University of Windsor

Scholarship at UWindsor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

10-30-2020

Finding Binding Sites in ChIP-Seq Data via a Linear-time Multi-level Thresholding Algorithm

Musab Mushtaque Naik
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Naik, Musab Mushtaque, "Finding Binding Sites in ChIP-Seq Data via a Linear-time Multi-level Thresholding Algorithm" (2020). *Electronic Theses and Dissertations*. 8463.
https://scholar.uwindsor.ca/etd/8463

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Finding Binding Sites in ChIP-Seq Data via a Linear-time Multi-level Thresholding Algorithm

By

Musab Naik

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2020

©2020 Musab Naik

Finding Binding Sites in ChIP-Seq Data via a Linear-time Multi-level Thresholding Algorithm

by

Musab Naik

APPROVED BY:

H. Wu
Department of Electrical and Computer Engineering

A. Biniaz
School of Computer Science

L. Rueda, Advisor
School of Computer Science

September 8, 2020

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

Chapter 2 of the thesis was co-authored with Luis Rueda. L. Rueda contributed with initial thoughts about this research area and the main ideas, and assisted in elaborating on the new novel approach implemented in this work. Both authors contributed in finalizing the idea and follow-up discussions. M. Naik implemented the method, data pre-processing, experimental design, and the data analysis. M. Naik wrote the contents of the chapter. L. Rueda assisted in writing and reviewing the content of the chapter.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis. I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes an original paper that has been previously submitted for publication in the following IEEE journal:

Thesis chapter	**Publication title**	**Publication Status**
Chapter 2	Musab Naik and Luis Rueda, Finding Binding Sites in ChIP-Seq Data via a Linear-time Multi-level Thresholding Algorithm, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2020	Submitted

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution. I understand that my thesis may be made electronically available to the public.

ABSTRACT

Chromatin immunoprecipitation (ChIP–Seq) has emerged as a superior alternative to microarray technology as it provides higher resolution, less noise, greater coverage and wider dynamic range. While ChIP-Seq enables probing of DNA-protein interaction over the entire genome, it requires the use of sophisticated tools to recognize hidden patterns and extract meaningful data. Over the years, various attempts have resulted in several algorithms making use of different heuristics to accurately determine individual peaks corresponding to unique DNA-protein binding sites. However, finding all the binding sites with high accuracy in a reasonable time is still a challenge.

In this work, we propose the use of Multi-level thresholding algorithm, which we call LinMLTBS, used to identify the enriched regions on ChIP-Seq data. Although various suboptimal heuristics have been proposed for multi-level thresholding, we emphasize on the use of an algorithm capable of obtaining an optimal solution, while maintaining linear-time complexity. Testing various algorithm on various ENCODE project datasets shows that our approach attains higher accuracy relative to previously proposed peak finders while retaining a reasonable processing speed.

DEDICATION

Dedicated to my parents for their endless love, support and encouragement.

AKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Luis Rueda, for his exceptional guidance, patience, and constant support. This thesis would not have been possible without this ideas and suggestions. I have greatly benefited from his immense knowledge and it has been such an honour for me to work under his supervision.

I would also like to thank my external reader Dr. Huapeng Wu and and internal reader Dr. Ahmad Biniaz for being a part of this thesis. Their feedback and comments have helped in further improving of this thesis.

Finally, I would like to thank all friends and family who those who have been a source of constant morale support.

TABLE OF CONTENTS

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION III

ABSTRACT V

DEDICATION VI

AKNOWLEDGEMENTS VII

LIST OF TABLES X

LIST OF FIGURES XI

LIST OF ABBREVIATIONS XII

1 Introduction 1
- 1.1 Introduction to Molecular Biology . . . 1
- 1.2 Gene Expression . . . 3
- 1.3 Next Generation Sequencing & ChIP-Seq . . . 3
- 1.4 Problem Statement . . . 6
- 1.5 Motivation . . . 6
- 1.6 Multi-level Thresholding . . . 7
  - 1.6.1 The Thresholding Problem . . . 7
  - 1.6.2 Thresholding Criteria . . . 8
  - 1.6.3 Polynomial-time Algorithm . . . 9
- 1.7 More Efficient Algorithms . . . 11
  - 1.7.1 Divide-and-Conquer . . . 14
  - 1.7.2 SMAWK . . . 14
- 1.8 Proposed Method . . . 16
  - 1.8.1 Contributions . . . 18
- References . . . 18

2 Binding-sites in ChIP-Seq via LinMLTBS 23
- 2.1 Introduction . . . 23
- 2.2 Methods . . . 27
  - 2.2.1 Multi-level Thresholding . . . 27
  - 2.2.2 SMAWK Algorithm using Otsu's Criterion . . . 31
  - 2.2.3 Cluster Validity Indices . . . 33
  - 2.2.4 Concurrent Multi-window Approach . . . 34
- 2.3 Results and Discussion . . . 35
  - 2.3.1 Datasets . . . 35
  - 2.3.2 Testing and Biological Validation . . . 37
  - 2.3.3 Comparison with Other Methods . . . 38

