An Efficient Exact Solution for the (l, d)-Planted Motif Problem

A Thesis

Presented to the

Faculty of the Graduate School

Ateneo de Manila University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Maria Clara Isabel D. Sia

2015

Abstract

DNA motif finding is widely recognized as a difficult problem in computational biology and computer science. Because of the usual large search space involved, exact solutions typically require a significant amount of execution time before discovering a motif of length l that occurs in each sequence S_i from an input set $\{S_1, ..., S_t\}$ of sequences, allowing for at most d substitutions. In this paper, we propose EMS-GT, a novel algorithm that operates on a compact bit-based representation of the search space and takes advantage of distance-related patterns in this representation to compute the exact solution for any arbitrary problem instance up to l=17. A Java implementation is shown to be highly competitive against PMS8 and qPMS9, two current state-of-the-art exact motif search algorithms. EMS-GT works extremely well for problems involving short motifs, outperforming both competitors for challenge instances with (l, d) values (9,2), (11,3), (13,4) and (15,5), showing runtime reductions of 87%, 79%, 77% and 37% respectively for these instances, while ranking second to qPMS9 for challenge instance (17,6).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

Introduction

DNA motif finding is widely recognized as a difficult problem in computational biology and computer science. Motifs are sequences that occur repeatedly in DNA and have some biological significance [3]; a motif might be a transcription factor binding site, a promoter element, a splicing site, or a marker useful for classification. There are many variants of motif finding problem in the literature. Some look for a motif that repeatedly occurs in a single sequence. Others look for a motif that occurs over some or all of a set of DNA sequences [4]. One of the latter type is the planted motif problem.

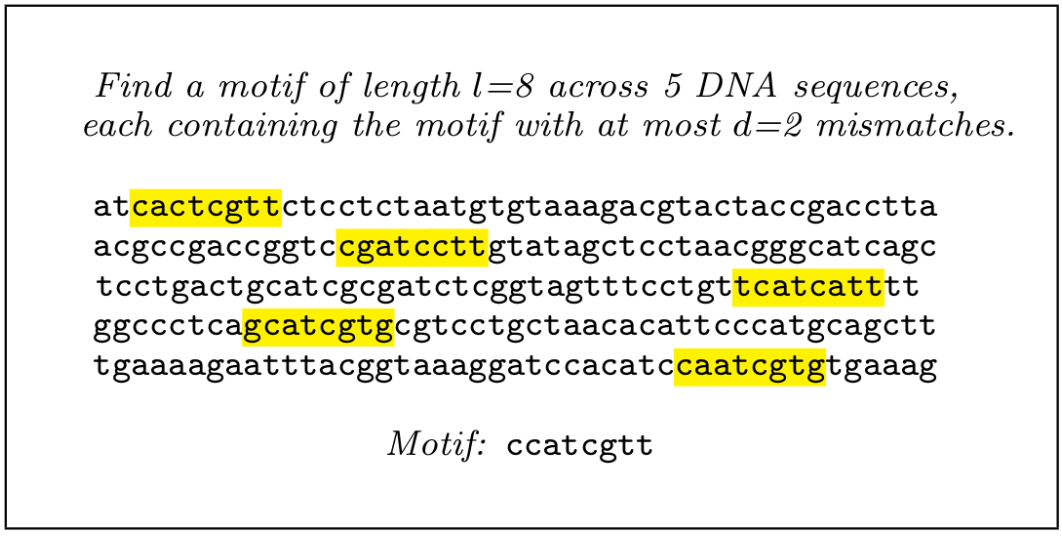

Figure 1.1. Sample instance of the planted motif problem.

The planted motif problem simply asks: "*Given a set of DNA sequences, can we find an unknown motif of length l that appears at different positions in each of the sequences* [13]?" Initially it seems an exhaustive string search will suffice for this problem. However, due to biological mutation, motif occurrences in DNA are allowed to differ from the original motif by up to d characters. This greatly impacts complexity: two distinct variants of a motif–both counting as valid occurrences of the motif–might differ in as many as $2d$ characters! Brute-force solutions quickly become infeasible as values of l and d increase. All of this shows why (l, d)-motifs are sometimes called "subtle" signals in DNA [13], and why finding them is difficult and computationally expensive. In fact, the motif finding problem has already been shown to be NP-complete [11].

