View Invariant Gait Recognition For Person Re-Identification in a Multi Surveillance Camera Environment

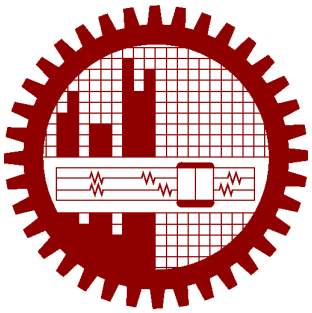
By

Md Mahedi Hasan

1014312019

MASTER OF SCIENCE

IN INFORMATION AND COMMUNICATIN TECHNOLOGY



Institute of Information and Communication Technology

Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

January 2019

CANDIDATE'S DECLARATION

This is to certify that the work presented in this thesis, titled, "View Invariant Gait Recognition For Person Re-Identification in a Multi Surveillance Camera Environment", is the outcome of the investigation and research carried out by us under the supervision of Name of the SupervisorN-2.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Md Mahedi Hasan
1014312019

CERTIFICATION

This thesis titled, **"View Invariant Gait Recognition For Person Re-Identification in a Multi Surveillance Camera Environment"**, submitted by Md Mahedi Hasan, Roll No.: 1014312019, Session: 2019, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of MASTER OF SCIENCE in Information and Communication Technology in January 2019.

BOARD OF EXAMINERS

Signature

Name of the Supervisor — chairman
Designation
Address

Signature

N-2 — Member
D-2
A-2

ACKNOWLEDGEMENT

First and foremost, I express my deepest gratitude to **Almighty Allah** for bestowing His blessings on me and enabled me to successfully accomplish this work.

I would like to express my sincere appreciation and gratitude to my supervisor **Dr. Hossen Asiful Mustafa**, who is a great mentor leading me to the research field of computer vision and deep learning. His cooperation and guidance has been a constant source of encouragement in shaping this work throughout my master thesis period. Without his great help and invaluable suggestions I would hardly finish this thesis.

I will always appreciate my classmates and brothers **Md Abdul Aowal, Abu Noman**, and **Imran Khan** for their firm-backing and support that has greatly helped me in accomplishing the research work. I am also highly grateful for the support of my roommate cum brother **Abdullah Al Mahmud**, who always tolerated my frustration. We studied together and shared a lot of discussion which were very helpful for this research.

Finally, I want to dedicate the essence of my purest respect to my parents and to my colleagues for providing me with support and continuous inspiration throughout my years of study and through the process of writing this thesis. This accomplishment would not have been possible without them. Thank you.

Dhaka
January 2019

Md Mahedi Hasan

Contents

CANDIDATE'S DECLARATION i

CERTIFICATION ii

ACKNOWLEDGEMENT iii

List of Figures vi

List of Tables vii

List of Algorithms viii

ABSTRACT ix

1 Introduction 1

1.1 What is Gait Recognition 3

1.2 Importance of Gait Recognition 3

1.3 Challenges Gait Recognition 3

2 Literature Review 4

2.1 Gait recognition 4

2.2 Deep learning for gait recognition 5

2.3 Pose-Based gait recognition 5

3 Methodology 7

3.1 Notations and Definitions 7

3.2 Previous Works 7

3.2.1 qPMSPrune 9

3.2.2 qPMS7 12

3.2.3 TraverStringRef 15

3.2.4 PMS8 17

3.3 Our Proposal qPMS-Sigma 19

3.3.1 String Compression 19
3.3.2 Motif Finding 20
3.3.3 Parallel Implementation of qPMS-Sigma 20
3.4 Space and Runtime Complexity 20
3.5 Experimental Results 21

4 Results & Discussion **24**
4.1 A Section 24
4.1.1 This is a Subsection 24
4.2 And Another Section 24

5 Conclusion **28**
5.1 Summary of Our Work 28
5.2 Future Prospects of Our Work 28

References **30**

Index **32**

A Codes **33**
A.1 Compressed String 33

List of Figures

3.1 $\mathcal{T}_2(1010)$ with alphabet $\Sigma = \{0, 1\}$ 10
3.2 $\mathcal{G}_2(1010, 1100)$ with alphabet $\Sigma = \{0, 1\}$ 13
3.3 l-mer Compression Example 20
3.4 Run Time Comparison between qPMS-Sigma vs TraverStringRef in the challenging states for q=20 22
3.5 Run Time Comparison between qPMS-Sigma vs TraverStringRef in the challenging states for q=10 23

List of Tables

3.1 Notations and Definitions 8
3.2 Parameter Setting for Testing Data Set 21
3.3 Computational Results for DNA Sequences for $q = 20$ 21
3.4 Computational Results for DNA Sequences for $q = 10$ 22

List of Algorithms

1 qPMSPrune 10
2 qPMSPrune_Tree($k, x_k, p_k, \mathcal{T}$) 11
3 FeasibleOccurrences2($k, x_k, \mathcal{Q}$) 11
4 IsMotif($x, q', \mathcal{T}$) 12
5 qPMS7 13
6 qPMS7_Tree($k, x_0, r, x_k, p_k, \mathcal{T}$) 14
7 FeasibleOccurrences3($x_0, r, x_k, p_k, \mathcal{Q}$) 14
8 TraverStringRef 16
9 TraverStringRef_Tree($k, x_0, r, x_k, p_k, \mathcal{T}, J$) 17

ABSTRACT

Recognizing individual people from gait is still a challenging problem in computer vision research due to the presence of various covariate factors like varying view angle, change in clothing, walking speed, and load carriage, etc. Most of the earlier works were based on human silhouettes which have proven to be efficient in recognition but are not invariant to change in illumination and clothing. In this research, to address this problem, we present a simple yet effective approach for robust gait recognition using a recurrent neural network (RNN). Our RNN network with GRU architecture is very powerful in capturing the temporal dynamics of the human body pose sequence and perform recognition. We also design a low-dimensional gait feature descriptor based on the 2D coordinates of human pose information which is proven to be not only invariant to various covariate factors but also effective in representing the dynamics of various gait pattern. The experimental results on challenging CASIA A and CASIA B gait datasets demonstrate that the proposed method has achieved state-of-the-art performance on both single-view and cross-view gait recognition which prove the effectiveness of our method.

Chapter 1

Introduction

Biometrics refers to the automatic identification or authentication of people by analyzing their physiological and behavioral characteristics. Physiological biometrics is related to the shape of body parts like fingerprints, shape of the hand, eye (iris and retina), etc., and is now used as the most stable means for authenticating and identifying people in reliable way. But, for efficient and accurate authentication, these traits require cooperation from the subject along with a comprehensive controlled environmental setup. Hence, these traits are not useful in surveillance systems. Behavioral biometrics such as signatures, gestures, gait, voice, etc., is related to a person's behavior. But, these traits are more prone to changes depending on factors such as aging, injuries, or even mood.

Gait recognition is a behavioral biometric modality that identifies a person based on the walking posture of that person. A unique advantage of gait as a biometric is that it offers recognition at a distance and at low-resolution without any user cooperation. That is why gait biometric signature is now considered the only likely identification method suitable for access control, covert video surveillance and forensic analysis which is not prone to spoofing attacks and signature forgery.

Due to the advantages of gait recognition, the past two decades have witnessed significant improvements of gait recognition system. However, unfortunately, there still exist many challenges that need to be addressed for robust gait recognition. The presence of various covariate factors like viewing angle, clothing, carrying heavy bags, wearing different shoes, variations in walking surface, and occlusions, etc., can badly affect people gait representation and drastically reduce the performances of gait recognition. Traditional body appearance-based methods in gait recognition are sensitive to these covariate factors. Because, the extraction of human silhouettes is affected by lighting changes. Even when the extraction step is performed correctly, the shape of the body

depends significantly on covariate factors. Therefore, appearance-based methods are not completely robust towards these covariates change.

