/978-1-59745-190-1_23.

[68] E. R. Mardis, "ChIP-seq: welcome to the new frontier", *Nature Methods*, vol. 4, no. 8, pp. 613–614, Aug. 2007, ISSN: 1548-7091. DOI: 10.1038/nmeth0807-613. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17664943http://www.nature.com/articles/nmeth0807-613.

[69] P. Machanick and T. L. Bailey, "MEME-ChIP: motif analysis of large DNA datasets", *Bioinformatics*, vol. 27, no. 12, pp. 1696–1697, Jun. 2011, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btr189. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr189.

[70] M. Hu, J. Yu, J. M. G. Taylor, A. M. Chinnaiyan, and Z. S. Qin, "On the detection and refinement of transcription factor binding sites using ChIP-Seq data", *Nucleic Acids Research*, vol. 38, no. 7, pp. 2154–2167, Apr. 2010, ISSN: 0305-1048. DOI: 10.1093/nar/gkp1180. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/20056654http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2853110https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkp1180.

[71] J. E. Reid and L. Wernisch, "STEME: efficient EM to find motifs in large data sets", *Nucleic Acids Research*, vol. 39, no. 18, e126–e126, Oct. 2011, ISSN: 1362-4962. DOI: 10.1093/nar/gkr574. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21785132http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3185442https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkr574.

[72] ——, "STEME: A Robust, Accurate Motif Finder for Large Data Sets", *PLoS ONE*, vol. 9, no. 3, T. J. Hubbard, Ed., e90735, Mar. 2014, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0090735. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24625410http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3953122https://dx.plos.org/10.1371/journal.pone.0090735.

[73] L. T. Dang, M. Tondl, M. H. H. Chiu, J. Revote, B. Paten, V. Tano, A. Tokolyi, F. Besse, G. Quaife-Ryan, H. Cumming, M. J. Drvodelic, M. P. Eichenlaub, J. C. Hallab, J. S. Stolper, F. J. Rossello, M. A. Bogoyevitch, D. A. Jans, H. T. Nim, E. R. Porrello, J. E. Hudson, and M. Ramialison, "TrawlerWeb: an online de novo motif discovery tool for next-generation sequencing datasets", *BMC Genomics*, vol. 19, no. 1, p. 238, Dec. 2018, ISSN: 1471-2164. DOI: 10.1186/s12864-018-4630-0. [Online]. Available: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-018-4630-0.

[74] C. Linhart, Y. Halperin, and R. Shamir, "Transcription factor and microRNA motif discovery: The Amadeus platform and a compendium of metazoan target sets", *Genome Research*, vol. 18, no. 7, pp. 1180–1189, Jul. 2008, ISSN: 1088-9051. DOI: 10.1101/GR.076117.108. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18411406http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2493407.

[75] Y. Halperin, C. Linhart, I. Ulitsky, and R. Shamir, "Allegro: Analyzing expression and sequence in concert to discover regulatory programs", *Nucleic Acids Research*, vol. 37, no. 5, pp. 1566–1579, Apr. 2009, ISSN: 1362-4962. DOI: 10.1093/nar/gkn1064. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkn1064.

[76] T. L. Bailey, "DREME: motif discovery in transcription factor ChIP-seq data", *Bioinformatics*, vol. 27, no. 12, pp. 1653–1659, Jun. 2011, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btr261. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21543442http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3106199https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr261.

[77] A. A. Sharov and M. S. Ko, "Exhaustive Search for Over-represented DNA Sequence Motifs with CisFinder", *DNA Research*, vol. 16, no. 5, pp. 261–273, Oct. 2009, ISSN: 1340-2838. DOI: 10.1093/dnares/dsp014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19740934http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2762409https://academic.oup.com/dnaresearch/article-lookup/doi/10.1093/dnares/dsp014.

[78] S. Georgiev, A. P. Boyle, K. Jayasurya, X. Ding, S. Mukherjee, and U. Ohler, "Evidence-ranked motif identification", *Genome Biology*, vol. 11, no. 2, R19, 2010, ISSN: 1465-6906. DOI: 10.1186/gb-2010-11-2-r19. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/20156354http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2872879http://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-2-r19.

[79] M. Thomas-Chollier, C. Herrmann, M. Defrance, O. Sand, D. Thieffry, and J. van Helden, "RSAT peak-motifs: motif analysis in full-size ChIP-seq datasets.", *Nucleic acids research*, vol. 40, no. 4, e31, Feb. 2012, ISSN: 1362-4962. DOI: 10.1093/nar/gkr1104. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22156162http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3287167.

[80] Q. Yu, D. Wei, and H. Huo, "SamSelect: a sample sequence selection algorithm for quorum planted motif search on large DNA datasets", *BMC Bioinformatics*, vol. 19, no. 1, p. 228, Dec. 2018, ISSN: 1471-2105. DOI: 10.1186/s12859-018-2242-y. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/29914360http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC6006848https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2242-y.

[81] N. K. Lee, F. L. Azizan, Y. S. Wong, and N. Omar, "DeepFinder: An integration of feature-based and deep learning approach for DNA motif discovery", *Biotechnology & Biotechnological Equipment*, vol. 32, no. 3, pp. 759–768, May 2018, ISSN: 1310-2818. DOI: 10.1080/13102818.2018.1438209.

[82] C. T. Harbison, D. B. Gordon, T. I. Lee, N. J. Rinaldi, K. D. Macisaac, T. W. Danford, N. M. Hannett, J.-B. Tagne, D. B. Reynolds, J. Yoo, E. G. Jennings, J. Zeitlinger, D. K. Pokholok, M. Kellis, P. A. Rolfe, K. T. Takusagawa, E. S. Lander, D. K. Gifford, E. Fraenkel, and R. A. Young, "Transcriptional regulatory code of a eukaryotic genome", *Nature*, vol. 431, no. 7004, pp. 99–104, Sep. 2004, ISSN: 0028-0836. DOI: 10.1038/nature02800. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/15343339http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3006441http://www.nature.com/articles/nature02800.

[83] J. S. Liu, A. F. Neuwald, and C. E. Lawrence, "Bayesian Models for Multiple Local Sequence Alignment and Gibbs Sampling Strategies", *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1156–1170, Dec. 1995, ISSN: 0162-1459. DOI: 10.1080/01621459.1995.10476622. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.1995.10476622.