References 40

3 Conclusion and Future Work 47
3.1 Conclusion 47
3.1.1 Contributions 47
3.2 Future Work 48
References 48

VITA AUCTORIS 50

LIST OF TABLES

1.3.1 Comparison of ChIP-Seq with other technologies. 4
1.6.1 An example table C, D for an histogram with n bins and $k = 4$ thresholds. 12

2.3.1 Accession no. & cell line for ChIP-Seq experiments targeting the H3K27ac histone modification. 37
2.3.2 Number of peaks detected by LinMLBTS and MACS2. 40
2.3.3 H3K27ac - (LinMLTBS with 1.5 times the number of peaks) 41

LIST OF FIGURES

1.1.1 Structure of DNA. 1

1.1.2 Central Dogma of modern molecular biology. 2

1.3.1 Workflow of ChIP-Seq data analysis. 5

1.7.1 An example of the REDUCE function in action. 15

1.8.1 A block diagram of the proposed method. 17

2.1.1 Histogram creation by the alignment of immunoprecipitated fragments. 24

2.2.1 An example of a trellis structure corresponding to the steps to identify the optimal thresholds. 30

2.3.1 Log-log graph of window size vs CPU time for quadratic and linear time algorithms. 39

LIST OF ABBREVIATIONS

DNA	Deoxyribonucleic acid
RNA	Ribonucleic acid
mRNA	Messenger RNA
tRNA	Transfer RNA
rRNA	Ribosomal RNA
NGS	Next Generation Sequencing
ChIP	Chromatin immunopr ecipitation
ChIP-Seq	Chromatin immunoprecipitation followed by high throughput sequencing
ENCODE	ENCyclopedia Of DNA Elements
BC	Between Class
DP	Dynamic Programming
STAR	Spliced Transcripts Alignment to a Reference
DIVCONQ	Divide-and-Conquer
MACS	Model-based analysis of ChIP-Seq
SICER	Spatial Clustering for Identification of ChIP-Enriched Regions
MOSAiCS	MOdel-based one and two Sample Analysis and Inference for ChIP-Seq Data

ZINBA	Zero-Inflated Negative Binomial Algorithm
BCP	Bayesian Change Point
CMT	Constrained Multi-level Thresholding
LinMLTBS	Linear-time Multi-level Threshold approach Binding Site detection

CHAPTER 1

Introduction

1.1 Introduction to Molecular Biology

Modern molecular biology is built upon the understanding of the structure and function of DNA (deoxyribonucleic acid) and RNA (ribonucleic acid), and the enzymes and proteins that interact with these structures. The structure of DNA, shown in Figure 1.1.1 [31], is the foundation upon which the Central Dogma of modern molecular biology is based. It contains the information necessary to code for the RNA and proteins used by a cell. RNA has a similar structure to DNA, but consists of only one strand and does not form a helix structure like DNA. It also has nucleotides which consist of a sugar, phosphate, and a base. The sugar, however, is a ribose instead of deoxyribose and hence the name RNA. Also, DNA base Thymine (T) is replaced with Uracil (U) in RNA. The structure of DNA is depicted in Figure 1.1.1 [31]

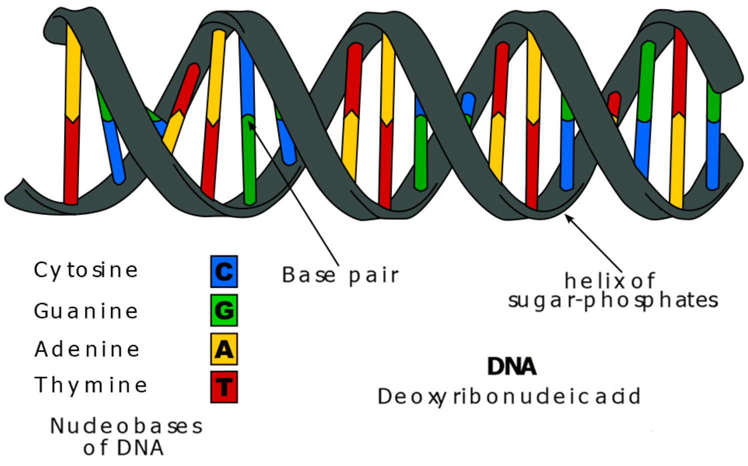

Fig. 1.1.1: Structure of DNA.

With the help of specific proteins and enzymes, a copy of the DNA is made through a process known as DNA replication. This provides the genetic information necessary for the future generation. Through the process of transcription, DNA is transcribed into RNA. The messenger RNA (mRNA) molecules contain the code for necessary for building of proteins. The process of creation of proteins by reading the RNA sequence is known as translation. Combining all of these roles together Figure 1.1.2 [31] depicts the Central Dogma of modern molecular biology [15, 2].