In this work, we consider human 2D pose sequence as our effective gait descriptors because it does not depend on people body appearance and shape. Therefore, it will be less affected by the variation of covariate factors. Again, in deep learning, recurrent neural networks (RNNs) have achieved promising performance in many sequence labeling tasks. The rationale behind their effectiveness for sequence-based tasks is their ability to capture long-range dependencies in a temporal context from sequence. So, the key to our proposed method is to develop a pose-based deep recurrent neural network to recognize human gait by extracting and modeling temporal dynamics of 2D pose sequence data.

For multi-view gait recognition, we also propose a two-stage network in which we first determine the walking direction, i.e. the viewpoint angle of the camera using a 3D convolutional network and later identify the subject using proposed RNN based temporal network trained on that particular angle. Our proposed two-staged network is far simpler and efficient in terms of time and space while outperforming present state-of-the-art networks on multi-view gait recognition.

The main contributions of this paper are summarized as follows:

- We propose a novel RNN network with GRU architecture and devise several strategies to effectively extract and model the temporal dynamics of 2D pose sequence data for robust gait recognition. This work will open a new avenue open for gait recognition which no longer needs to calculate gait energy image (GEI) or expensive optical flows as gait descriptors.
- We also propose a two-stage network for multi-view gait recognition in which we first identify the walking direction by extracting spatio-temporal features using 3D convolution and then performs subject recognition using temporal network trained on that particular angle.
- The proposed pose-based RNN network achieves the best results on two challenging benchmark dataset CASIA A and CASIA B by outperforming other prevailing methods in single-view as well as multi-view gait recognition at a significant margin.

1.1 What is Gait Recognition

1.2 Importance of Gait Recognition

1.3 Challenges Gait Recognition

Chapter 2

Literature Review

Over last two decades, a lot of methods have been studied to develop a robust gait recognition system [1]. However, robust recognition is still challenging due to the presence of large intra-class variations in person's gait which substantially changed the performance of recognition. In this section, we briefly discuss the existing literature proposed for robust gait recognition which are closely related to our work.

2.1 Gait recognition

Gait Recognition techniques can generally be classified into two categories: 1) appearance-based methods, and 2) model-based ones. The former is based on the traditional way of extracting appearance features from human silhouettes. Most of the previous work following this approach [2–4] used silhouette masks as the main source of information and extracted features that show how these mask change. The most popular descriptor of gait used in such work is gait energy image (GEI) [2], a binary mask averaged over the gait cycle of human figure. Though, there are many other alternatives for GEI, e.g. gait entropy image (GEnI) [3], and gait flow images [4] due to it's insensitiveness of incidental silhouettes error, GEI is considered as the most stable and effective gait descriptors.

The second one is, a model-based approach [5–7], which is based on modeling of human body structure and local movement patterns of different body parts. Model-based approaches are generally invariant to various intra-class variations like view angle, clothing, and carrying bag which can greatly affect the appearance of the subject. However, it mainly relies on extraction of body parameters like stride length, height, knee, torso,

and hip from raw video stream which are computational intensive.

2.2 Deep learning for gait recognition

In recent years due to the powerful feature learning abilities of deep neural networks, deep learning-based methods have significantly improved the performance of gait recognition. Convolutional neural network (CNN) based gait recognition methods [8–11] has gained increasing popularity among other deep learning-based methods due to its ability to automatically learn gait features from the given training images. Wu *et al.* [**?**] performed cross-view gait recognition by developing three convolutional layer network using subject's GEI as input. Shiraga *et al.* [9] designed a four-layer CNN network consisting two convolutional layers and two fully connected layers for large-scale gait recognition on OU-ISIR database. In [10], Wolf *et al.* used 3D convolutions for multi-view gait recognition by capturing spatio-temporal features from raw images and optical flow information. In addition, several other methods were proposed where CNN was combined to traditional machine learning algorithm such as PCA, SVM, and Bayesian classifier. For example, In [11] CNN was first used as a feature extractor, thereafter SVM was used to classify those features. In [**?**], Yu *et al.* used generative adversarial nets to design a feature extractor in order to learn the invariant features.

2.3 Pose-Based gait recognition

Recently, deep learning-based methods have also been used in real-time 2D pose estimation from image and video. The task of pose estimation involves estimating the locations of body parts. It can broadly be classified into two categories single-person and multi-person pose estimation. To recognize multi-person pose, Cao *et al.* [**?**] developed a deep neural network based regression to estimate the coordinates of body parts. In this work, we used their pretrained model to get an accurate 2D human pose estimation on our experimental dataset.

With the advent of pose-estimation algorithm in computer vision, the recognition of human gait based on pose information has received much more attention [**?**, **?**] due to its effective representation on gait dynamics and robustness towards covariates condition variations. Feng et al. [**?**] used the human body joint heatmap to describe each frame. They fed the joint heatmap of consecutive frames to long short term memory (LSTM).

Their gait features are the hidden activation values of the last timestep. In [**?**], Liao *et al.* construct a temporal-spatial network (PTSN) to extract gait features from 2D human pose information.

In recent years, a large number of methods in computer vision have been proposed to employ recurrent neural network due to its ability to model the long-term contextual information of temporal sequences effectively in the complicated activity recognition task [**?**, **?**]. Song *et al.* [**?**] proposed an end-to-end spatial and temporal attention model with LSTM for human action recognition from skeleton data. In [**?**], Du *et al.* proposed an end-to-end hierarchical RNN network for skeleton based action recognition. They divided the human skeleton into five different parts and then separately feed them into five subnetworks. Our approach in gait recognition is similar to these approaches of activity recognition. In our work, we design a novel RNN architecture to effectively model the temporal dynamics of human 2D pose sequence.

Chapter 3

Methodology

In this chapter we have discussed our approach to solve the generalized Quorum Planted Motif Search problem. We have proposed some increment technique on some existing algorithms for solving qPMS problem more efficiently. We have also proposed the idea to implement parallelism for solving this problem.

3.1 Notations and Definitions

We have used some common notations to describe all the algorithms described and reviewed. The notations and definitions have been summarized in table table 3.1. The first five notations (Σ, s_i, l, d, q) are the input data of the problem. Other notations have been used in several steps the algorithms.

3.2 Previous Works

There are several exact algorithms to solve the planted motif search problem. For the sake of completeness we will describe the previous works which have motivated us to develop our approach. In this section we will review qPMSPruneI [12], qPMS7 [**?**], Traver String Ref [13] and PMS8 [**?**].