[84] G. Z. Hertz and G. D. Stormo, "Identifying DNA and protein patterns with statistically significant alignments of multiple sequences.", *Bioinformatics (Oxford, England)*, vol. 15, no. 7-8, pp. 563–77, 1999, ISSN: 1367-4803. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/10487864.

[85] E. M. Conlon, X. S. Liu, J. D. Lieb, and J. S. Liu, "Integrating regulatory motif discovery and genome-wide expression analysis.", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 6, pp. 3339–44, Mar. 2003, ISSN: 0027-8424. DOI: 10.1073/pnas.0630591100. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/12626739http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC152294.

[86] M. C. Frith, U. Hansen, J. L. Spouge, and Z. Weng, "Finding functional sequence elements by multiple local alignment", *Nucleic Acids Research*, vol. 32, no. 1, pp. 189–200, Jan. 2004, ISSN: 1362-4962. DOI: 10.1093/nar/gkh169. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkh169.

[87] G Thijs, M Lescot, K Marchal, S Rombauts, B De Moor, P Rouzé, and Y Moreau, "A higher-order background model improves the detection of promoter regulatory elements by Gibbs sampling.", *Bioinformatics (Oxford, England)*, vol. 17, no. 12, pp. 1113–22, Dec. 2001, ISSN: 1367-4803. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/11751219.

[88] A. V. Favorov, M. S. Gelfand, A. V. Gerasimova, D. A. Ravcheev, A. A. Mironov, and V. J. Makeev, "A Gibbs sampler for identification of symmetrically structured, spaced DNA motifs with improved estimation of the signal length", *Bioinformatics*, vol. 21, no. 10, pp. 2240–2245, May 2005, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti336. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bti336.

[89] S. T. Jensen and J. S. Liu, "BioOptimizer: a Bayesian scoring function approach to motif discovery", *Bioinformatics*, vol. 20, no. 10, pp. 1557–1564, Jul. 2004, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bth127. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/14962923https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bth127.

[90] M. C. Frith, N. F. W. Saunders, B. Kobe, and T. L. Bailey, "Discovering Sequence Motifs with Arbitrary Insertions and Deletions", *PLoS Computational Biology*, vol. 4, no. 5, G. Stormo, Ed., e1000071, May 2008, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000071. [Online]. Available: http://dx.plos.org/10.1371/journal.pcbi.1000071.

[91] C. E. Grant, T. L. Bailey, and W. S. Noble, "FIMO: scanning for occurrences of a given motif", *Bioinformatics*, vol. 27, no. 7, pp. 1017–1018, Apr. 2011, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btr064. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21330290http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3065696https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr064.

[92] T. L. Bailey and W. S. Noble, "Searching for statistically significant regulatory modules.", *Bioinformatics (Oxford, England)*, vol. 19 Suppl 2, pp. 16–25, Oct. 2003, ISSN: 1367-4811. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/14534166.

[93] S. Gupta, J. A. Stamatoyannopoulos, T. L. Bailey, and W. S. Noble, "Quantifying similarity between motifs.", *Genome biology*, vol. 8, no. 2, R24, 2007, ISSN: 1474-760X. DOI: 10.1186/gb-2007-8-2-r24. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17324271http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1852410.

[94] T. Whitington, M. C. Frith, J. Johnson, and T. L. Bailey, "Inferring transcription factor complexes from ChIP-seq data", *Nucleic Acids Research*, vol. 39, no. 15, e98–e98, Aug. 2011, ISSN: 1362-4962. DOI: 10.1093/nar/gkr341. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkr341.

[95] F. A. Buske, M. Bodén, D. C. Bauer, and T. L. Bailey, "Assigning roles to DNA regulatory motifs using comparative genomics", *Bioinformatics*, vol. 26, no. 7, pp. 860–866, Apr. 2010, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btq049. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btq049.

[96] T. L. Bailey and P. Machanick, "Inferring direct DNA binding from ChIP-seq", *Nucleic Acids Research*, vol. 40, no. 17, e128–e128, Sep. 2012, ISSN: 1362-4962. DOI: 10.1093/nar/gks433. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22610855http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3458523https://academic.oup.com/nar/article/40/17/e128/2411117.

[97] G. E. Crooks, G. Hon, J.-M. Chandonia, and S. E. Brenner, "WebLogo: A Sequence Logo Generator", *Genome Reseach*, no. 14, pp. 1188–1190, 2004. DOI: 10.1101/gr.849004. [Online]. Available: www.genome.org.

[98] C. T. Workman, Y. Yin, D. L. Corcoran, T. Ideker, G. D. Stormo, and P. V. Benos, "enoLOGOS: a versatile web tool for energy normalized sequence logos", *Nucleic Acids Research*, vol. 33, no. Web Server, W389–W392, Jul. 2005, ISSN: 0305-1048. DOI: 10.1093/nar/gki439. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/15980495http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1160200https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gki439.

[99] S. Mahony and P. V. Benos, "STAMP: a web tool for exploring DNA-binding motif similarities.", *Nucleic acids research*, vol. 35, no. Web Server issue, pp. 253–8, Jul. 2007, ISSN: 1362-4962. DOI: 10.1093/nar/gkm272. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17478497http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1933206.

[100] A. M. Szalkowski and C. D. Schmid, "Rapid innovation in ChIP-seq peak-calling algorithms is outdistancing benchmarking efforts", *Briefings in Bioinformatics*, vol. 12, no. 6, pp. 626–633, Nov. 2011, ISSN: 1467-5463. DOI: 10.1093/bib/bbq068. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21059603https://academic.oup.com/bib/article-lookup/doi/10.1093/bib/bbq068.

[101] A Lihu and Holban, "A review of ensemble methods for de novo motif discovery in ChIP-Seq data", *Briefings in Bioinformatics*, vol. 16, no. 6, pp. 964–973, 2015.

[102] T. Okumura, H. Makiguchi, Y. Makita, R. Yamashita, and K. Nakai, "Melina II: a web tool for comparisons among several predictive algorithms to find potential motifs from promoter regions.", *Nucleic acids research*, vol. 35, no. Web Server issue, pp. 227–31, Jul. 2007, ISSN: 1362-4962. DOI: 10.1093/nar/gkm362. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17537821http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1933176.

[103] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, J. C. Wootton, and others, "Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment", *SCIENCE-NEW YORK THEN WASHINGTON-*, vol. 262, p. 208, 1993.