This paper proposes EMS-GT, a novel algorithm that solves the planted motif problem. It operates on a compact bit-based representation of the search space, and takes advantage of distance-related patterns in this representation to compute the exact solution for any arbitrary problem instance up to l=17.

1.1 Context of the Study

This section formally defines the planted motif problem. It also defines key terms used throughout this paper in discussing exact motif-search algorithms.

DEFINITION 1. l-mer, Hamming distance, d-neighborhood

An ***l*-mer** is a sequence of length l. Given a sequence S of length $L > l$, the i^{th} l-mer in S starts at the i^{th} position. The **Hamming distance dH** between two l-mers of equal length is the number of characters that differ between them.

Ex. If l = 5, the second l-mer in `gattaca` is `attac`.

$dH(\texttt{gattaca}, \texttt{cgttaga}) = 3$.

The ***d*-neighborhood of an *l*-mer x** is the set $N(x, d)$ of all d-neighbors of x: all l-mers x' whose Hamming distance from x is at most d, i.e., $dH(x, x') \leq d$. Meanwhile,the ***d*-neighborhood of a sequence *S*** of length $L > l$ is the set $\mathcal{N}(S, d)$ of all d-neighbors of all the l-mers in S.

Ex. `gatct`, `cctta`, and `aatta` are all in $N(\texttt{gatta}, 2)$.

For l = 5, $\mathcal{N}(\texttt{gattaca}, 2) = N(\texttt{gatta}, 2) \cup N(\texttt{attac}, 2) \cup N(\texttt{ttaca}, 2)$.

DEFINITION 2. (l, d) Planted Motif Problem

We formally define the (l,d) planted motif problem as follows:

Given a set $S = \{S_1, ...S_t\}$ of n DNA sequences of length L,

find M, the set of sequences (or motifs) of length $l < L$

which have at least one d-neighbor in each sequence in S.

1.2 Research Questions

1.

2.

3.

1.3 Objectives of the Study

1.

2.

3.

1.4 Significance of the Study

1.5 Scope and Limitations of the Study

CHAPTER II

Review of Related Literature

Motif finding is a well-studied problem in computing. Various motif search algorithms have been developed, falling into two categories: *heuristic* and *exact*.

2.1 Heuristic Algorithms

Heuristic algorithms perform an iterative local search, for instance by repeatedly refining an input sampling or projection until a motif is found. Gibbs sampling [8] and Expectation Maximization (EM), used in the motif-finding tool MEME [9, 1] both use probabilistic computations to optimize an initial random alignment. (An alignment is simply a set $\{a_1, a_2, ..., a_n\}$ of n positions, which predicts that the motif occurs at position a_i in the given sequence S_i.) Gibbs sampling tries to refine the alignment one position at a time; EM may recompute the entire alignment in a single iteration. Projection [2] combines a pattern-based approach with EM's probabilistic approach, trying to guess every successive character of a tentative motif and using EM to verify its guesses. GARPS [7] uses a random version of projection, in tandem with the Genetic Algorithm (GA), for yet another iterative approach. These are just some of many successful heuristic algorithms.

2.2 Exact Algorithms

Heuristic approaches are non-exhaustive and thus not always guaranteed to find a solution. Exact algorithms, on the other hand, perform an exhaustive search of possible motifs and so always find the planted motif.

WINNOWER [13] and its successor MITRA [6] are exact algorithms that look at pairwise l-mer similarity to find motifs. In a set of DNA sequences, there are numerous pairs of "similar" l-mers, which come from different sequences and have Hamming distances of at most $2d$ from each other (meaning that they could be two d-neighbors of the same l-mer). WINNOWER represents these pairs in a graph, with l-mers as nodes and edges connecting l-mer pairs. It then prunes the graph to identify "cliques" of pairs that indicate a motif. MITRA refines this graph representation into a mismatch tree containing all possible l-mers, organized by prefix. The tree structure allows MITRA to eliminate entire branches at a time, making it faster than WINNOWER at removing the spurious edges that are not part of any motif clique.