Table 3.1: Notations and Definitions

Notations	Definitions
Σ:	The alphabet
s_i:	The input string of length m over Σ ($1 \leq i \leq n$)
l:	The motif length
d:	The maximum number of mismatches allowed in each occurrences
q:	The minimum number of input strings that should have at least one occurrence.
$\|a\|$:	the length of a string a.
$a[j]$:	The jth letter of a string a.
$a \circ b$:	The string generated by concatenating two strings a and b
$d_H(a, b)$:	The Hamming distance between two strings a and b of the same length. It is given by the number of positions j such that $a[j] \neq b[j]$
s_{ij}^l:	The l-length substring of an input string s_i that starts from the jth position.
S_i:	The set of all the l-length substrings of an input string s_i. $S_i = \{s_{i1}^l, ..., s_{i,ml+1}^l\}$.
S_i^d:	The set of all the $(l+1)$-length substrings of an input string s_i defined by S_i^d= $\{s_{l+1}^{i0}, ..., s_{i,ml+1}^{l+1}\}$. Here, $s_{i0}^{l+1}[1] = s_{i,m-l+1}^{l+1}[l+1] = \emptyset$, and $d_H(\emptyset, \alpha) = \infty$ is assumed for any $\alpha \epsilon \Sigma$.
$\mathcal{B}(a, R)$:	The set of strings in the sphere of radius R centered at a. $\mathcal{B}(a, R)$ = $\{b \mid \|b\| = \|a\|, d_H(a, b) \leq R\}$
$n_{\mathcal{B}}(l, d)$:	$\|\mathcal{B}(a, d)\|$ for an l-length string a.
$x\|_P$:	The substring of x composed by sequencing the letters of x at the positions in a vector P. For example, $x\|_P$ = GCA for x = ACCGAT and $P = (4, 2, 5)$.
$P(j)$:	The jth element of a vector P.
$P_+(j)$:	The j-dimensional vecor composed of the first j elements of a vector P.
$P_-(j)$:	The $(\|P\| - j)$-dimensional vector composed of the last $\|P\| - j$ elements of a vector P.
$R_1(a, b, c)$:	The set of indices j satisfying $a[j] = b[j] = c[j]$.
$R_2(a, b, c)$:	The set of indices j satisfying $a[j] = b[j] \neq c[j]$.
$R_3(a, b, c)$:	The set of indices j satisfying $c[j] = a[j] \neq b[j]$.
$R_4(a, b, c)$:	The set of indices j satisfying $b[j] = c[j] \neq a[j]$.
$R_5(a, b, c)$:	The set of indices j satisfying $a[j] \neq b[j], b[j] \neq c[j]$ and $c[j] \neq a[j]$.

3.2.1 qPMSPrune

Algorithm qPMSPrune for the qPMS problem was proposed by [12]. Algorithm qPMSPrune introduces a tree structure to find the possible motif candidates. Then it uses branch-and-bound technique to reduce the search space and find the expected (l, d) motifs for the given set of inputs. Algorithm qPMSPrune uses the d-neighborhood concept to build the tree structure. The key points of the algorithm is described as follows.

Key Steps of qPMSPruneI

Most of the motif search algorithms combine a sample diven approach with a pattern driven approach. Here in the sample driven part the d-neighborhood ($\mathcal{B}(x_0, d)$) of a given input string x_0 is generated as a tree. Every l-mer in the neighborhood is a candidate motif.

The qPMSPrune algorithm is based on the following observation.

Observation 3.1 *Let M be any (l, d, q)-motif of the input strings $s_1, \ldots, s_n$. Then there exists an i (with $1 \leq i \leq n - q + 1$) and a l-mer $x \epsilon s_i^l$ such that M is in $\mathcal{B}(\S, \lceil)$ and M is a $(l, d, q - 1)$-motif of the input strings excluding s_i.*

Based on the above observation, for any l-mer x, it represents $\mathcal{B}(x, d)$ as a tree $\mathcal{T}_d(x)$. It generates the d-neighborhood of every l-mer x in s_1 by generating the tree $\mathcal{T}_d(x)$. The height of $\mathcal{T}_d(x)$ is d. The root of this tree will have x. If a node t is parent of a node t' then the hamming distance between the strings is $d_H(t, t') = 1$. As a result, if a node is at level h, then its hamming distance from the root is h.
For example, the tree $\mathcal{T}_2(1010)$ with alphabet $\Sigma = \{0, 1\}$ is illustrated in Figure fig. 3.1.

The trees are generated and explored in depth first manner. In pattern driven part, each node is checked whether it is a valid motif or not. While traversing a node t in the tree $\mathcal{T}_d(x)$ in a depth-first manner, the hamming distance between the input strings are calculated incrementally. If q' is the number of input strings s_j such that $d_H(t, s_j) \leq d$. If $q' \geq q - 1$, output t as a motif. Moreover the algorithm prunes the branch if $q" < q - 1$ where $q"$ is the number of input strings s_j such that $d_H(t, s_j) \leq 2d - d_H(t, x)$.

The time and space complexities of algorithm qPMSPrune are given by $O((n - q +$

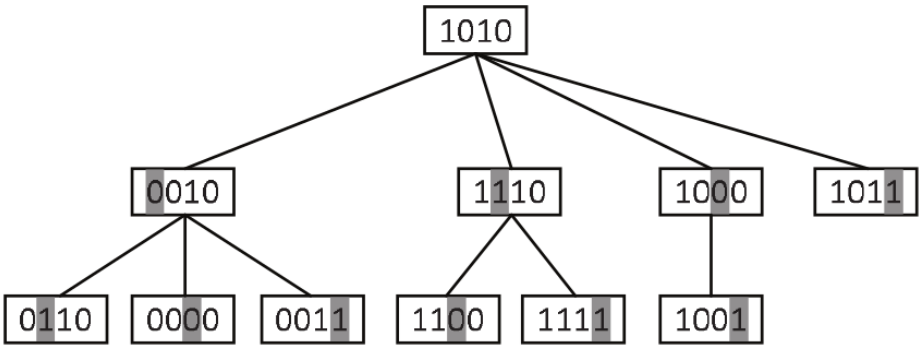


Figure 3.1: $\mathcal{T}_2(1010)$ with alphabet $\Sigma = \{0, 1\}$

$1)nm^2 n_{\mathcal{B}}(l, d))$ and $O(nm^2)$ respectively. The pseudocode of qPMSPrune is shown in algorithm algorithm 1.

Algorithm 1 qPMSPrune

1: $M \leftarrow \emptyset$
2: **for** $i = 1$ **to** $n - q + 1$ **do**
3: **for** $j = 1$ **to** $m - l + 1$ **do**
4: $T \leftarrow \{S_h | 1 \leq h \leq N, h \neq i\}$
5: qPMSPruneI_Tree(0,$s_{ij}^l, 0, T$)
6: **end for**
7: **end for**
8: Output M

Algorithm 2 qPMSPrune_Tree($k, x_k, p_k, \mathcal{T}$)

```
T' ← ∅
for all Q ϵ T do
    Q' ← FeasibleOccurrences2(k, x_k, Q)
    if Q' ≠ ∅ then
        T ← T' ∪ {Q'}
    end if
end for
if |T'| < q − 1 then
    return
end if
if IsMotif(x_k, q − 1, T') = true then
    M ← M ∪ {x_k}
end if
if k = d then
    return
end if
for p_{k+1} = p_k + 1 to l do
    for all α ϵ Σ \ {x_k[p_{k+1}]} do
        x_{k+1} ← x_k
        x_{k+1}[p_{k+1}] ← α
        qPMSPruneI_Tree(k + 1, x_{k+1}, p_{k+1}, T')
    end for
end for
```

Algorithm 3 FeasibleOccurrences2($k, x_k, \mathcal{Q}$)

```
Q' ← ∅
for all y ϵ Q do
    if d_H(x_k, y) ≤ 2d − k then
        Q' ← Q' ∪ {y}
    end if
end for
return Q'
```

Algorithm 4 IsMotif($x, q', \mathcal{T}$)

```
1:  matched ← 0
2:  for all Q ϵ T do
3:      found ← false
4:      for all y ϵ Q do
5:          if d_H(x, y) ≤ d then
6:              found ← true
7:              break the inner loop
8:          end if
9:      end for
10:     if found = true then
11:         matched ← matched + 1
12:         if matched ≥ q' then
13:             return true
14:         end if
15:     end if
16: end for
17: return false
```

3.2.2 qPMS7

Algorithm qPMS7 was proposed by [**?**] which was based on qPMSPrune [12]. Some speedup techniques were introduced to improve the runtime of Algorithm qPMSPrune. qPMS7 also searches for motifs by traversing trees. The primary difference from qPMSPrune is that it utilizes $r \epsilon \mathcal{S}_2$ as well as $x_0 \epsilon \mathcal{S}_1$ to traverse a tree. The algorithm is based on the following observations.