[104] R Cavin Perier, T. Junier, and P. Bucher, "The Eukaryotic Promoter Database EPD", *Nucleic Acids Research*, vol. 26, no. 1, pp. 353–357, Jan. 1998, ISSN: 13624962. DOI: 10.1093/nar/26.1.353. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/26.1.353.

[105] I. Yevshin, R. Sharipov, T. Valeev, A. Kel, and F. Kolpakov, "GTRD: a database of transcription factor binding sites identified by ChIP-seq experiments.", *Nucleic acids research*, vol. 45, no. D1, pp. D61–D67, 2017, ISSN: 1362-4962. DOI: 10.1093/nar/gkw951. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/27924024http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5210645.

[106] I. Yevshin, R. Sharipov, S. Kolmykov, Y. Kondrakhin, and F. Kolpakov, "GTRD: a database on gene transcription regulation—2019 update", *Nucleic Acids Research*, vol. 47, no. D1, pp. D100–D105, Jan. 2019, ISSN: 0305-1048. DOI: 10.1093/nar/gky1128. [Online]. Available: https://academic.oup.com/nar/article/47/D1/D100/5184717.

[107] A. Sandelin, W. Alkema, P. Engström, W. W. Wasserman, and B. Lenhard, "JASPAR: an open-access database for eukaryotic transcription factor binding profiles.", *Nucleic acids research*, vol. 32, no. Database issue, pp. 91–4, Jan. 2004, ISSN: 1362-4962. DOI: 10.1093/nar/gkh012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/14681366http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC308747.

[108] R. Munch, K. Hiller, H. Barg, D. Heldt, S. Linz, E. Wingender, and D. Jahn, "PRODORIC: prokaryotic database of gene regulation", *Nucleic Acids Research*, vol. 31, no. 1, pp. 266–269, Jan. 2003.

[109] S. Gama-Castro, H. Salgado, A. Santos-Zavaleta, D. Ledezma-Tejeida, L. Muñiz-Rascado, J. S. García-Sotelo, K. Alquicira-Hernández, I. Martínez-Flores, L. Pannier, J. A. Castro-Mondragón, A. Medina-Rivera, H. Solano-Lira, C. Bonavides-Martínez, E. Pérez-Rueda, S. Alquicira-Hernández, L. Porrón-Sotelo, A. López-Fuentes, A. Hernández-Koutoucheva, V. D. Moral-Chávez, F. Rinaldi, and J. Collado-Vides, "RegulonDB version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond", *Nucleic Acids Research*, vol. 44, no. D1, pp. D133–D143,

Jan. 2016, ISSN: 0305-1048. DOI: 10.1093/nar/gkv1156. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/26527724http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4702833https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkv1156.

[110] J. Cherry, C. Adler, C. Ball, S. A. Chervitz, S. S. Dwight, E. T. Hester, Y. Jia, G. Juvik, T. Roe, M. Schroeder, S. Weng, and D. Botstein, "SGD: Saccharomyces Genome Database", *Nucleic Acids Research*, vol. 26, no. 1, pp. 73–79, Jan. 1998, ISSN: 13624962. DOI: 10.1093/nar/26.1.73. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/26.1.73.

[111] E Wingender, P. Dietze, H. Karas, and R. Knüppel, "TRANSFAC: a database on transcription factors and their DNA binding sites", *Nucleic Acids Research*, vol. 24, no. 1, pp. 238–241, Jan. 1996, ISSN: 13624962. DOI: 10.1093/nar/24.1.238. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/24.1.238.

[112] M. C. Teixeira, P. T. Monteiro, M. Palma, C. Costa, C. P. Godinho, P. Pais, M. Cavalheiro, M. Antunes, A. Lemos, T. Pedreira, and I. Sá-Correia, "YEASTRACT: an upgraded database for the analysis of transcription regulatory networks in Saccharomyces cerevisiae.", *Nucleic acids research*, vol. 46, no. D1, pp. D348–D353, Jan. 2018, ISSN: 1362-4962. DOI: 10.1093/nar/gkx842. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/29036684http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5753369.

[113] D. E. Newburger and M. L. Bulyk, "UniPROBE: an online database of protein binding microarray data on protein-DNA interactions.", *Nucleic acids research*, vol. 37, no. Database issue, pp. 77–82, Jan. 2009, ISSN: 1362-4962. DOI: 10.1093/nar/gkn660. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18842628http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC2686578.

[114] J. M. Carlson, A. Chakravarty, C. E. DeZiel, and R. H. Gross, "SCOPE: a web server for practical de novo motif discovery", *Nucleic Acids Research*, vol. 35, no. Web Server, W259–W264, May 2007, ISSN: 0305-1048. DOI: 10.1093/nar/gkm310. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkm310.

[115] E. Wijaya, S.-M. Yiu, N. T. Son, R. Kanagasabai, and W.-K. Sung, "MotifVoter: a novel ensemble method for fine-grained integration of generic motif finders", *Bioinformatics*, vol. 24, no. 20, pp. 2288–2295, Oct. 2008, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btn420. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18697768https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btn420.

[116] S. J. van Heeringen and G. J. C. Veenstra, "GimmeMotifs: a de novo motif prediction pipeline for ChIP-sequencing experiments.", *Bioinformatics (Oxford, England)*, vol. 27, no. 2, pp. 270–1, Jan. 2011, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btq636. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21081511http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3018809.

[117] N. Bruse and S. J. v. Heeringen, "GimmeMotifs: an analysis framework for transcription factor motif analysis", *bioRxiv*, p. 474 403, Nov. 2018. DOI: 10.1101/474403. [Online]. Available: https://www.biorxiv.org/content/early/2018/11/20/474403?rss=1.

[118] J. Hu, Y. D. Yang, and D. Kihara, "EMD: an ensemble algorithm for discovering regulatory motifs in DNA sequences.", *BMC bioinformatics*, vol. 7, p. 342, Jul. 2006, ISSN: 1471-2105. DOI: 10.1186/1471-2105-7-342. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/16839417http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1539026.

[119] K. A. Romer, G.-R. Kayombya, and E. Fraenkel, "WebMOTIFS: automated discovery, filtering and scoring of DNA sequence motifs using multiple programs and Bayesian approaches", *Nucleic Acids Research*, vol. 35, no. Web Server, W217–W220, May 2007, ISSN: 0305-1048. DOI: 10.1093/nar/gkm376. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17584794http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC1933171https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkm376.