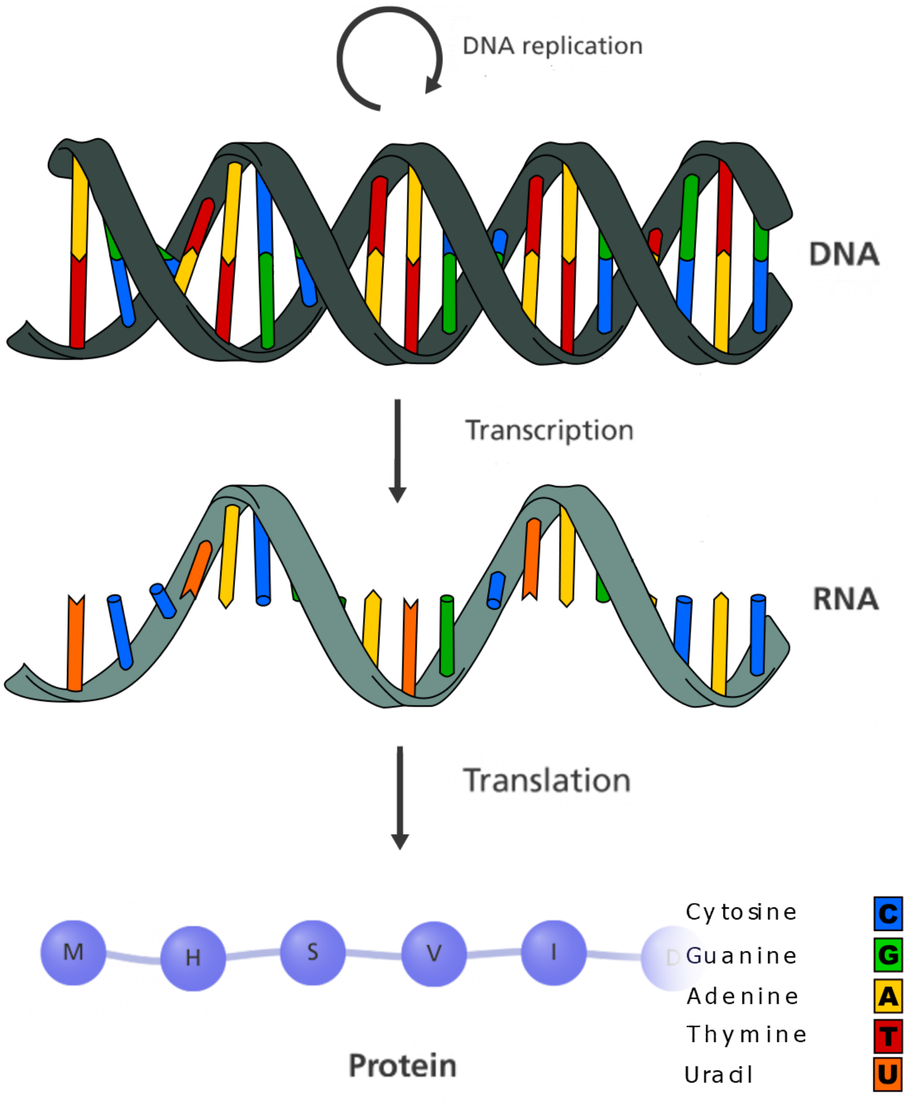

Fig. 1.1.2: Central Dogma of modern molecular biology.

1.2 Gene Expression

A gene, physically, is a sequence of DNA that codes for an RNA (mRNA, tRNA, or rRNA) and the mRNA codes for a protein. Proteins dictate cell functions. Therefore, the thousands of genes expressed in a particular cell determine what that cell can do. It is worth noting that protein production starts at transcription (DNA to RNA) and continues with translation (RNA to protein). Thus, control of these processes plays a critical role in determining which proteins are present in a cell and in which amounts. In addition, the way in which a cell processes its RNA transcripts and newly made proteins also greatly influences protein levels.

To control the expression of genes, a larger number of regulatory proteins are involved. These proteins interact with specify binding sites on the DNA sequence and depending on the nature of the protein, they may inhibit or assist the expression of the gene. The regulatory binding sites may be located quite far from protein binding sites. Thus, to understand many biological processes and disease states, study of the DNA-Protein interactions is a very crucial step. Studies of gene expression employ a wide variety of molecular biology techniques and experimental methods. Gene expression analysis studies can be broadly divided into four areas: RNA expression, promoter analysis, protein expression, and post-translational modification.

1.3 Next Generation Sequencing & ChIP-Seq

Over the past few decades, DNA sequencing methodology has evolved rapidly. Sanger dideoxy synthesis [30] and Maxam-Gilbert chemical cleavage [23] are founding methods in DNA sequencing. Latest Sanger sequencing instruments, making use of capillary-based automated electrophoresis, can analyze 8–96 sequencing reactions simultaneously. On the other hand, modern-day Next Generation Sequencing (NGS), introduced in the past decade [22], allow parallel sequencing reactions on a much larger scale. These systems have the capability of analyzing millions or even billions of sequencing reactions in parallel at the same time. Out of all NGS technologies, ChIP-Seq is by far the most popular.

Chromatin immunoprecipitation followed by high-throughput sequencing (ChIP-Seq) is a gene expression analysis tool that can be categorized under the promoter analysis method [18]. It works on the principle of chromatin immunoprecipitation (ChIP), where the protein of interest is targeted using a specific antibody. This antibody is special in the sense that it attaches only to the protein of interest. In this method, the DNA sequences are first treated with antibody specific to the protein of interest. After this the DNA is fragmented, usually by sonication. Then a precipitation agent is introduced. This results in the precipitation of the DNA fragments to which the protein of interest binds. All other fragments, not containing the protein of interest, are filtered out. These fragments undergo a final step of reverse cross-linking before reading of the sequences. The basic concept of ChIP, first introduced in 1980's [17], has been around for a long time and has been used by many other genomic analysis method, such as ChIP-Pet [16], ChIP-chip [11], and ChIP-exo [28], just to name a few. Table 1.3.1 shows a comparison between previous ChIP technologies and ChIP-Seq [27].