The current state-of-the-art in exact motif search is qPMS9, the most recent in a series [5, 11, 12] of Planted Motif Search algorithms. It performs a sample-driven step, which generates a k-tuple of l-mers from each of k input strings, followed by a pattern-driven step, which generates the common d-

neighborhood of the tuple and then checks whether any of the l-mers in this common neighborhood is a motif. To identify neighbors, qPMS9 efficiently traverses the tree of all possible l-mers, using certain pruning criteria explored by predecessors PMSPrune and qPMS7 [5] to quickly discard non-neighbor branches. Sampling in qPMS9 is an improvement on its predecessor PMS8 [11]; in building a k-tuple, qPMS9 intelligently prioritizes l-mers that have fewer matches with the l-mers already selected, such that the common d-neighborhood becomes smaller and thus faster to check through. Finally, both PMS8 and qPMS9 have been implemented to run on multiple processors, allowing them to solve problem instances with (l, d) as large as (50, 21) in a few hours.

2.3 EMS-GT

EMS-GT [10] is an exact motif search algorithm based on the candidate generate-and-test principle. It operates on a compact bit-based representation of the search space, identifying the common d-neighbors of the n given DNA sequences as motifs. The main idea of EMS-GT is to narrow down the search space to a small set of "candidate" motifs based on the first n' sequences, then do a brute-force search for each candidate on the remaining $(n - n')$ sequences to confirm whether or not it is a motif. This approach proceeds in two main steps:

1. *Generate candidates*

   This step takes the intersection of the d-neighborhoods of the first n' sequences $S_1, S_2, ..., S_{n'}$. Every l-mer in the resulting set C is a candidate motif.

$$C = \mathcal{N}(S_1, d) \cap \mathcal{N}(S_2, d) \cap ... \cap \mathcal{N}(S_{n'}, d). \tag{2.1}$$

2. *Test candidates*

   This step simply checks each candidate motif c in C, to determine whether a d-neighbor of c appears in all of the remaining sequences $S_{n'+1}, S_{n'+2}, ..., S_n$. If this is the case, c is accepted as a motif in set M.

$$M = C \cap \mathcal{N}(S_{n'+1}, d) \cap ... \cap \mathcal{N}(S_n, d). \tag{2.2}$$

Algorithm 1 EXACT MOTIF SEARCH - GENERATE AND TEST

Input: set of L-length sequences $S = \{S_1, S_2, ..., S_n\}$,
motif length l, allowable mismatches d

Output: set of candidate motifs **M**

```
C ← {}                                              ▷ generate candidates
N(S_1, d) ← {}
for j ← 1 to L − l + 1 do
    x ← j^th l-mer in S_1
    N(S_1, d) ← N(S_1, d) ∪ N(x, d)
end for
C ← N(S_1, d)
for i ← 2 to n' do
    N(S_i, d) ← {}
    for j ← 1 to L − l + 1 do
        x ← j^th l-mer in S_1
        N(S_i, d) ← N(S_i, d) ∪ N(x, d)
    end for
    C ← C ∩ N(S_i, d)
end for
M ← {}                                              ▷ test candidates
for each l-mer u in C do
    isMotif ← true
    for i ← (n' + 1) to n do
        found ← false
        for j ← 1 to L − l + 1 do
            x ← j^th l−mer in S_i
            compute dH(x, u)
            if dH(x, u) ≤ d then
                found ← true
                break
            end if
        end for
        if !found then
            isMotif ← false
            break
        end if
    end for
    if isMotif then
        M ← M ∪ u
    end if
end for
```

CHAPTER III

Theoretical Framework

This chapter discusses how EMS-GT currently performs fast operations on a bit-based representation of the motif search space. It then presents a proof for certain regular properties of the Hamming distance-based patterns that were observed to appear in this representation. These properties form the basis of a novel optimization technique for EMS-GT, which is described in Chapter 4.

3.1 EMS-GT efficiency strategies

EMS-GT has efficiency strategies for four key tasks: (1) representing the search space with bits; (2) searching the bit-based representation; (3) determining whether two l-mers are neighbors; and (4) generating all the neighbors for a given l-mer.