[120] L. Kuttippurathu, M. Hsing, Y. Liu, B. Schmidt, D. L. Maskell, K. Lee, A. He, W. T. Pu, and S. W. Kong, "CompleteMOTIFs: DNA motif discovery platform for transcription factor binding experiments", *Bioinformatics*, vol. 27, no. 5, pp. 715–717, Mar. 2011, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btq707. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/21183585http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3105477https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btq707.

[121] E. Dassi and A. Quattrone, "DynaMIT: the dynamic motif integration toolkit", *Nucleic Acids Research*, vol. 44, no. 1, e2–e2, Jan. 2016, ISSN: 0305-1048. DOI: 10.1093/nar/gkv807. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/26253738http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC4705680https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkv807.

[122] P. Huggins, S. Zhong, I. Shiff, R. Beckerman, O. Laptenko, C. Prives, M. H. Schulz, I. Simon, and Z. Bar-Joseph, "DECOD: fast and accurate discriminative DNA motif finding", *Bioinformatics*, vol. 27, no. 17, pp. 2361–2367, Sep. 2011, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btr412. [Online]. Available: http://www.ncbi.nlm.

nih.gov/pubmed/21752801http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3157928https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr412.

[123] S. Luehr, H. Hartmann, and J. Soding, "The XXmotif web server for eXhaustive, weight matriX-based motif discovery in nucleotide sequences", *Nucleic Acids Research*, vol. 40, no. W1, W104–W109, Jul. 2012, ISSN: 0305-1048. DOI: 10.1093/nar/gks602. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22693218http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3394272https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gks602.

[124] M. J. Mason, K. Plath, and Q. Zhou, "Identification of Context-Dependent Motifs by Contrasting ChIP Binding Data", *Bioinformatics*, vol. 26, no. 22, pp. 2826–2832, Nov. 2010, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btq546. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btq546.

[125] G. Sandve, O. Abul, V. Walseng, and F. Drabløs, "Improved benchmarks for computational motif discovery", *BMC Bioinformatics*, vol. 8, no. 1, p. 193, Jun. 2007, ISSN: 14712105. DOI: 10.1186/1471-2105-8-193. [Online]. Available: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-8-193.

[126] M. Burset and R. Guigó, "Evaluation of Gene Structure Prediction Programs", *Genomics*, vol. 34, no. 3, pp. 353–367, Jun. 1996, ISSN: 08887543. DOI: 10.1006/geno.1996.0298.

[127] H. Al-Shaikhli and E. De Doncker, "SMF: Approximate Algorithm for the Planted (l, d) Motif Finding Problem in DNA Sequences", in *International Conference on Bioinformatics & Computational Biology*, 2018, pp. 123–129, ISBN: 1601324715. [Online]. Available: https://csce.ucmss.com/cr/books/2018/LFS/CSREA2018/BIC4274.pdf.

[128] H. C. Leung and F. Y. Chin, "Generalized Planted (l,d)-Motif Problem with Negative Set", in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3692 LNBI, 2005, pp. 264–275, ISBN: 3540290087. DOI: 10.1007/11557067{_}22. [Online]. Available: http://link.springer.com/10.1007/11557067_22.

[129] N. Li and M. Tompa, "Analysis of computational approaches for motif discovery.", *Algorithms for molecular biology : AMB*, vol. 1, p. 8, 2006, ISSN: 1748-7188. DOI: 10.1186/1748-7188-1-8.

[130] H. Al-Shaikhli and E. de Doncker, "qSMF: an Approximate Algorithm for Quorum Planted Motif Search on ChIP-Seq Data", in *2019 IEEE International Conference on Electro/Information Technology*, To Appear in IEEE Xplore, 2019.

[131] M. Blanchette and M. Tompa, "Discovery of Regulatory Elements by a Computational Method for Phylogenetic Footprinting", *Genome Research*, vol. 12, no. 5, pp. 739–748, May 2002, ISSN: 10889051. DOI: 10.1101/gr.6902.

[132] M. M. Abbas, Q. M. Malluhi, and P. Balakrishnan, "Scalable multi-core implementation for motif finding problem", in *Proceedings - IEEE 13th International Symposium on Parallel and Distributed Computing, ISPDC 2014*, IEEE, Jun. 2014, pp. 178–183, ISBN: 9780769552651. DOI: 10.1109/ISPDC.2014.27.

[133] P. Kheradpour and M. Kellis, "Systematic discovery and characterization of regulatory motifs in ENCODE TF binding experiments", *Nucleic Acids Research*, vol. 42, no. 5, pp. 2976–2987, Mar. 2014, ISSN: 0305-1048. DOI: 10.1093/nar/gkt1249. [Online]. Available: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkt1249.

[134] *MEME - MEME Suite*. [Online]. Available: http://web.mit.edu/meme_v4.11.4/share/doc/meme.html.

[135] H. Touzet and J.-S. Varré, "Efficient and accurate P-value computation for Position Weight Matrices", *Algorithms for Molecular Biology*, vol. 2, no. 1, p. 15, Dec. 2007, ISSN: 1748-7188. DOI: 10.1186/1748-7188-2-15. [Online]. Available: https://almob.biomedcentral.com/articles/10.1186/1748-7188-2-15.

[136] F. Wolfertstetter, K. Frech, G. Herrmann, and T. Werner, "Identification of functional elements in unaligned nucleic acid sequences by a novel tuple search algorithm", *Bioinformatics*, vol. 12, no. 1, pp. 71–80, Feb. 1996, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/12.1.71. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/12.1.71.

[137] I. Jonassen, "Efficient discovery of conserved patterns using a pattern graph", *Bioinformatics*, vol. 13, no. 5, pp. 509–522, Oct. 1997, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/13.5.509. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/13.5.509.

[138] I. Rigoutsos and A. Floratos, "Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm", *Bioinformatics*, vol. 14, no. 1, pp. 55–67, Feb. 1998, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/14.1.55. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/14.1.55.

[139] C. T. Workman and G. D. Stormo, "ANN-Spec: a method for discovering transcription factor binding sites with improved specificity.", *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pp. 467–78, 2000, ISSN: 2335-6928. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/10902194.

[140] U Keich and P. A. Pevzner, "Finding motifs in the twilight zone.", *Bioinformatics (Oxford, England)*, vol. 18, no. 10, pp. 1374–81, Oct. 2002, ISSN: 1367-4803. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/12376382.