Table 1.3.1: Comparison of ChIP-Seq with other technologies.

Features	**ChIP-Seq**	**Other ChIP technologies**
Resolution	1 bp	30-100 bp
Coverage	Whole genome	limited by sequence on array
Cost	\$500-1000 per lane	\$400-800 per array
Required amount of ChIP DNA	Low (few nano grams)	High (few micro grams)
Dynamic range	Not Limited	Lower detection limit
Multiplexing	Possible	Not Possible

Out of all the existing methods for ChIP data analysis, ChIP-Seq has proven to be to one of the best [27]. It has several advantages such as high resolution, wide coverage of the genome, high signal-to-noise ratio, and large number of localized peaks, among many others. Through ChIP-Seq, projects such as ENCODE (ENCyclopedia Of DNA Elements) [13], a public research consortium aimed at identifying all functional elements in the human genome, have been made possible. ChIP-Seq can take advantage of the next generation sequencing machines such as Illumina NovaSeq 6000 [25], to accurately read millions of short reads generated by the ChIP process. Once the sequences are read, they are then mapped to the original genome. There are many tools available for this task such as Tophat2 [32] and STAR [14]. After the reads are aligned over the reference genome, the data can now be analyzed through process known as *peak calling*. Peak calling can help determine the locations on the DNA sequence at which the protein of interest interacts with. Once these regions of interest are detected, further analysis can be performed such as finding binding sites, visualization, motif discovery, and others.

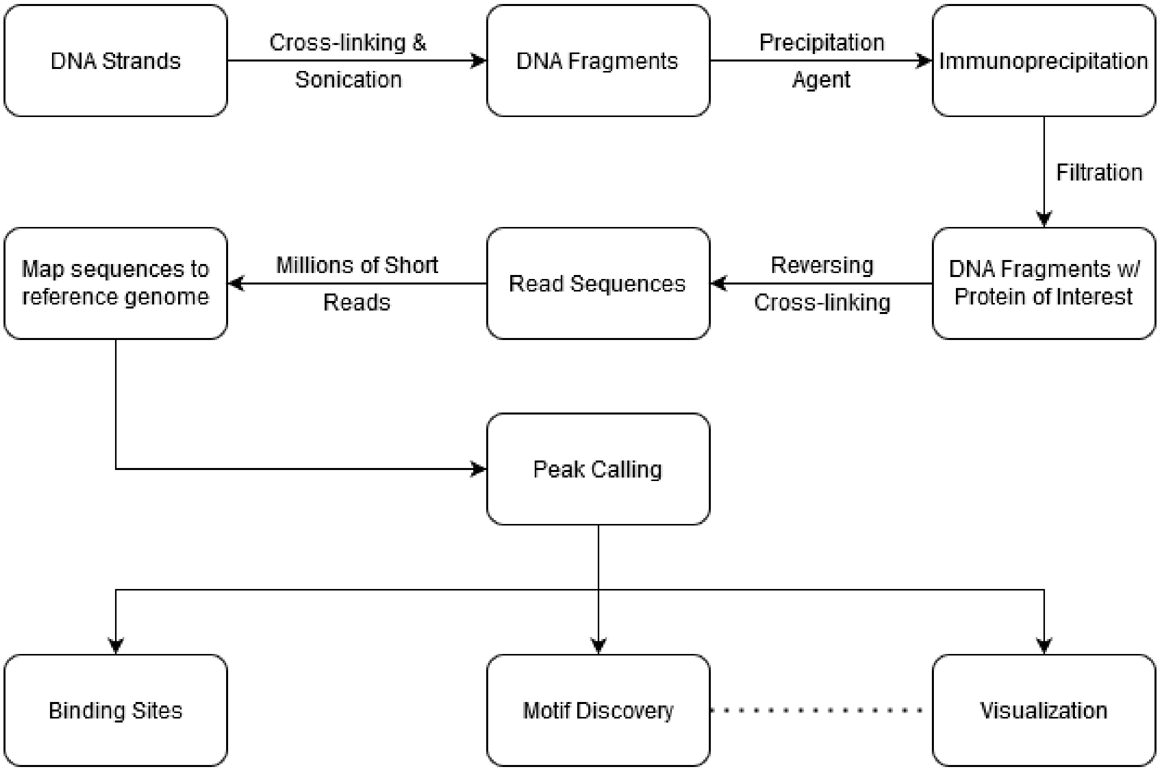

Fig. 1.3.1: Workflow of ChIP-Seq data analysis.

1.4 Problem Statement

ChIP-Seq experiments generate millions of short read, which are then aligned over the reference genome. As the human genome is around 3 billion bp in length, putative identification of the protein binding sites is truly a bioinformatic challenge that requires considerable computational resources [4]. To solve this problem, a computational technique known as "peak calling" is used. The basic idea behind peak calling is to read the entire genome with the fragments aligned over it and then identify regions over which a large number of reads overlap. The overlapping of the reads can be visualized as peaks, as depicted in Figure 2.1.1. These peaks indicate that a large number of fragments align over a particular region on the genome and are an evidence of the presence of a binding site. Over the years, numerous peak finding algorithms have been developed in order to satisfy the ever growing popularity of ChIP-Seq. But yet, identification of binding sites (peaks) of varying size with accuracy and in respectable time is still a challenge.