Observation 3.2 Let M be any (l, d, q)-motif of the input strings $s_1, \ldots, s_n$. Then there exist $1 \leq i \neq j \leq n$ and l-mer $x \epsilon s_i^l$ and l-mer $y \epsilon s_j^l$ such that M is in $\mathcal{B}(x, d) \cup \mathcal{B}(y, d)$ and M is a $(l, d, q-2)$-motif of the input strings excluding s_i and s_j.
Using an argument similar to the one in [12], we infer that it is enough to consider every pair of input strings s_i and s_j with $1 \leq i, j \leq (n - q + 2)$. As a result, the above observation gets strengthened as follows.
Observation 3.3 Let M be any (l, d, q)-motif of the input strings $s_1, \ldots, s_n$. Then there exist $1 \leq i \neq j \leq n - q + 2$ and l-mer $x \epsilon s_i^l$ and l-mer $y \epsilon s_j^l$ such that M is in $\mathcal{B}(x, d) \cup \mathcal{B}(y, d)$ and M is a $(l, d, q-2)$-motif of the input strings excluding s_i and s_j.

Algorithm qPMS7 uses a routine based on the above observations which finds all of the possible motifs. Algorithm qPMSPrune explores $\mathcal{B}(x, d)$ by traversing the tree $\mathcal{T}_d(x)$.

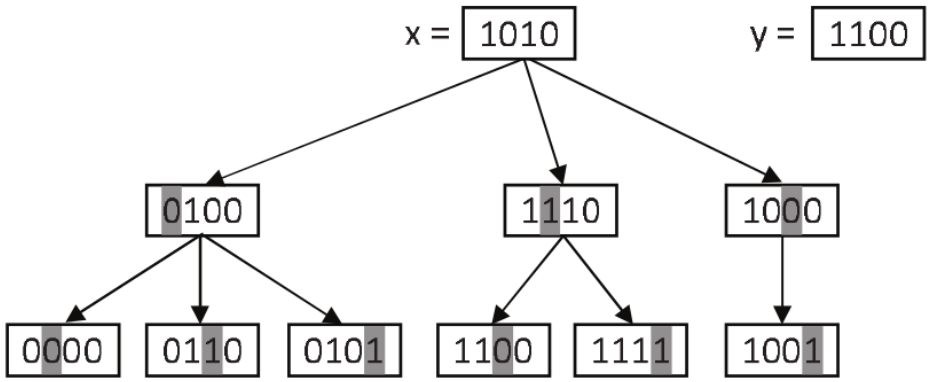


Figure 3.2: $\mathcal{G}_2(1010, 1100)$ with alphabet $\Sigma = \{0, 1\}$

In Algorithm qPMS7, $\mathcal{B}(x, d) \cup \mathcal{B}(y, d)$ is explored by traversing an acyclic graph, denoted as $\mathcal{G}_d(x, y)$ with similar constructing rule as $\mathcal{T}_d(x)$.

The time and space complexity of Algorithm qPMS7 are $O((n-q+1)^2 nm^2 n_{\mathcal{B}}(x, d))$ and $O(nm^2)$ respectively. So in worst case scenario the time runtime of qPMS7 is worse than that of Algorithm qPMSPrune by a factor of $n-q+1$. However, Algorithm qPMS7 is much faster than Algorithm qPMSPrune in practice.

Algorithm 5 qPMS7

$\mathcal{M} \leftarrow \emptyset$
for $i_1 = 1$ **to** $n - q + 1$ **do**
  for $j_1 = 1$ **to** $m - l + 1$ **do**
    for $i_2 = i_1 + 1$ **to** $n - q + 1$ **do**
      for $j_2 = 1$ **to** $m - l + 1$ **do**
        $\mathcal{T} \leftarrow \{S_h | 1 \leq h \leq N, h \neq i_1, h \neq i_2\}$
        qPMS7_Tree$(0, s^l_{i_1,j_1}, s^l_{i_2,j_2}, s^l_{i_1,j_1}, 0, \mathcal{T})$
      end for
    end for
  end for
end for
Output $\mathcal{M}$

Algorithm 6 qPMS7_Tree($k, x_0, r, x_k, p_k, \mathcal{T}$)

$\mathcal{T}' \leftarrow \emptyset$
for all $Q \in \mathcal{M}$ **do**
  $Q' \leftarrow$ FeasibleOccurrences3(x_0, r, x_k, p_k, Q)
  if $Q' \neq \emptyset$ **then**
    $\mathcal{T} \leftarrow \mathcal{T}' \cup \{Q'\}$
  end if
end for
if $|\mathcal{T}'| < q - 2$ **then**
  return
end if
if $d_H(x_k, x_0) \leq d$ and $d_H(x_k, r) \leq d$ and IsMotif($x_k, q-2, \mathcal{T}'$)=**true then**
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{x_k\}$
end if
if $k = d$ **then**
  return
end if
for $p_{k+1} = p_k + 1$ **to** l **do**
  for all $\alpha \in \Sigma \setminus \{x_k[p_{k+1}]\}$ **do**
    $z|_{I_+(p_{k+1}-1)} \leftarrow x_k|_{I_+(p_{k+1}-1)}$
    $z[p_{k+1}] \leftarrow \alpha$
    $x_{k+1}|_{I_+(p_{k+1})} \leftarrow z$
    if $d_H(z, x_0|_{I_+(p_{k+1})}) > d_H(z, r|_{I_+(p_{k+1})})$ **then**
      $x_{k+1}|_{I_-(p_{k+1})} \leftarrow x_k|_{I_-(p_(k+1))}$
    else
      $x_{k+1}|_{I_-(p_{k+1})} \leftarrow r|_{I_-(p_(k+1))}$
    end if
    qPMS7_Tree($k+1, x_0, r, x_{k+1}, p_{k+1}, \mathcal{T}'$)
  end for
end for

Algorithm 7 FeasibleOccurrences3($x_0, r, x_k, p_k, \mathcal{Q}$)

$Q' \leftarrow \emptyset$
$d_x \leftarrow d - d_H(x_k|_{I_+(p_k)}, x_0|_{I_+(p_k)})$
$d_r \leftarrow d - d_H(x_k|_{I_+(p_k)}, r|_{I_+(p_k)})$
for all $y \in Q$ **do**
  $d_y \leftarrow d - d_H(x_k|_{I_+(p_k)}, y|_{I_+(p_k)})$
  if $B(x_0|_{I_-(p_k)}, d_x) \cap B(r|_{I_-(p_k)}, d_r) \cap B(y|_{I_-(p_k)}, d_y) \neq \emptyset$ **then**
    $Q' \leftarrow Q' \cup \{y\}$
  end if
end for
return Q'

3.2.3 TraverStringRef

TraverStringRef is an improved version of qPMS7. Four improvements were proposed for which will be explained below:

Feasibility Check without Precomputed Table

Like qPMS7 feasibility check is not performed by checking an occurrnce in the precomputed table. Here a theorem is followed: Three strings a, b, c of the same length satisfy

$$\mathcal{B}(a, d_a) \cap \mathcal{B}(b, d_b) \cap \mathcal{B}(c, d_c) \neq \emptyset$$

if and only if

$$\begin{gathered} d_a \geq 0, d_b \geq 0, d_c \geq 0, \\ d_a + d_b \geq d_H(a, b), \\ d_b + d_c \geq d_H(b, c), \\ d_c + d_a \geq d_H(c, a), \\ d_a + d_b + d_c \geq |R_2(a, b, c)| + |R_3(a, b, c)| + |R_4(a, b, c)| + |R_5(a, b, c)| \end{gathered}$$

Elimination of unnecessary Combinations

Unnecessary combinations are eliminated in qPMS7 and the procedure is quite same as qPMSPruneI. The unnecessary checks are suppressed when $q < n$ and $\mathcal{T} \leftarrow \{S_h | i_2 + 1 \leq h \leq n\}$.