[141] E. Eskin and P. A. Pevzner, "Finding composite regulatory patterns in DNA sequences", *Bioinformatics*, vol. 18, no. Suppl 1, S354–S363, Jul. 2002, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/18.suppl{_}1.S354. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/18.suppl_1.S354.

[142] S. Sinha, E. van Nimwegen, and E. D. Siggia, "A probabilistic method to detect regulatory modules", *Bioinformatics*, vol. 19, no. Suppl 1, pp. i292–i301, Jul. 2003, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btg1040. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btg1040.

[143] S. LIANG, M. P. SAMANTA, and B. A. BIEGEL, "cWINNOWER ALGORITHM FOR FINDING FUZZY DNA MOTIFS", *Journal of Bioinformatics and Computational Biology*, vol. 02, no. 01, pp. 47–60, Mar. 2004, ISSN: 0219-7200. DOI: 10.1142/S0219720004000466. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0219720004000466.

[144] P. A. Evans and A. D. Smith, "Toward Optimal Motif Enumeration", in, Springer, Berlin, Heidelberg, 2003, pp. 47–58. DOI: 10.1007/978-3-540-45078-8{_}5. [Online]. Available: http://link.springer.com/10.1007/978-3-540-45078-8_5.

[145] A. Price, S. Ramabhadran, and P. A. Pevzner, "Finding subtle motifs by branching from sample strings", *Bioinformatics*, vol. 19, no. Suppl 2, pp. ii149–ii155, Sep. 2003, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btg1072. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btg1072.

[146] S. Sinha, M. Blanchette, and M. Tompa, "PhyME: A probabilistic algorithm for finding motifs in sets of orthologous sequences", *BMC Bioinformatics*, vol. 5, no. 1, p. 170, Oct. 2004, ISSN: 14712105. DOI: 10.1186/1471-2105-5-170. [Online]. Available: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-5-170.

[147] X. Yang and J. C. Rajapakse, "Graphical approach to weak motif recognition.", *Genome informatics. International Conference on Genome Informatics*, vol. 15, no. 2, pp. 52–62, 2004, ISSN: 0919-9454. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/15706491.

[148] F. Liu, J. Tsai, R. Chen, S. Chen, and S. Shih, "FMGA: finding motifs by genetic algorithm", in *Proceedings. Fourth IEEE Symposium on Bioinformatics and Bioengineering*, IEEE, pp. 459–466, ISBN: 0-7695-2173-8. DOI: 10.1109/BIBE.2004.1317378. [Online]. Available: http://ieeexplore.ieee.org/document/1317378/.

[149] R. Siddharthan, E. D. Siggia, and E. van Nimwegen, "PhyloGibbs: A Gibbs Sampling Motif Finder That Incorporates Phylogeny", *PLoS Computational Biology*, vol. 1, no. 7, e67, 2005, ISSN: 1553-734X. DOI: 10.1371/journal.pcbi.0010067. [Online]. Available: http://dx.plos.org/10.1371/journal.pcbi.0010067.

[150] S. Rajasekaran, S. Balla, and C.-H. Huang, "Exact algorithms for planted motif problems.", *Journal of computational biology : a journal of computational molecular cell biology*, vol. 12, no. 8, pp. 1117–1128, 2005, ISSN: 1066-5277.

[151] N. Pisanti, A. M. Carvalho, L. Marsan, and M.-F. Sagot, "RISOTTO: Fast Extraction of Motifs with Mismatches", in, Springer, Berlin, Heidelberg, 2006, pp. 757–768. DOI: 10.1007/11682462{_}69. [Online]. Available: http://link.springer.com/10.1007/11682462_69.

[152] J. Davila, S. Balla, and S. Rajasekaran, "Space and Time Efficient Algorithms for Planted Motif Search", in, Springer, Berlin, Heidelberg, 2006, pp. 822–829. DOI: 10.1007/11758525{_}110. [Online]. Available: http://link.springer.com/10.1007/11758525_110.

[153] S. Tareeq, S Saha, T Islam, R. Q. I. Technology, and U. 2007, "ANT: A Novel Heuristic Algorithm for Finding Motif", *Information Technology Journal*, vol. 6, no. 2, pp. 189–195, 2007. [Online]. Available: http://docsdrive.com/pdfs/ansinet/itj/2007/189-195.pdf.

[154] J. Davila, S. Balla, and S. Rajasekaran, "Fast and Practical Algorithms for Planted (l, d) Motif Search", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 4, pp. 544–552, Oct. 2007, ISSN: 1545-5963. DOI: 10.1109/TCBB.2007.70241.

[155] J. Davila, S. Balla, and S. Rajasekaran, "Pampa: An Improved Branch and Bound Algorithm for Planted (l, d) Motif Search", Tech. Rep. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.93.6500&rep=rep1&type=pdf.

[156] J. A. Young, J. R. Johnson, C. Benner, S. F. Yan, K. Chen, K. G. Le Roch, Y. Zhou, and E. A. Winzeler, "In silico discovery of transcription regulatory elements in Plasmodium falciparum", *BMC Genomics*, vol. 9, no. 1, p. 70, Feb. 2008, ISSN: 1471-2164. DOI: 10.1186/1471-2164-9-70. [Online]. Available: http://bmcgenomics.biomedcentral.com/articles/10.1186/1471-2164-9-70.

[157] D. Sharma and S. Rajasekaran, "A Simple Algorithm for (l, d) Motif Search", in *2009 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, IEEE, Mar. 2009, pp. 148–154, ISBN: 978-1-4244-2756-7. DOI: 10.1109/CIBCB.2009.4925721. [Online]. Available: http://ieeexplore.ieee.org/document/4925721/.

[158] H. Q. Sun, M. Y. H. Low, W. J. Hsu, and J. C. Rajapakse, "RecMotif: a novel fast algorithm for weak motif discovery", *BMC Bioinformatics*, vol. 11, no. S11, S8, Dec. 2010, ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-S11-S8. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-S11-S8.

[159] He Quan Sun, M. Y. Hean Low, Wen Jing Hsu, and J. C. Rajapakse, "ListMotif: A time and memory efficient algorithm for weak motif discovery", in *2010 IEEE International Conference on Intelligent Systems and Knowledge Engineering*, IEEE, Nov. 2010, pp. 254–260, ISBN: 978-1-4244-6791-4. DOI: 10.1109/ISKE.2010.5680875. [Online]. Available: http://ieeexplore.ieee.org/document/5680875/.