The problem can be informally stated as follows. Design and implement a peak calling algorithm that can correctly identify binding sites (peaks) of varying size, ideally, as quickly as possible. Given that existing methods either trade accuracy for speed or speed for accuracy, the schemes used to solve this problem should be maintain a balance between both.

1.5 Motivation

As previously stated, studying the interactions of proteins with DNA to regulate gene expression is a crucial step towards the understanding of many biological processes. This correlation between proteins and gene expression can help in identification of many diseases and even assist in finding a cure for them. Next generation sequencing, more specifically ChIP-Seq, is an essential tool to achieve this goal as it offers high resolution, less noise, and great coverage of the genome in comparison with its counterparts. As ChIP-Seq has been growing in popularity, there is a need to develop a peak calling algorithm, one that can maintain a balance between speed and accuracy. This tool also needs to be easy to use and understand. Existing methods are very complicated and require the user to "tweak" the parameters of the algorithm for optimal results.

In this work, we propose the use of an ultrafast multi-level thresholding algorithm to find peaks in a histogram, which in turn would find unique binding sites. To achieve the previously stated goal of speed and accuracy, it is very important that the multi-level thresholding algorithm used finds nothing but optimal thresholds, in linear-time complexity. We show that when a linear-time optimal multi-level thresholding algorithm is coupled with a model that can find the optimal number of peaks while taking advantage of modern-day multi-core CPU architecture, can outperform the existing methods for peak calling.

1.6 Multi-level Thresholding

Multi-level thresholding is one of the most widely-used techniques that has many applications in signal and image processing, including segmentation, classification and object discrimination. Given the frequencies or probabilities for each bin of a histogram, the objective of multi-level thresholding is to divide the histogram into a number of groups (or classes) of continuous bins such that a given criteria is optimized. This division of the histograms is done by placing thresholds, namely positions, which determine whether the criteria is optimized or not.

1.6.1 The Thresholding Problem

To understand the thresholding problem, let us consider a histogram H, an ordered set $\{1, 2, ..., n-1, n\}$ and $F = \{f_1, f_2, ..., f_n\}$ be the frequencies of the histogram. Here, n is the number if bins in the histogram and the i^{th} value corresponds to the i^{th} bin having a probability, p_i associated with it. The probabilities of H are give by the set $P = \{p_1, p_2, ..., p_n\}$, where $p_i \geq 0, \sum_{i=1}^{n} p_i = 1$, and $p_i = f_i/N$ with $N = \sum_{i=1}^{n} f_i$. Also consider, a threshold set T defined as an ordered set $T = \{t_0, t_1, ..., t_k, t_{k+1}\}$, where $0 = t_0 < t_1 < ... < t_k < t_{k+1} = n$ and $t_i \in \{0\} \cup H$.

Thresholding Problem: The problem of multi-level thresholding consists of finding a threshold set T, in such a way that a function $f : H^k \times [0,1]^n \to \mathbb{R}^+$ is maximized/minimized. Using this threshold set, H is divided into $k+1$ classes: $\zeta_1 = \{1, 2, ..., t_1\}, \zeta_2 = \{t_1+1, t_1+2, ..., t_2\}, ..., \zeta_k = \{t_{k-1}+1, t_{k-1}+2, ..., t_k\}, \zeta_{k+1} = \{t_k+1, t_k+2, ..., n\}$.

1.6.2 Thresholding Criteria

Various parametric and non-parametric thresholding criteria have been proposed over the years, the three most important criteria being as follows [29]:

- Between class variance [26]:

$$\Psi_{\mathrm{BC}}(T) = \sum_{j=1}^{k+1} \omega_j \mu_j^2, \tag{1}$$

  where $\omega_j = \sum_{i=t_{j-1}+1}^{t_j} p_i, \mu_j = \frac{1}{\omega_j} \sum_{i=t_{j-1}+1}^{t_j} i p_i$.

- Entropy-based [19]:

$$\Psi_{\mathrm{H}}(T) = \sum_{j=1}^{k+1} H_j, \tag{2}$$

  where $H_j = -\sum_{i=t_{j-1}+1}^{t_j} \frac{p_i}{\omega_j} \log \frac{p_i}{\omega_j}$.

- Minimum error [20]:

$$\Psi_{\mathrm{ME}}(T) = 1 + 2 \sum_{j=1}^{k+1} \omega_j (\log \sigma_j + \log \omega_j), \tag{3}$$

  where $\sigma_j^2 = \sum_{i=t_{j-1}+1}^{t_j} \frac{p_i (i-\mu_j)^2}{\omega_j}$.

The selection of the thresholds is based on optimization of the chosen criterion. Optimal thresholds are the ones that either maximize or minimize the selected criterion. The most straight forward method to ensure that the criteria is optimized is to carry out an exhaustive search of all subsets of possible threshold positions. This brute-force approach is usually good for only a few thresholds, say, 4 or 5. However, as the number of thresholds grow, the computational complexity grows exponentially with the number of thresholds, stated as $O(n^k)$, where n is the number of bins in the histogram and k is the number of thresholds.