3.1.1 Bit-based set representation and l-mer enumeration

The motif search space consists of the 4^l possible l-mers that can be formed from the nucleic alphabet {a, g, c, t}. To efficiently represent sets—such as a d-neighborhood, or a set of candidate motifs—within this space, EMS-GT assigns each of the 4^l l-mers a bit flag in an array, set to 1 if the l-mer is a member of the set and 0 otherwise. Bit flags correspond to l-mers via a simple mapping:

EMS-GT maps an l-mer s to a bit flag index x by replacing each character with 2 bits (`a=00`, `c=01`, `g=10`, `t=11`). Note that this mapping scheme enumerates l-mers in strict alphabetical order.

Ex. `tacgt` maps to `1100011011` = 795 in decimal.

Thus, the flag for `tagct` is bit 795 in the array.

3.1.2 Bit-array compression

EMS-GT's implementation compresses the required set-representation array of 4^l bits into an equivalent array of $\frac{4^l}{32}$ 32-bit integers. The x^{th} bit is now found at position (x mod 32) of the integer at array index $\frac{x}{32}$.

Ex. `tacgt` maps to `1100011011` = 795 in decimal.

array index = $\frac{795}{32}$ = 24 , *bit position* = 795 mod 32 = 27;

Thus, the flag for `tacgt` is the 27^{th} bit of the integer at array index 24.

3.1.3 XOR-based Hamming distance computation

The mapping of l-mers to binary numbers is also useful for computing Hamming distances. An exclusive OR (XOR) bitwise operation between the mappings of two l-mers will produce a nonzero pair of bits at every mismatch position; counting these nonzero pairs of bits in the XOR result gives us the Hamming distance. This is implemented in Algorithm 2.

Ex. `tacgt` maps to `1100011011`

`ttcgg` maps to `1111011010`

XOR produces `0011000001` = 2 mismatches.

3.1.4 Recursive neighborhood generation

To generate a d-neighbor of an l-mer x, we choose $d' \leq d$ positions from 1, 2,..., l-1, l and change the character at each of the d' positions in x. EMS-GT uses a recursive procedure (Algorithm 3) to do this, effectively (1) traversing the tree of all d-neighbors and (2) setting the bit flag in the neighborhood array N for each neighbor it encounters. Since we choose up to d positions in the l-mer, and have 3 possible substitute characters at each position, the size of the neighborhood $N(x, d)$ is given by:

$$|N(x,d)| = \sum_{i=0}^{d} \binom{l}{i} 3^i \tag{3.1}$$

Algorithm 2 HAMMING DISTANCE COMPUTATION

Input: l-mer mappings u and v
Output: $dH(u, v)$

$dH(u, v) = 0$
$z \leftarrow u^v$
for $i \leftarrow 1$ to l **do**
    if z & 3 != 0 **then**
        $dH(u, v) \leftarrow dH(u, v) + 1$
    end if
    $z \leftarrow z >> 2$
end for
return $dH(u, v)$

Algorithm 3 RECURSIVE NEIGHBORHOOD GENERATION

Input: DNA sequence S, motif length l, mismatches d
Output: bit-array $\mathcal{N}$ representing $\mathcal{N}(S, d)$

$\mathcal{N} \leftarrow \{\}$
for each l-mer x in S **do**
    ADDNEIGHBORS(x, 0, d) ▷ *recursive procedure*
end for
▷ *make d changes in l-mer x, from position s onward*
procedure ADDNEIGHBORS(x, s, d)
    for $i \leftarrow s$ to l **do**
        $\Sigma \leftarrow \{\mathtt{a}, \mathtt{g}, \mathtt{c}, \mathtt{t}\} - x_i$ ▷ *i^{th} character in x*
        for $j \leftarrow 1$ to $|\Sigma|$ **do**
            $neighbor \leftarrow x_{1\ldots i-1} + \Sigma_j + x_{i+1\ldots l}$
            $\mathcal{N}[neighbor] \leftarrow 1$
            if $d > 1$ and $i < l$ **then**
                ADDNEIGHBORS($neighbor$, $i + 1$, $d - 1$)
            end if
        end for
    end for
end procedure

3.2 Proof for block-based patterns in Hamming distances

We can represent the neighborhood $N(x, d)$ of an l-mer x as an array N of 4^l bit flags, set to 1 if the corresponding l-mer is a neighbor and 0 otherwise.

$$N_{x'} = \begin{cases} 1 & \text{if } dH(x, x') \leq d, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for any } l\text{-mer } x'.$$

We aim to prove that if we divide this bit array N into consecutive blocks of 4^k flags each, for some **block degree** $\boldsymbol{k}$, $0 < k < l$, each non-empty block (i.e., not all zeros) will conform to $\min(d, k + 1)$ possible bit patterns.