[160] P. P. Kuksa and V. Pavlovic, "Efficient motif finding algorithms for large-alphabet inputs", *BMC Bioinformatics*, vol. 11, no. S8, S1, Oct. 2010, ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-S8-S1. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-S8-S1.

[161] S. Rajasekaran and H. Dinh, "A speedup technique for (l, d)-motif finding algorithms", *BMC Research Notes*, vol. 4, no. 1, p. 54, Dec. 2011, ISSN: 1756-0500. DOI: 10.1186/1756-0500-4-54. [Online]. Available: https://bmcresnotes.biomedcentral.com/articles/10.1186/1756-0500-4-54.

[162] H. Q. Sun, M. Y. H. Low, W. J. Hsu, C. W. Tan, and J. C. Rajapakse, "Tree-structured algorithm for long weak motif discovery", *Bioinformatics*, vol. 27, no. 19, pp. 2641–2647, Oct. 2011, ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btr459. [Online]. Available: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btr459.

[163] H. Dinh, S. Rajasekaran, and V. K. Kundeti, "PMS5: an efficient exact algorithm for the (, d)-motif finding problem", *BMC Bioinformatics*, vol. 12, no. 1, p. 410, Oct. 2011, ISSN: 1471-2105. DOI: 10.1186/1471-2105-12-410. [Online]. Available: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-410.

[164] Z.-Z. Zhi-Zhong Chen and L. Lusheng Wang, "Fast Exact Algorithms for the Closest String and Substring Problems with Application to the Planted (L,d)-Motif Model", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 8, no. 5, pp. 1400–1410, Sep. 2011, ISSN: 1545-5963. DOI: 10.1109/TCBB.2011.21. [Online]. Available: http://ieeexplore.ieee.org/document/5708134/.

[165] S. Bandyopadhyay, S. Sahni, and S. Rajasekaran, "PMS6: A fast algorithm for motif discovery", in *2012 IEEE 2nd International Conference on Computational Advances in Bio and medical Sciences (ICCABS)*, IEEE, Feb. 2012, pp. 1–6, ISBN: 978-1-4673-1321-6. DOI: 10.1109/ICCABS.2012.6182627. [Online]. Available: http://ieeexplore.ieee.org/document/6182627/.

[166] Q. Yu, H. Huo, Y. Zhang, and H. Guo, "PairMotif: A New Pattern-Driven Algorithm for Planted (l, d) DNA Motif Search", *PLoS ONE*, vol. 7, no. 10, S. Gómez, Ed., e48442, Oct. 2012, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0048442. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0048442.

[167] H. Dinh, S. Rajasekaran, and J. Davila, "qPMS7: A Fast Algorithm for Finding (, d)-Motifs in DNA and Protein Sequences", *PLoS ONE*, vol. 7, no. 7, V. Brusic, Ed., e41425, Jul. 2012, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0041425. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0041425.

[168] H. Hartmann, E. W. Guthöhrlein, M. Siebert, S. Luehr, and J. Söding, "P-value-based regulatory motif discovery using positional weight matrices.", *Genome research*, vol. 23, no. 1, pp. 181–94, Jan. 2013, ISSN: 1549-5469. DOI: 10.1101/gr.139881.112. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22990209http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3530678.

[169] S. Tanaka, "Improved Exact Enumerative Algorithms for the Planted (l, d)-Motif Search Problem", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 2, pp. 361–374, Mar. 2014, ISSN: 1545-5963. DOI: 10.1109/TCBB.2014.2306842. [Online]. Available: http://ieeexplore.ieee.org/document/6744602/.

[170] M. Nicolae and S. Rajasekaran, "Efficient sequential and parallel algorithms for planted motif search", *BMC Bioinformatics*, vol. 15, no. 1, p. 34, Dec. 2014, ISSN: 1471-2105. DOI: 10.1186/1471-2105-15-34. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-15-34.

[171] ——, "qPMS9: An Efficient Algorithm for Quorum Planted Motif Search", *Scientific Reports*, vol. 5, no. 1, p. 7813, Jul. 2015, ISSN: 2045-2322. DOI: 10.1038/srep07813.

[172] P. Xiao, S. Pal, and S. Rajasekaran, "qPMS10: A randomized algorithm for efficiently solving quorum Planted Motif Search problem", in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, IEEE, Dec. 2016, pp. 670–675, ISBN: 978-1-5090-1611-2. DOI: 10.1109/BIBM.2016.7822598.

[173] J. Basha Gutierrez and K. Nakai, "A study on the application of topic models to motif finding algorithms", *BMC Bioinformatics*, vol. 17, no. S19, p. 502, Dec. 2016, ISSN: 1471-2105. DOI: 10.1186/s12859-016-1364-3. [Online]. Available: http://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-016-1364-3.

[174] A. Muttakin and M. R. Huq, "Motif discovery in unaligned DNA sequences using genetic algorithm", in *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, IEEE, Sep. 2017, pp. 725–730, ISBN: 978-1-5386-0869-2. DOI: 10.1109/ICAEE.2017.8255450. [Online]. Available: http://ieeexplore.ieee.org/document/8255450/.

[175] C. Saad, L. Noé, H. Richard, J. Leclerc, M.-P. Buisine, H. Touzet, and M. Figeac, "DiNAMO: highly sensitive DNA motif discovery in high-throughput sequencing data", *BMC Bioinformatics*, vol. 19, no. 1, p. 223, Dec. 2018, ISSN: 1471-2105. DOI: 10.1186/s12859-018-2215-1. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2215-1.

[176] M. A. H. Samee, B. G. Bruneau, and K. S. Pollard, "A De Novo Shape Motif Discovery Algorithm Reveals Preferences of Transcription Factors for DNA Shape Beyond Sequence Motifs", *Cell Systems*, vol. 8, no. 1, pp. 27–42, Jan. 2019, ISSN: 2405-4712. DOI: 10.1016/J.CELS.2018.12.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405471218304757.

A. Algorithms for Motif Discovery Problem

This appendix presents the current algorithms that are dedicated to solve motif finding problem from 1993 until 2019 (27 years). The listed algorithms categories are approximate and exact. There are more algorithms that are not listed. The table only lists the algorithms that have names.