1.6.3 Polynomial-time Algorithm

A dynamic programming algorithm for optimal multi-level thresholding that runs in quadratic polynomial time, has been proposed earlier and extended to large, sparse histograms in a subsequent work [29]. Here, the criterion is represented by a function Φ, which can be decomposed as a sum of terms given by a second function ψ. The function is defined as follows:

$$\Psi(T_{0,m}) = \Psi(\{t_0, t_1, ..., t_m\}) \triangleq \sum_{j=1}^{m} \psi_{t_{j-1}+1,t_j}, \tag{4}$$

where $1 \leq m \leq k+1$ and the function $\psi_{l,r}$, where $l \leq r$, is a real, positive function of $p_l, p_{l+1}, ..., p_r$, $\psi_{l,r} : H^2 \times [0,1]^{l-r+1} \to \mathbb{R}^+ \cup \{0\}$

Which means that if the optimal solution, $\Psi^*(T_{0,j-1})$, for $T_{0,j-1} = \{t_0, t_1, ..., t_{j-1}\}$, is known. Then, the optimal solution, $\Psi^*(T_{0,j})$, for $T_{0,j}$, is computed as follows:

$$\Psi^*(T_{0,j}) = \begin{cases} 0 & \text{if } j = 0 \\ \max_{\min\{t_{j-1}\} \geq t_{j-1} \geq \max\{t_{j-1}\}} \Psi^*(T_{0,j-1}) + \psi_{t_{j-1}+1,t_j} & \text{if } 1 \leq j \leq k+1 \end{cases} \tag{5}$$

where

$$\min\{t_j\} = \begin{cases} j & \text{if } 0 \leq j \leq k \\ n & \text{if } j = k+1, \end{cases} \tag{6}$$

Algorithm 1.6.1 Polynomial-time Multi-level Thresholding

```
 Input: Probabilities, $P = \{p_1, p_2, ..., p_n\}$, Number of thresholds $k$
 Output: A threshold set, $T = \{t_0, t_1, p_2, ..., t_k, t_{k+1}\}$
minTj, maxTj ← findThresholdsRanges($k$)
Fill columns 1 ro $k + 1$
$C(0, 0) \leftarrow 0; D(0, 0) \leftarrow 0$
for $j \leftarrow 1$ to $k + 1$ do
  for $t_j \leftarrow minTj(j)$ to $maxTj(j)$ do
    $C(t_j, j) \leftarrow 0$; psi← $\psi_{j,t_j}$
    for $i \leftarrow minTj(j - 1)$ to $min\{maxTj(j - 1), t_j - 1\}$ do
      if $C(i, j - 1)$+psi $C(t_j, j)$ then
        $C(t_j, j) \leftarrow C(i, j - 1) + psi$
        $D(t_j, j) \leftarrow i$
      end if
      psi ← Compute $\psi_{i+2,t_j}$ from psi, $i + 1$ and $p_{i+1}$
    end for
  end for
end for
return findThresholds($D$)
procedure findThresholdRanges($k$: integer)
for $j \leftarrow 0$ to $k + 1$ do
  if $j = k + 1$ then
    minTj($j$) ← $n$
  else
    minTj($j$) ← $j$
  end if
  if $j = 0$ then
    maxTj($j$) ← 0
  else
    maxTj($j$) ← $n - k + j - 1$
  end if
end for
return minTj, maxTj
end procedure
procedure findThresholds($D$ : table)
$T(k + 1) \leftarrow n$
for $j \leftarrow k$ downto 0 do
  $T(j) \leftarrow D(T(j + 1), j + 1)$
end for
return $T$
end procedure
```

and

$$\max\{t_j\} = \begin{cases} 0 & \text{if } j = 0 \\ n-k+j-1 & \text{if } 1 \leq j \leq k+1, \end{cases} \tag{7}$$

The polynomial-time thresholding algorithm is described in Algorithm 1.6.1. The algorithm uses tables C & D, whose number of rows is equal to the number of bins in the histogram, and number of columns is equal to $k+2$, where k is the number of thresholds. The table $C(t_j, j)$ contains the optimal solution for $T_{0,j} = \{t_0, t_1, ..., t_j\}, \Psi^*(T_{0,j})$, which is found from $\text{min}t_j \leq t_j \leq \text{max}t_j$. While the table $D(t_j, j)$, contains the value of t_{j1} for which $\Psi^*(T_0, j)$ is optimal. The algorithm runs in $O(kn^2)$, where n is the number of bins in the histogram and k is the number of thresholds. This means that the computational cost increases quadratically with the increase in the length of the histogram. This algorithm is good for medium to large-sized histograms, but extremely large sized histogram are still a problem. Over the years various metaheuristic algorithms have been proposed as a solution to the problem, including those published in [8, 6, 3, 9, 5, 21, 7], just to mention a few, though they all lead to sub-optimal solutions.