String Reordering

This improvement is ensured by pruning the subtree rooted at current node as early as possible by checking the feasibility occurences in non decreasing order. Here the difference is when the input string from where the input reference occurences are taken is determined. Since the number of string in $\mathcal{B}(x_0, d) \cap \mathcal{B}(r, d)$ is a non decreasing function of $d_H(x_0, r)$ the reference r is taken from the input that maximizes minimum hammimg distance between x_0 and r to reduce size of the search tree.

Position Reordering

To ensure efficient pruning of subtrees the search tree's structure is investigated. Only the nodes satisfying the condition of not changing the hamming distance change the structure of the search tree. To take the advantage of this procedure the position of the strings reordered so that the positions where the letters are different come earlier. For this an l-dimensional vector is computed for each pair at the root node of the search tree.

The pseudo-code of the algorithm TraverStringRef is given in Algorithm [algorithm 8].

Algorithm 8 TraverStringRef

$\mathcal{M} \leftarrow \emptyset$
for $i_1 \leftarrow 1$ **to** $n - q + 1$ **do**
  for $j_1 \leftarrow 1$ **to** $m - l + 1$ **do**
    $T \leftarrow \{S_h | i_1 + 1 \leq h \leq n\}$
    Sort the elements of $\mathcal{T}$ in the nonincreasing order
    while $|\mathcal{T}| \geq q - 2$ **do**
      $\mathcal{R} \leftarrow$ first element of $\mathcal{T}$
      $\mathcal{T} \leftarrow \mathcal{T} \backslash \mathcal{R}$
      for all $r \in \mathcal{R}$ **do**
        Initialize the vector J
        if $d_H(s^l_{i_1 j_1}, r) \leq 2d$ **then**
          TraverStringRef_Tree(0,$s^l_{i_1 j_1}, r, s^l_{i_1 j_1}, 0, \mathcal{T}, J$)
        end if
      end for
    end while
  end for
end for
Output $\mathcal{M}$

Algorithm 9 TraverStringRef_Tree($k, x_0, r, x_k, p_k, \mathcal{T}, J$)

$\mathcal{T}' \leftarrow \emptyset$
missed $\leftarrow 0$
for all $Q \in T$ **do**
  $Q' \leftarrow$ FeasibleOccurrencesReordered3($k, x_0, r, x_k, p_k, Q, J$)
  if $Q' \neq \emptyset$ **then**
    $\mathcal{T} \leftarrow \mathcal{T}' \cup \{Q'\}$
  else
    missed $\leftarrow$ missed$+ 1$
    if missed $> |\mathcal{T}| - q + 2$ **then**
      return
    end if
  end if
end for
if IsMotifFast($x_k, q-2, \mathcal{T}'$) = **true then**
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{x_k\}$
end if
if $k = d$ **then**
  return
end if
if $|\mathcal{T}'| < q - 2$ **then**
  return
end if
Sort the elements of $\mathcal{T}'$ in the nonincreasing order of $\min_{y \in \mathcal{Q}} d_H(x_k, y)$(20) {The reference needs to be corrected}
for $p_{k+1} = p_k + 1$ **to** l **do**
  for all $\alpha \in \Sigma \setminus \{x_k[J(p_{k+1})]\}$ **do**
    $x_{k+1} \leftarrow x_k$
    $x_{k+1}[J(p_{k+1})] \leftarrow \alpha$
    $d_0 \leftarrow d + d_H(x_0|_{J_-(p_{k+1})}, r|_{J_-(p_{k+1})})$
    if $d_H(x_{k+1}, r) \leq \min(d_0, 2d - k - 1)$ **then**
      TraverStringRef_Tree($k+1, x_0, r, x_{k+1}, p_{k+1}, \mathcal{T}', J$)
    end if
  end for
end for

3.2.4 PMS8

The key concepts of increasing efficiency in PMS8 from qPMS7 and qPMSPrune are described below:

Sort rows by size

Sorted rows speed up the filtering step. As for sorted rows we need less tuples for lower stack size, it helps to reduce expensive filtering. It is because fewer l-mers remain to be filtered when the stack size increases.

Compress l-mers

To calculate the hamming distance between two l-mers at first we have to perform exclusive or of their compressed representation. One compressed l-mer requires $1 \times \lceil \log |\Sigma| \rceil$ bits of storage. So the table of compressed l-mers only requires O(n(m-l+1)) words of memory as we only need the first 16 bit of this representation.

Preprocess distances for pairs of l-mers

For every pair of l-mers we test if the distance is no more than 2d in advance.

Cache locality

As a certain row of an updated matrix is the subset of that row so we can store the updated row in the previous location of the row. so we have to keep track the number of elements belongs to the new row. This process can be repeated in every step of recursion and can be perform using by only a single stack where the subset elements are in contiguous position of memories and thus cache locality sustains.

Find motifs for a subset of strings

we will find the motif for some input strings and then test them against the remaining strings.

Memory and Runtime

since we store all the matrices in the place of a single matrix they only require $O(n(n-l+1))$ words of memory. $O(n^2)$ words of row size will be added for at most n matrices which share the same space. So total bits of l-mer pair takes $O((n(m-l+1)^2)/w)$

words where w is the number of bits in a machine word. So total memory used for this algorithm is $O(n(n-l+1)+(n(m-l+1)^2)/w)$.

Parallel implementation

If we think dividing the problem into $m-l+1$ subproblems then the number of subproblems will be embarrassingly greater for parallelization. So a fixed number of subproblems are assigned to each processor. Scheduler then spawns a separate worker thread to avoid use of a processor just for scheduling. The scheduler loops until all the subproblems are solved and after the completion of all threads the motifs are given to scheduler and it outputs the result.

3.3 Our Proposal qPMS-Sigma

In this section we propose an algorithm qPMS-Sigma with some techniques to optimize the space complexity as well as the runtime to solve the Quorum Planted Motif Search problem. Our work is mainly based on the Algorithm TraverStringRef.

3.3.1 String Compression

We can compress the input strings $s_1, \ldots s_n$ as a part of preprocessing of the algorithm. For our work we have compressed all the l-mers of the input strings into some group of unsigned 32-bit integers. Thus we make the motif matrix using the compressed l-mers. To be specific for an alphabet set Σ we compressed a l-mer into $\lceil \log_2(|\Sigma|) \rceil$ groups. Each of this group contains $\lceil l/32 \rceil$ 32-integers.

For example, for a DNA string of 32 characters, we need $\lceil \log_2(|\Sigma|) \rceil = 2$ groups of unsigned 32-bit integers, where each group contains $\lceil n/32 \rceil = 1$ integers.

We can calculate the hamming distance between two compressed l-mers in two steps. First for each group of integers find out the bitwise XOR values of the integers of both the l-mers. After that calculate the bitwise OR values among the XOR-ed values of each group. The number of set bits in the last set of integers is the hamming distance between the two strings.