Say we have a bit-array representing $N(x, d)$ for a certain l-mer $x = yz$, where y is x's **prefix** (first $l-k$ characters) and z is its ***k*-suffix** (last k characters).

Ex. For k = 5, x = `acgtacgtacgt` → y = `acgtacg` and z = `tacgt`.

For any other l-mer $x' = y'z'$in the search space

CHAPTER IV

Methodology

4.1 Datasets

Synthetic datasets were created using a DNA sequence generator written in Java. Each nucleotide character in a sequence is randomly generated; {a, g, c, t} each have a 25% chance of being selected, independent from other characters in the sequence. The motif is then planted at a random position in the sequence. As prescribed in [13] every dataset contains 20 DNA sequences each 600 bases long, with an (l, d) motif planted exactly once in each sequence.

4.2 Implementation

Thee Java implementation of EMS-GT operates on a compact, bit-based enumerative representation of the motif search space. Since a significant part of runtime is spent locating and setting bits in the bit-based representation, optimizations were explored for the bit-setting portion of the algorithm. Investigation of some Hamming distance-based patterns in the search space led to a bit-masking optimization which exploited these patterns to set bits more quickly and in contiguous blocks.

4.3 Evaluation

EMS-GT was compared to known state-of-the-art algorithms PMS8 and qPMS9 by benchmarking their performance on challenging instances of the (l, d) planted motif problem. An (l, d) problem instance is defined to be a challenging instance if d is the largest value for which the expected number of l-length motifs that would occur in the input by random chance does not exceed some limit—typically 500 random motifs [12]. The specific challenge instances used were (9,2), (11,3), (13,4), (15,5), and (17,6), as identified in [12, 5].

CHAPTER V

Results and Analysis

5.1 Procedure for block-based neighborhood generation

We can represent the neighborhood $N(x, d)$ of an l-mer x as an array N of 4^l bit flags, set to 1 if the corresponding l-mer is a neighbor and 0 otherwise.

$$N_{x'} = \begin{cases} 1 & \text{if } dH(x, x') \leq d, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for any } l\text{-mer } x'.$$

We find that if we divide this bit array N into consecutive blocks of 4^k flags each, for some k, $0 < k < l$, each block will conform to one of at most $(k + 2)$, possible bit patterns. We exploit this regularity in order to generate N in blocks. Say we wish to generate $N(x, d)$ for some l-mer x. We perform the following steps:

1. We select a value for the **block degree** $\boldsymbol{k}$, and divide x into its **prefix** $\boldsymbol{y}$ (first $l - k$ characters) and its $\boldsymbol{k}$**-suffix** $\boldsymbol{z}$ (last k characters).

   Ex. Choose k = 5; x = `acgtacgtacgt` $\rightarrow$ y = `acgtacg` and z = `tacgt`.

2. We generate the **distribution** $\boldsymbol{\mathcal{D}(z)}$ of Hamming distances from z to all 4^k possible k-suffixes, in which $\mathcal{D}(z)_{z'} = dH(z, z')$ for any k-suffix z'.

3. For a given block in N, we compute the **prefix distance** $d_{y'}$, which is simply the Hamming distance $dH(y, y')$ between x's prefix y and the block's common prefix y'. (The l-mers of a block in N all have the same prefix, due to the strictly alphabetical enumeration scheme.)

4. Given $d_{y'}$ and $\mathcal{D}(z)$, we can compute the Hamming distance from x to any l-mer $x' = y'z'$ in the given block as:

$$dH(x, x') = d_{y'} + \mathcal{D}(z)_{z'} \tag{5.1}$$

5. With the above equation, we redefine the criteria for setting a bit in N:

$$N_{x'} = \begin{cases} 1 & \text{if } d_{y'} + \mathcal{D}(z)_{z'} \leq d, \\ 0 & \text{otherwise.} \end{cases} \quad \text{for } x' = y'z'.$$

6. From this we see that a bit at position z' within a block with prefix y' will be set if and only if $D(z)_{z'} \leq d - d_{y'}$. Knowing that the values in $D(z)$ range from 0 to k, we have three cases for the value of $d - d_{y'}$:

   when $d - d_{y'} < 0$, no bits in the block are set;

   $0 \leq d - d_{y'} < k$, some bits are set (there are k possible configurations);

   $d - d_{y'} \geq k$, all bits in the block are set.