Table 1: List of Current Algorithms for Motif Finding Problem

Year	Algorithm(s)
1993	Gibbs Sampler [103]
1994	MEME [40]
1995	Gibbs Motif Sampler [83]
1996	CoreSearch [136]
1997	Pratt2 [137]
1998	AlignACE [63], TEIRESIAS [138], SPELLER [19]
1999	CONSENSUS [84]
2000	ANN-Spec [139], WINNOWER [20], SP-STAR [20]
2001	Weeder [18], BioProspector [61], MotifSampler [87]
2002	MDscan [60], PROJECTION [16], MULTIPROFILER [140], MITRA [141]
2003	YMF [50], Stubb [142], FootPrinter [55], cWINNOWER [143], CENSUS [144], Pattern Branching [145]
2004	PhyME [146], DPCFG [147], GLAM [86], FMGA [148]
2005	SeSiMCMC [88], PhyloGibbs [149], Voting [10], PMS1 [150], PMS2 [150], PMS3 [150]
2006	MotifCut [59], RISOTTO [151], PMSi [152], PMSP [152]
2007	ANT [153], PMSPrune [154], Pampa [155], Trawler [57]
2008	GEMS [156]
2009	PMS3p [157]
2010	RecMotif [158], ListMotif [159], stemming [160], PMS4 [161], CMF [124]
2011	VINE [6], TreeMotif [162], PMS5 [163], Provable [164], DECOD [122], STEME [71]
2012	PMS6 [165], PairMotif [166], qPMSPrunel [167], qPMS7 [167]
2013	XXmotif [168]
2014	FMotif [64], TravStrR [169], TravStrD [169], PMS8 [170]
2015	APMotif [47], qPMS9 [171]
2016	qPMS10 [172], CTM [173]
2017	MDGA [174]
2018	SMF [127], DiNAMO [175], DeepFinder [81], MFMD [30]
2019	ShapeMF [176], qSMF [130]

B. Planted (l, d) Motif Finding Problem Solvability Analysis

In order to assess the SMF algorithm performance, we tested the algorithm on several challenge instances with motif lengths ranging from 8 to 30. To achieve this goal, we need to determine which problem instances are solvable and can be classified as challenge instances, and which are unsolvable. Buhler and Tompa in [16] gave a probabilistic analysis regarding the calculation of the expected number of motifs in ($t = 20$) random DNA sequences each with the length of ($n = 600$) nucleotides. They performed this analysis to investigate the reasons behind the failure of their algorithm PROJECTION in solving instances such as (9, 2), (11, 3), (13, 4), (15, 5), and (17, 6). They found that the expected number $E_t(l, d)$ of these instances is high compared to $E_t(l + 1, d)$, where $E_t(l, d)$ is the expected number of motifs of length l occurring with up to d variations at least once in each of the t sequences,

$$E_t(l, d) = 4^l \left(1 - (1 - p_d)^{n-l+1}\right)^t \tag{1}$$

where

$$p_d = \sum_{i=0}^{d} \binom{l}{i} \left(\frac{3}{4}\right)^i \left(\frac{1}{4}\right)^{l-i} \tag{2}$$

denotes the probability that a fixed substring of length l occurs with up to d variations at a given position of a random sequence [16]. They included a table of $E_t(l, d)$ and $E_t(l + 1, d)$ for motif lengths $9 \leq l \leq 17$. The authors of [132] used the probabilistic analysis of both [16] and [128] to generate a table for motifs of lengths $13 \leq l \leq 21$. They defined a problem instance (l, d) as solvable if the $E_t(l, d)$ value is small. If $E_t(l, d)$ is large ($>> 1$), then the problem instance is called unsolvable. The last solvable instance can be considered as a challenge instance.

We used (1) and (2) to generate Table 3. This table shows the expected numbers of motifs for lengths $8 \leq l \leq 30$. We list $E_t(l, d - 1)$, $E_t(l, d)$, and $E_t(l, d + 1)$ to gauge the solvability of the instances. For example, when $l = 13$ and $d = 4$, a random sequence of length 600 without implanted motifs is expected to have 5.32 spurious motifs by chance, while the same random sequence is expected to have 8.14E-16 spurious motifs with $d = 3$, which is a negligible value.

The cutoffs to distinguish between these categories are vague. For example, Pevzner and Sze in [20] considered the problem instance (15, 4) as a challenge problem. Table 3 shows that *E_t(15, 4)=2.17E-15*, which is very small, *E_t(15, 5)=2.84*, while *E_t(15, 6)=1.81E+08*. So, which of these is challenging and which is unsolvable? They considered instance (15, 4) as a challenge because if there are $d = 4$ mutations in a motif of length 15, then the Hamming distance between any two d-neighbors will be $\leq 8 = 2d$. In this case it will be difficult for the algorithms to retrieve the consensus motif because the distance between the ($d = 4$)-neighbors can be as large as 8, which is more than the half of the motif length. Thus, the cutoff may depend on the relationship between d and l.

Table 3 with highlighted cells shows these *critical* instances. We labeled them as *critical*

Table 2: Last Solvable Problem Instances Depending on E_t Values

l	d	$E_t(l,d)$	l	d	$E_t(l,d)$
9	1	1.46E-19	8	1	8.80E-10
23	8	4.77E-17	22	8	2.02E-09
11	2	5.43E-17	20	7	1.41E-08
21	7	2.51E-16	10	2	6.11E-08
13	3	8.14E-16	18	6	7.11E-08
19	6	9.11E-16	16	5	2.33E-07
17	5	2.00E-15	12	3	3.19E-07
15	4	2.17E-15	14	4	4.20E-07
30	12	1.16E-13	29	12	1.06E-06
28	11	1.65E-12	27	11	1.66E-05
26	10	2.08E-11	25	10	2.28E-04
24	9	2.25E-10			

because it is difficult to discern whether they are unsolvable or challenge instances. Some of the highlighted cell values in Table 3 are larger than one and some are smaller than one. So, we will not consider 1 as a threshold to judge whether an instance is solvable or challenging. We show results of experimentation based on the *critical* instances.