An example of comparing two compressed l-mers is given in fig. 3.3.

String	After Ordering the Characters	Group 1	Group 0	XOR 1	XOR 0	OR	Number of Set Bits
AGGCTA	022130	011010	000110	110001	000011	110011	4
GAGCAT	202103	101011	000101				

Figure 3.3: l-mer Compression Example

A code snippet appendix A.1 is written in C++ describing the structure of compressed l-mer and the method of calculating hamming distance.

3.3.2 Motif Finding

After compressing the input sequences we need to find the motifs in them. For these we've modified the TraverStringRef algorithm by using the compressed l-mers. Then we have followed the steps of Algorithm TraverStringRef to find out the motifs.

3.3.3 Parallel Implementation of qPMS-Sigma

The Algorithm qPMS-Sigma can easily implement in parallel by traversing several search trees in parallel. Similar to the idea of parallelism mentioned in Algorithm TraverStringRef [13] we can divide the problem into $(m - l + 1)$ subproblem where each subproblem explores different search trees.

3.4 Space and Runtime Complexity

Here the space complexity of our algorithm is $O(nml(\log\lceil\Sigma\rceil)/w)$. Here w is the size of the integers that contain the compressed input sequences as well as the l-mers. For DNA sequences $\log|\Sigma| = 2$. So the space complexity becomes $O(nml/w)$.

The worst case time complexity is similar to TraverStringRef $O((n-q+1)^2 nm^2(m + \log n)n_{\mathcal{B}}(l,d))$. However, finding the hamming distance between two l-mers take about constant time for DNA sequence. Normally bitwise operations are a little faster than

Table 3.2: Parameter Setting for Testing Data Set

Parameter	Setting
$\|\Sigma\|$	4 (DNA)
n	20
m	600
q	10,20
l	13, 15, 17,...

Table 3.3: Computational Results for DNA Sequences for $q = 20$

Algorithm	**(13,4)**	**(15,5)**	**(17,6)**	**(19,7)**	**(21,8)**	**(23,9)**
qPMS-Sigma	12s	50s	165s	755s	2909s	12153s
TraverStringRef	11s	47s	179s	745s	3098s	13027s
qPMS7	14s	133s	1046s	8465s	71749s	
PMS8	7s	48s	312s	1596s	5904s	19728s
qPMS9	6s	34s	162s	804s	2724s	8136s

the other arithmetic operations. So, practically our algorithm runs a little faster than the former.

3.5 Experimental Results

We have compared the runtime of Algorithm qPMS-Sigma in the challenging states with the other well known algorithms. The proposed algorithm is implemented in C++ (using GNU C++ compiler). We have run the experiment on Ubuntu 14.04 (64 bit) operating system. The machine configuration is Intel®Core™i3-4005U CPU @ 1.70GHz 4, 4GB RAM.

The test dataset was generated in computational experiment of Algorithm TraverStringRef [13]. The testing dataset is randomly generated according to the FM (fixed number of mutation) model [14]. We have used it in our experiment and showed the result in table 3.3 and table 3.4. Our algorithm shows a little better result than Algorithm TraverStringRef in some challenging states. Although it lags behind the qPMS9 in all cases. The comparisons are shown in table 3.3, table 3.4, fig. 3.4, fig. 3.5.

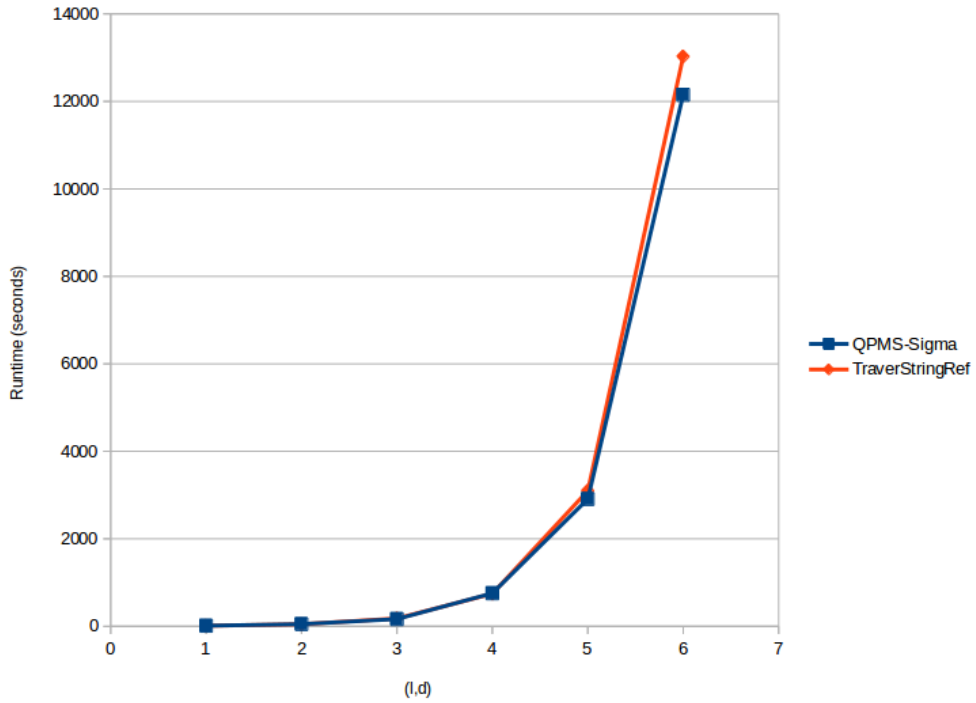


Figure 3.4: Run Time Comparison between qPMS-Sigma vs TraverStringRef in the challenging states for q=20

Table 3.4: Computational Results for DNA Sequences for $q = 10$

Algorithm	**(13,4)**	**(15,5)**	**(17,6)**	**(19,7)**	**(21,8)**	**(23,9)**
TraverStringRef	7	36	160	760.6	3643.3	17232.4
TraverStringRef	5.5	32	166.1	779.9	3700.6	17922.2
qPMS7	31	152	850.7		4865.0	29358
qPMSPrune	12.8	126.7	1116.5	10540.8		
PMS9						

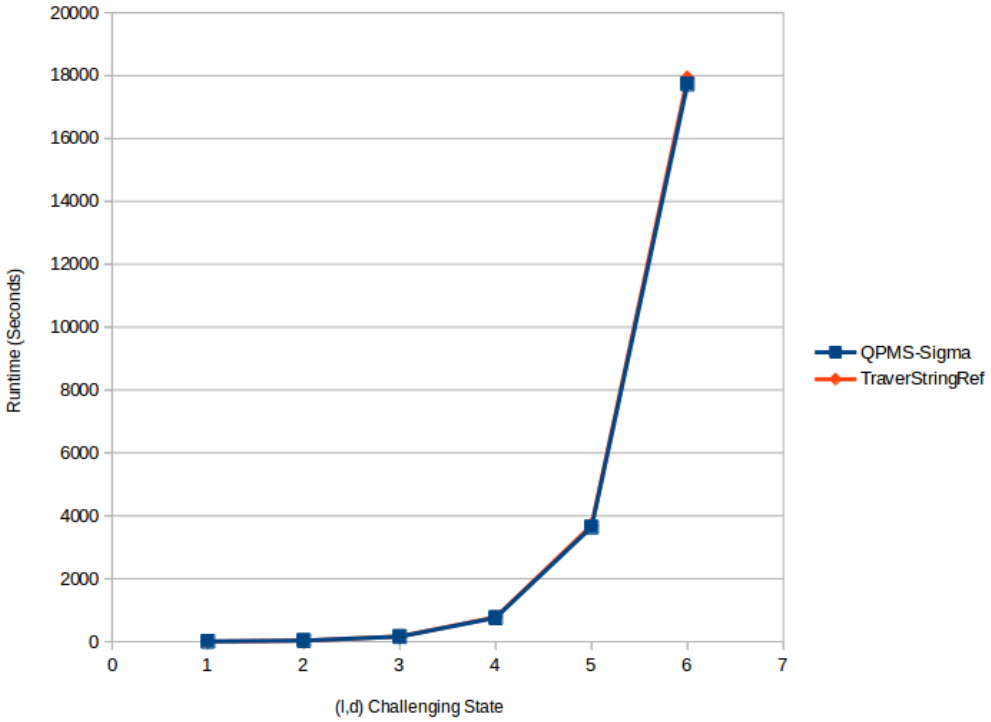


Figure 3.5: Run Time Comparison between qPMS-Sigma vs TraverStringRef in the challenging states for q=10

Chapter 4

Results & Discussion

4.1 A Section

Some text.