   This results in a maximum of $(k + 2)$ possible patterns of bits. Note that, if

we choose k greater than d, only up to $(d+1)$.

EMS-GT and two competitor algorithms were run on an Intel Xeon, insert details machine. Their performance, averaged over 20 synthetic datasets for each (l, d) challenge instance, is outlined in Table 1:

(l, d)	**PMS8**	**qPMS9**	**EMS-GT**	**% speedup**
9,2	0.74 s	0.47 s	**0.11 s**	76.6%
11,3	1.58 s	1.06 s	**0.20 s**	81.1%
13,4	5.39 s	4.52 s	**1.04 s**	77.0%
15,5	36.45 s	24.63 s	**15.51 s**	37.0%
17,6	3.91 min	**1.96 min**	*2.93 min*	–

Table 5.1. Runtimes of PMS8, qPMS9 and EMS-GT

For every challenge instance except (17,6) EMS-GT outperforms qPMS9; it outperforms PMS8 for instance (17,6). EMS-GT was run including the block-masking optimization, with the default suffix length of k=5. Observe that our EMS-GT implementation can only solve problem instances where $l \leq 17$. This is because when we reach l=18, the size of the integer array needed to represent the entire search space ($\frac{4^{18}}{32} = \frac{2^{36}}{2^5} = 2^{31}$ integers) begins to exceed the maximum size for Java arrays, which is $(2^{31} - 1)$ elements.

CHAPTER VI

Conclusions

BIBLIOGRAPHY

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

[2] Mathieu Blanchette and Martin Tompa. Discovery of regulatory elements by a computational method for phylogenetic footprinting. *Genome research*, 12(5):739–748, 2002.

[3] Modan K Das and Ho-Kwok Dai. A survey of dna motif finding algorithms. *BMC bioinformatics*, 8(Suppl 7):S21, 2007.

[4] Naga Shailaja Dasari, Ranjan Desh, and Mohammad Zubair. An efficient multicore implementation of planted motif problem. In *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, pages 9–15. IEEE, 2010.

[5] Jaime Davila, Sudha Balla, and Sanguthevar Rajasekaran. Fast and practical algorithms for planted (l, d) motif search. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 4(4):544–552, 2007.

[6] Eleazar Eskin and Pavel A Pevzner. Finding composite regulatory patterns in dna sequences. *Bioinformatics*, 18(suppl 1):S354–S363, 2002.

[7] Hongwei Huo, Zhenhua Zhao, Vojislav Stojkovic, and Lifang Liu. Combining genetic algorithm and random projection strategy for (l, d)-motif discovery. In *Bio-Inspired Computing, 2009. BIC-TA'09. Fourth International Conference on*, pages 1–6. IEEE, 2009.

[8] Charles E Lawrence, Stephen F Altschul, Mark S Boguski, Jun S Liu, Andrew F Neuwald, and John C Wootton. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *science*, 262(5131):208–214, 1993.

[9] Charles E Lawrence and Andrew A Reilly. An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins: Structure, Function, and Bioinformatics*, 7(1):41–51, 1990.

[10] Julieta Q. Nabos. *New Heuristics and Exact Algorithms for the Planted DNA (l, d)-Motif Finding Problem*. PhD thesis, Ateneo de Manila University, 2015.

[11] Marius Nicolae and Sanguthevar Rajasekaran. Efficient sequential and parallel algorithms for planted motif search. *BMC bioinformatics*, 15(1):34, 2014.

[12] Marius Nicolae and Sanguthevar Rajasekaran. qpms9: An efficient algorithm for quorum planted motif search. *Scientific reports*, 5, 2015.

[13] Pavel A Pevzner, Sing-Hoi Sze, et al. Combinatorial approaches to finding subtle signals in dna sequences. In *ISMB*, volume 8, pages 269–278, 2000.