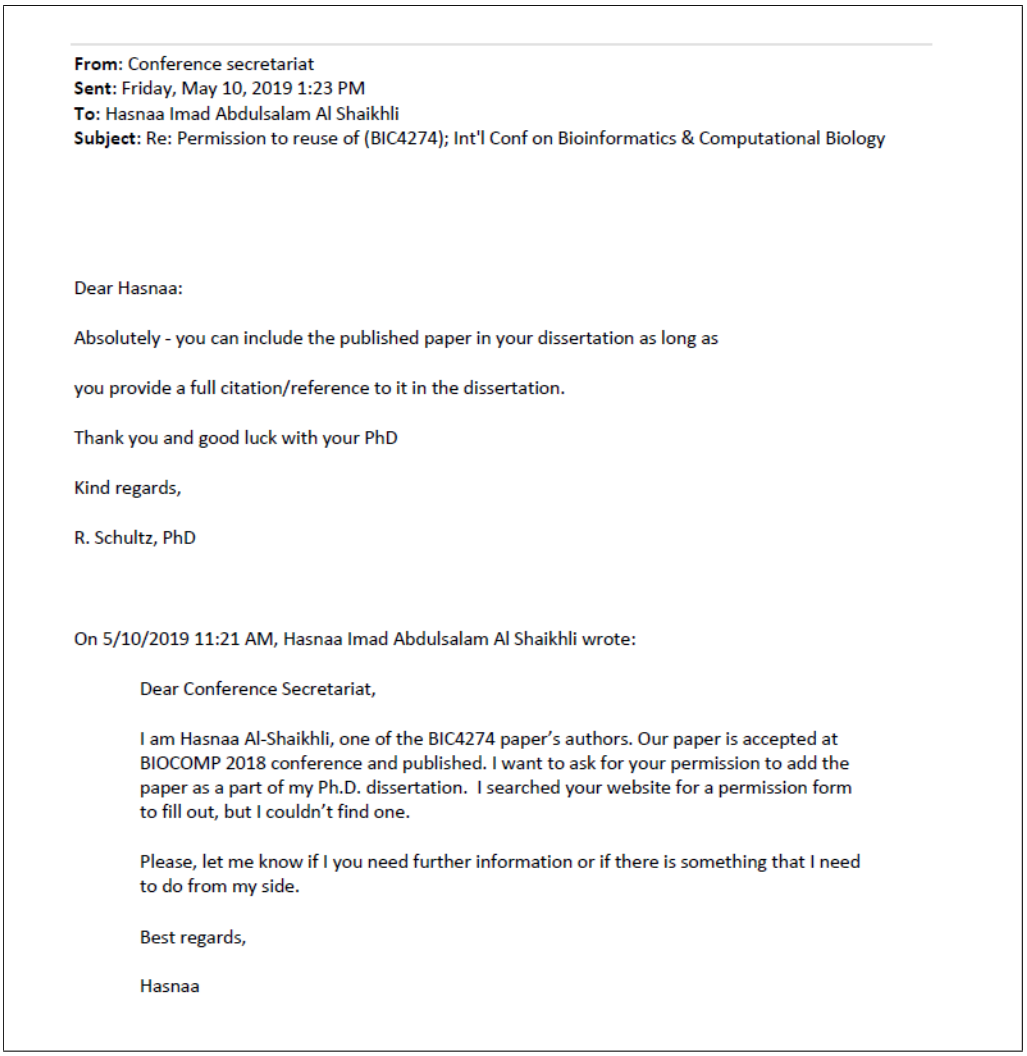
For the experiments we will consider the set of problem instances with relatively high E_t shown in Table 2, where they are sorted in ascending order with respect to the E_t values.

Table 3: Expected Numbers of Motifs $t = 20$, $n = 600$, and $8 \leq l \leq 30$ with the *Critical* Instances

l	d	$E_t(l, d-1)$	$E_t(l, d)$	$E_t(l, d+1)$
8	1	8.11E-37	8.80E-10	1.21E+04
9	2	1.46E-19	1.60E+00	2.49E+05
10	2	6.19E-30	6.11E-08	7.18E+04
11	3	5.43E-17	4.72E+00	3.34E+06
12	3	1.09E-26	3.19E-07	2.25E+05
13	4	8.14E-16	5.23E+00	3.24E+07
14	4	5.05E-25	4.20E-07	3.56E+05
15	5	2.17E-15	2.84E+00	1.81E+08
16	5	3.16E-24	2.33E-07	2.85E+05
17	6	2.00E-15	8.84E-01	4.89E+08
18	6	5.70E-24	7.11E-08	1.23E+05
19	7	9.11E-16	1.77E-01	6.05E+08
20	7	4.47E-24	1.41E-08	3.12E+04
21	8	2.51E-16	2.49E-02	3.57E+08
22	8	1.94E-24	2.02E-09	5.10E+03
23	9	4.77E-17	2.66E-03	1.11E+08
24	9	5.48E-25	2.25E-10	5.87E+02
25	10	6.88E-18	2.28E-04	2.04E+07
26	11	2.08E-11	5.14E+01	1.00E+11
27	11	8.01E-19	1.66E-05	2.46E+06
28	12	1.65E-12	3.64E+00	2.30E+10
29	12	7.87E-20	1.06E-06	2.16E+05
30	13	1.16E-13	2.20E-01	3.24E+09

C. Permissions

This appendix presents a permission from the the 19th International Conference on Bioinformatics and Computational Biology (BIOCOMP 2018) to include the published paper [127] in this dissertation as can be seen in Figure 1.



From: Conference secretariat
Sent: Friday, May 10, 2019 1:23 PM
To: Hasnaa Imad Abdulsalam Al Shaikhli
Subject: Re: Permission to reuse of (BIC4274); Int'l Conf on Bioinformatics & Computational Biology

Dear Hasnaa:

Absolutely - you can include the published paper in your dissertation as long as

you provide a full citation/reference to it in the dissertation.

Thank you and good luck with your PhD

Kind regards,

R. Schultz, PhD

On 5/10/2019 11:21 AM, Hasnaa Imad Abdulsalam Al Shaikhli wrote:

> Dear Conference Secretariat,
>
> I am Hasnaa Al-Shaikhli, one of the BIC4274 paper's authors. Our paper is accepted at BIOCOMP 2018 conference and published. I want to ask for your permission to add the paper as a part of my Ph.D. dissertation. I searched your website for a permission form to fill out, but I couldn't find one.
>
> Please, let me know if I you need further information or if there is something that I need to do from my side.
>
> Best regards,
>
> Hasnaa

Figure 1: BIOCOMP 2018 Conference Permission