4.1.1 This is a Subsection

And some more.

This is a Subsubsection

Yet some more.

4.2 And Another Section

Here are some dummy texts.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras et ultricies massa. Nulla a sapien lobortis, dignissim nibh in, aliquet mauris. Integer at dictum metus. Quisque in tortor congue ipsum ultricies tristique. Maecenas ut tortor dapibus, sagittis enim at, tincidunt massa. Ut sollicitudin sagittis ipsum, ac tincidunt quam gravida ac. Nullam quis faucibus purus. Aliquam vel pretium turpis. Aliquam a quam non ex interdum sagittis id vitae quam. Nullam sodales ligula malesuada maximus consequat. Proin a justo eget lacus vulputate maximus luctus vitae enim. Aliquam libero turpis, pharetra a

tincidunt ac, pulvinar sit amet urna. Pellentesque eget rutrum diam, in faucibus sapien. Aenean sit amet est felis. Aliquam dolor eros, porttitor quis volutpat eget, posuere a ligula. Proin id velit ac lorem finibus pellentesque.

Maecenas vitae interdum mi. Aenean commodo nisl massa, at pharetra libero cursus vitae. In hac habitasse platea dictumst. Suspendisse iaculis euismod dui, et cursus diam. Nullam euismod, est ut dapibus condimentum, lorem eros suscipit risus, sit amet hendrerit justo tortor nec lorem. Morbi et mi eget erat bibendum porta. Ut tristique ultricies commodo. Nullam iaculis ligula sed lacinia ornare.

Sed ultricies cursus nisi at vestibulum. Aenean laoreet viverra efficitur. Ut eget sapien lorem. Mauris malesuada, augue in pulvinar consectetur, ex tortor tristique ligula, sit amet faucibus metus lectus interdum nisl. Nam eget turpis vitae ligula pulvinar bibendum a ut ipsum. Mauris fringilla lacinia malesuada. Fusce id orci velit. Donec tristique rhoncus urna, a hendrerit arcu vehicula imperdiet. Integer tristique erat at gravida condimentum. Sed ornare cursus quam, eget tincidunt enim bibendum sed. Aliquam elementum ligula scelerisque leo sagittis, quis convallis elit dictum. Donec sit amet orci aliquam, ultricies sapien nec, gravida nisi. Etiam et pulvinar diam, et pellentesque arcu. Nulla interdum metus sed aliquet consequat.

Proin in mi id nulla interdum aliquet ac quis arcu. Duis blandit sapien commodo turpis hendrerit pharetra. Phasellus sit amet justo orci. Proin mattis nisl dictum viverra fringilla. Interdum et malesuada fames ac ante ipsum primis in faucibus. Curabitur facilisis euismod augue vestibulum tincidunt. Nullam nulla quam, volutpat vitae efficitur eget, porta sit amet nunc. Phasellus pharetra est eget urna ornare volutpat. Aenean ultrices, libero eget porttitor fringilla, purus tortor accumsan neque, sit amet viverra felis tortor eget justo. Nunc id metus a purus tempus euismod condimentum non lacus. Nam vitae diam aliquam, facilisis diam quis, pharetra nunc. Nulla eget vestibulum tellus, ut cursus tellus. Vestibulum euismod pellentesque sodales.

Maecenas at mi interdum, faucibus lorem sed, hendrerit nisi. In vitae augue consequat diam commodo porta sit amet eu purus. Mauris mattis condimentum feugiat. Nulla commodo molestie risus vitae maximus. Proin hendrerit neque malesuada urna laoreet convallis. Etiam a diam pulvinar, auctor sem ac, hendrerit risus. Ut urna urna, venenatis ac tellus non, scelerisque tristique ligula. Vestibulum sollicitudin vel leo malesuada accumsan. Donec sit amet erat diam. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Vivamus odio dui, scelerisque et lorem egestas, posuere ullamcorper nunc. Integer varius nunc nec velit tincidunt commodo. Mauris rhoncus ultrices sapien non suscipit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras et ultricies massa. Nulla a sapien lobortis, dignissim nibh in, aliquet mauris. Integer at dictum metus. Quisque in tortor congue ipsum ultricies tristique. Maecenas ut tortor dapibus, sagittis enim at, tincidunt massa. Ut sollicitudin sagittis ipsum, ac tincidunt quam gravida ac. Nullam quis faucibus purus. Aliquam vel pretium turpis. Aliquam a quam non ex interdum sagittis id vitae quam. Nullam sodales ligula malesuada maximus consequat. Proin a justo eget lacus vulputate maximus luctus vitae enim. Aliquam libero turpis, pharetra a tincidunt ac, pulvinar sit amet urna. Pellentesque eget rutrum diam, in faucibus sapien. Aenean sit amet est felis. Aliquam dolor eros, porttitor quis volutpat eget, posuere a ligula. Proin id velit ac lorem finibus pellentesque.

Maecenas vitae interdum mi. Aenean commodo nisl massa, at pharetra libero cursus vitae. In hac habitasse platea dictumst. Suspendisse iaculis euismod dui, et cursus diam. Nullam euismod, est ut dapibus condimentum, lorem eros suscipit risus, sit amet hendrerit justo tortor nec lorem. Morbi et mi eget erat bibendum porta. Ut tristique ultricies commodo. Nullam iaculis ligula sed lacinia ornare.

Sed ultricies cursus nisi at vestibulum. Aenean laoreet viverra efficitur. Ut eget sapien lorem. Mauris malesuada, augue in pulvinar consectetur, ex tortor tristique ligula, sit amet faucibus metus lectus interdum nisl. Nam eget turpis vitae ligula pulvinar bibendum a ut ipsum. Mauris fringilla lacinia malesuada. Fusce id orci velit. Donec tristique rhoncus urna, a hendrerit arcu vehicula imperdiet. Integer tristique erat at gravida condimentum. Sed ornare cursus quam, eget tincidunt enim bibendum sed. Aliquam elementum ligula scelerisque leo sagittis, quis convallis elit dictum. Donec sit amet orci aliquam, ultricies sapien nec, gravida nisi. Etiam et pulvinar diam, et pellentesque arcu. Nulla interdum metus sed aliquet consequat.