1.7 More Efficient Algorithms

In the previous section, we have mentioend that by making use of dynamic programming, it is possible to optimize the thresholding criterion and find the optimal thresholds with quadratic time complexity, $O(kn^2)$. This is not suitable for extremely large histograms, 3 billion bins incase of human genome [10], as the time needed to find the optimal thresholds increases quadratically with the number of bins in the histogram. However, it possible to further reduce the complexity of the thresholding algorithm by making use of some special properties. Unlike the dynamic programming solution, which is applicable for many criteria, the method introduced here can only be used if the criterion used satisfy certain properties.

Table 1.6.1: An example table C, D for an histogram with n bins and $k = 4$ thresholds.

$C, D:$

i, j	0	1	2	3	4	5
0	0	x	x	x	x	x
1	x		x	x	x	x
2	x			x	x	x
3	x				x	x
4	x					x
:	x					x
:	x					x
:	x					x
n-4	x					x
n-3	x	x				x
n-2	x	x	x			x
n-1	x	x	x	x		x
n	x	x	x	x	x	

From Table 1.6.1, a few observations can be made. The number of rows corresponds to the number of bins, n, in the histogram and the number of columns depends on the number of thresholds, k. Each cell, represents the partial sum of the function, described in Equation (4), up to that particular point on the histogram for given value of j. At any given value of j, the histogram is divided into $j + 1$ classes. Also, at any given value of j, the task of filling in the column is equivalent to the problem of finding the maxima of each row of a lower-triangular matrix, M. The definition of this matrix, hereby known as the search matrix, can be derived from Equation (4) and (5) as follows:

$$M(r, c) = \begin{cases} -\infty & \text{if } c > r, \\ \Psi^*_{j-1}(c + j - 2) + \psi(c + j - 2, r + j - 1) & \text{if } c \leq r, \end{cases} \tag{8}$$

where j denotes the stage in the trellis, r the row index and c the column index.

For a normal matrix with no special properties, finding the row-wise maxima in the lower-triangular region will require calculation of all the elements in that region, which is exactly the same as filling up one column in the table C of the dynamic programming algorithm.

However, depending on the criterion used, the resulting matrix can satisfy certain properties that makes it possible to find the row-wise maxima without calculating all elements, which as a result, reduces the overall time complexity of the algorithm.

Let us assume that for a certain criterion, the resulting function $\psi_{l,r}$, satisfies the following property:

$$\psi_{a,u} + \psi_{b,v} \geq \psi_{a,v} + \psi_{b,u}, \tag{9}$$

where $1 \leq a < b < u < b \leq n$

This property is known as *convex quadrangle inequality.* If the criterion used satisfies this property and we take four elements from the lower region of the resulting matrix M such that $1 \leq r_1 < r_2 \leq n - k + 1$ and $1 \leq c_1 < c_2 \leq r_1$, here r and c are the rows and columns respectively. From Equation (8) and (9) it can be seen that:

$$M(r_1, c_1) + M(r_2, c_2) \geq M(r_1, c_2) + M(r_2, c_1)\,. \tag{10}$$

This would indicate that M lower triangular inverse Monge matrix. Monge matries have certain properties that can be exploited to solve optimization problems [12]. The property that we are interested and can take advantage of is the fact that inverse Monge matrix are always totally monotone.

In the following subsections, two algorithms that take advantage of the inverse Monge matrix are discussed briefly. The first algorithm exploits the monotonicity of the matrix. This algorithm has not been used in this work as the other algorithm exhibits a better performance. The second algorithm requires a totally monotone matrix and can achieve linear-time complexity. When working with both algorithms, it is not necessary to explicitly define the matrix and work with a matrix that is implicitly defined. This means that the matrix entries are not calculated until accessed by the algorithm. In fact, if the matrix was pre-calculated, it would be no different from the dynamic programming approach and would end up with the same quadratic-time complexity.

1.7.1 Divide-and-Conquer

The row maxima in a monotone matrix build a staircase like structure, where the maxima of the current row can appear only in the columns appearing after the column containing the row maixma of the previous row. By exploiting this fact, the algorithm maxima of the middle row of the matrix, recursively repeating itself on two submatrices, stopping when there is only one row remaining. This algorithm achieves a time complexity of $O(n \log m)$, where n is the number of rows and m is the number of columns of the given matrix. The pseudocode of this algorithm is given in Algorithm 1.7.1.

Algorithm 1.7.1 DIVCONQ(M)

```
$[m, n] \leftarrow$ size of $M$ {rows, columns}
$j \leftarrow$ position of leftmost maxima in row $[\frac{m}{2}]$ of M
store the postion of the maxima
if $m = 1$ then
  return
else
  if $[\frac{m}{2} \neq 1]$ then
    $A \leftarrow$ submatrix with rows 1 to $[\frac{m}{2}]$-1 and columns 1 to $j$ of $M$
    DIVCONQ($A$)
  end if
  $B \leftarrow$ submatrix with rows $[\frac{m}{2}]$+1 to $m$ and columns $j$ to $n$ of $M$
  DIVCONQ($B$)
end if
```

1.7.2 SMAWK

The SMAWK algorithm [1] was named after its inventors Shor, Moran, Aggarwal, Wilbe and Klawe. The SMAWK algorithm requires that the search matrix be a totally monotone matrix and will not work if the matrix is only monotone. Just like Divide-and-Conquer, it is also a recursive algorithm, but is capable of finding the row-wise minima or maxima of an $m \times n$ matrix in $O(n)$ time, as opposed to $O(n \log m)$ time required by the divide-and-conquer algorithm. A more detailed explanation is available in [1].