Proin in mi id nulla interdum aliquet ac quis arcu. Duis blandit sapien commodo turpis hendrerit pharetra. Phasellus sit amet justo orci. Proin mattis nisl dictum viverra fringilla. Interdum et malesuada fames ac ante ipsum primis in faucibus. Curabitur facilisis euismod augue vestibulum tincidunt. Nullam nulla quam, volutpat vitae efficitur eget, porta sit amet nunc. Phasellus pharetra est eget urna ornare volutpat. Aenean ultrices, libero eget porttitor fringilla, purus tortor accumsan neque, sit amet viverra felis tortor eget justo. Nunc id metus a purus tempus euismod condimentum non lacus. Nam vitae diam aliquam, facilisis diam quis, pharetra nunc. Nulla eget vestibulum tellus, ut cursus tellus. Vestibulum euismod pellentesque sodales.

Maecenas at mi interdum, faucibus lorem sed, hendrerit nisi. In vitae augue consequat diam commodo porta sit amet eu purus. Mauris mattis condimentum feugiat. Nulla

commodo molestie risus vitae maximus. Proin hendrerit neque malesuada urna laoreet convallis. Etiam a diam pulvinar, auctor sem ac, hendrerit risus. Ut urna urna, venenatis ac tellus non, scelerisque tristique ligula. Vestibulum sollicitudin vel leo malesuada accumsan. Donec sit amet erat diam. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Vivamus odio dui, scelerisque et lorem egestas, posuere ullamcorper nunc. Integer varius nunc nec velit tincidunt commodo. Mauris rhoncus ultrices sapien non suscipit.

Chapter 5

Conclusion

5.1 Summary of Our Work

The main task of our thesis is to develop an effective algorithm for the Planted Motif Search problem. Our proposed algorithm qPMS-Sigma tries to find the (l, d)-motifs for n given sequences. It is an exact version of the PMS algorithm. qPMS-Sigma is based on the previous PMS algorithms qPMSPrune, qPMS7, TraverStringRef and PMS8. In our proposed algorithm we introduce clever techniques to compress the input sequences and thus space complexity is improved. We also include a faster comparison technique of the l-mers by adding bitwise comparison techniques. As we know the PMS is a NP-Hard problem, it is exponential in terms of time. So, parallel implementation techniques are proposed at the end of our work.

5.2 Future Prospects of Our Work

Though in our thesis we have given some ideas about parallel implementation of our algorithm, it is not implemented and tested on multiprocessor system. All comparison of our algorithm with the stated algorithms are done in single processor system. To understand the real speedup and slackness we need to experiment in a system involving a network of nodes having multiple processors. Our work is the extension of PMS8 codes which involves openMPI libraries. In future we want to continue our work using OpenMPI project which is a open source Message Passing Interface. Apart from the parallelism, more advanced pruning conditions can be used to reduce the search space. Different randomized approaches can also be included for improving the run-

time, though it will make the exact version of our algorithm approximate.

References

[1] I. Rida, A. Noor, and A. Somaya, "Robust gait recognition: a comprehensive survey," *IET Biometrics*, vol. 8, pp. 14 – 28, January 2019.

[2] J. Han and B. Bhanu, "Individual recognition using gait energy image," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 28, pp. 316–322, February 2006.

[3] K. Bashir, T. Xiang, and S. Gong, "Gait recognition using gait entropy image," in *3rd Int. Conf. on Imaging for Crime Detection and Prevention*, London, UK, 2009.

[4] T. H. W. Lam, K. H. Cheung, and J. N. K. Liu, "Gait flow image: A silhouette-based gait representation for human identification," *Pattern Recognition*, vol. 44, pp. 973 – 987, April 2011.

[5] G. Ariyanto and M. S. Nixon, "Model-based 3d gait biometrics," in *International Joint Conference on Biometrics*, pp. 1–7, Washington DC, USA, 2011.

[6] F. Tafazzoli and R. Safabakhsh, "Model-based human gait recognition using leg and arm movements," *Engineering Appl. of Art. Intell.*, vol. 23, pp. 1237 – 1246, April 2010.

[7] Y. Feng, Y. Li, and J. Luo, "Learning effective gait features using lstm," in *23rd International Conference on Pattern Recognition*, pp. 325–330, Cancun, Mexico, 2016.

[8] Z. Wu, H. Yongzhen, and W. a. Liang, "A comprehensive study on cross-view gait based human identification with deep cnns," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 39, pp. 209–226, February 2017.

[9] K. Shiraga, Y. Makihara, and D. a. Muramatsu, "Geinet: View-invariant gait recognition using a convolutional neural network," in *International Conference on Biometrics (ICB)*, Halmstad, Sweden, 2016.

[10] T. Wolf, M. Babaee, and G. Rigoll, "Multi-view gait recognition using 3d convolutional neural networks," in *IEEE International Conference on Image Processing*, pp. 4165–4169, Phoenix, AZ, USA, 2016.

[11] F. M. Castro, M. J. Marin-Jimenez, and N. a. Guil, "Automatic learning of gait signatures for people identification," in *International Work-Conference on Artificial Neural Networks*, pp. 257–270, Springer, Cham, 2017.

[12] 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 (TCBB)*, vol. 4, no. 4, pp. 544–552, 2007.

[13] 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, 2014.

[14] P. A. Pevzner, S.-H. Sze, *et al.*, "Combinatorial approaches to finding subtle signals in dna sequences.," in *ISMB*, vol. 8, pp. 269–278, 2000.

Index

Compression, 19

Parallel, 20
PMS8, 17

qPMS7, 12
qPMSPrune, 9

space complexity, 20

TraverStringRef, 15, 20

Appendix A

Codes

A.1 Compressed String

Here is a sample C++ structure for compressing a 1000 character long DNA string

```
1 //letterToInt[i] means a unique id for each character of
     the alphabet;
2 //Here,
3 //letterToInt['A']=0;
4 //letterToInt['B']=1;
5 //letterToInt['C']=2;
6 //letterToInt['D']=3;
7
8 struct CompressedlMer
9 {
10     unsigned int **cs;//[2][32];
11     int l;
12     CompressedlMer(char *string, int start, int l)    {//
          constructor
13         int group = 0;
14         int intPerGroup = ceil(l/sizeof(unsigned int));
15         int temp = l;
16         while(temp)    {
17             group++;
18             temp = temp/2;
```

```
19          }
20          cs = new unsigned int*[group];
21          for(int i = 0; i < group; i++)
22              cs[i] = new unsigned int[intPerGroup];
23
24          for(int i = 0; i < l; i++)    {
25              int j = letterToInt[string[start+i]];
26              int positionMask = 1<<(i%sizeof(unsigned int)
                    );
27              int intNo = i/sizeof(unsigned int);
28              int grp = 0;
29              while(j>0)    {
30                  if(j&1) cs[grp][intNo] = cs[grp][intNo] |
                        positionMask;
31                  j = j/2;
32                  grp++;
33              }
34          }
35      }
36
37      ~CompressedlMer()    {//destructor
38          for(int i = 0,j=1; j < l; i++,j*=2)    {
39              delete [] cs[i];
40          }
41          delete [] cs;
42      }
43  };
44
45  int hammingDist(const CompressedlMer &sa, const
        CompressedlMer &sb)    {
46      int intPerGroup = ceil(sa.l/sizeof(unsigned int));
47      int hamDist = 0;
48      for(int i = 0; i < intPerGroup; i++)    {
49          unsigned int flag = ~(0);    //all bits are set
                to 1
50          for(int group = 0, j = 1; j < sa.l; group++, j
                *=2)    {
```

```
51              flag = flag | (sa.cs[group][i]^sb.cs[group][i
                   ]);
52          }
53          hamDist = hamDist + __builtin_popcount(flag);
54      }
55      return hamDist;
56  }
```

Generated using Undegraduate Thesis LaTeX Template, Version 1.2. Institute of Information and Communication Technology, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

This thesis was generated on Sunday 12th January, 2020 at 8:31pm.