The pseudocode of the SMAWK algorithm is presented in Chapter 2. For the sake of clarity, furhter discussions and an example are included in this section. The algorithm is composed of three functions, viz SMAWK, REDUCE, and MFILL. It starts with the call of the SMAWK function. The REDUCE function is the central part of the algorithm as it reduces an $m \times n$ matrix to an $m \times m$ matrix by removing $n - m$ columns that do not contain row maxima. This can be done in $O(n)$ time.

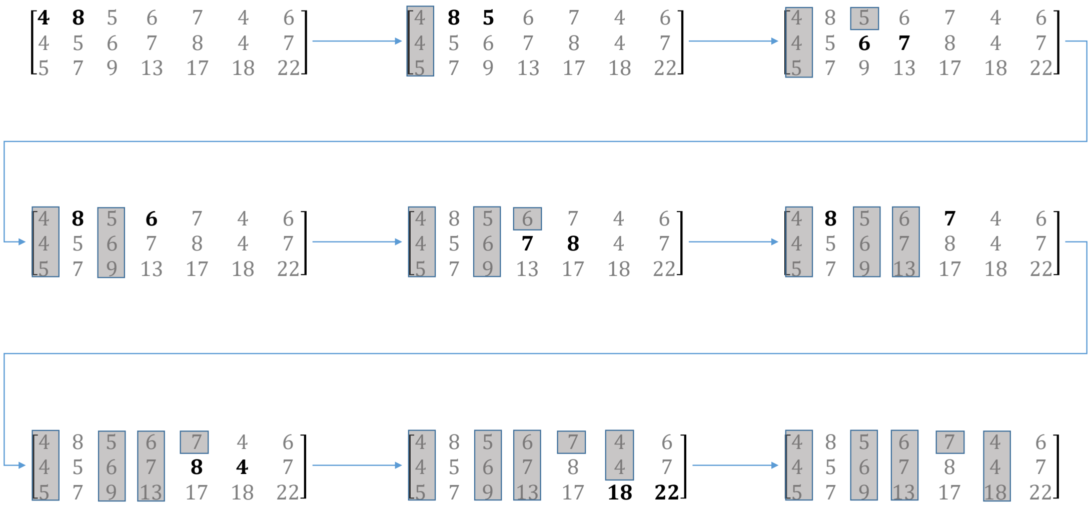

Fig. 1.7.1: An example of the REDUCE function in action.

An example of the REDUCE process is shown in Figure 1.7.1. The REDUCE function compares the elements which are highlighted in bold face. Only when the algorithm compares the elements, these elements are calculated, while the other elements are never calculated. Positions that are determined to not yield a maximum are depicted in a gray background. In the example depicted in Figure 1.7.1, we start with at the first row comparing the first two elements, '4' and '8'. Since '4' is smaller than '8', we delete the column of '4' as it is guaranteed to not contain the row maximum of any of the three rows. When comparing the next two elements, '8' and '5', since '8' is greater than '5', we do not delete the column of '8' as it may contain a row maximum. Then, we move to the next row comparing elements '6' and '7', and since '6' is the smallest of the two, we delete its column, as it will not contain any row maximum. Afterwards, the previously found maximum of the previous row, '8', is compared with the next available element, '7'. This makes sure that the maximum of the previous row still remains. The same process is repeated until we reach the element at the last row and last column of the matrix. This results in the deletion of four columns of a 3×7 matrix, while preserving the row maximum containing columns.

After the REDUCE function is called, even-numbered rows of the reduced matrix are selected and the SMAWK function recursively calls itself until the REDUCE function returns a 1×1 matrix containing a row maxima. The MFILL function finds the maxima in the odd-numbered rows very efficiently, since the position of the maxima in even-numbered rows is already known. The optimal thresholds can be found in $O(n)$ by combining the dynamic programming approach and the SMAWK algorithm. The use of the SMAWK algorithm to achieve linear-time optimal multi-level thresholding is further explained in Chapter 2.

1.8 Proposed Method

This thesis proposes a new method for detection of binding sites in ChIP-Seq experiments. The proposed method makes use of the concept of optimal multi-level thresholding to effectively achieve optimal one-dimensional clustering of histograms. As the data resulting from ChIP-Seq experiments can generate extremely large histograms, in the magnitude of 3 billion bins (base pairs), the use of a quadratic-time algorithm is not convenient. For this reason the emphasis is on the use of a linear-time algorithm.

The combination of dynamic programming and the SMAWK algorithm results in a method linear in the length of the histogram as well as the number of thresholds. Thus, making fast and optimal thresholding of large histograms a possibility. The use of cluster validity indices makes sure that not only the positioning of the thresholds is optimal, but also that the number of thresholds is also optimal. Finally, all of these concepts are harmoniously implemented into a self-contained unit, which we call a sliding-window. The use of multiple concurrent sliding-windows, independent of each other, results in a framework capable of utilizing modern-day multi-core CPU architecture to its fullest potential. The performance of this approach is examined using multiple ChIP-Seq datasets targeting the well-known histone modification protein H3K27ac. Details of this approach is described in Chapter 2.

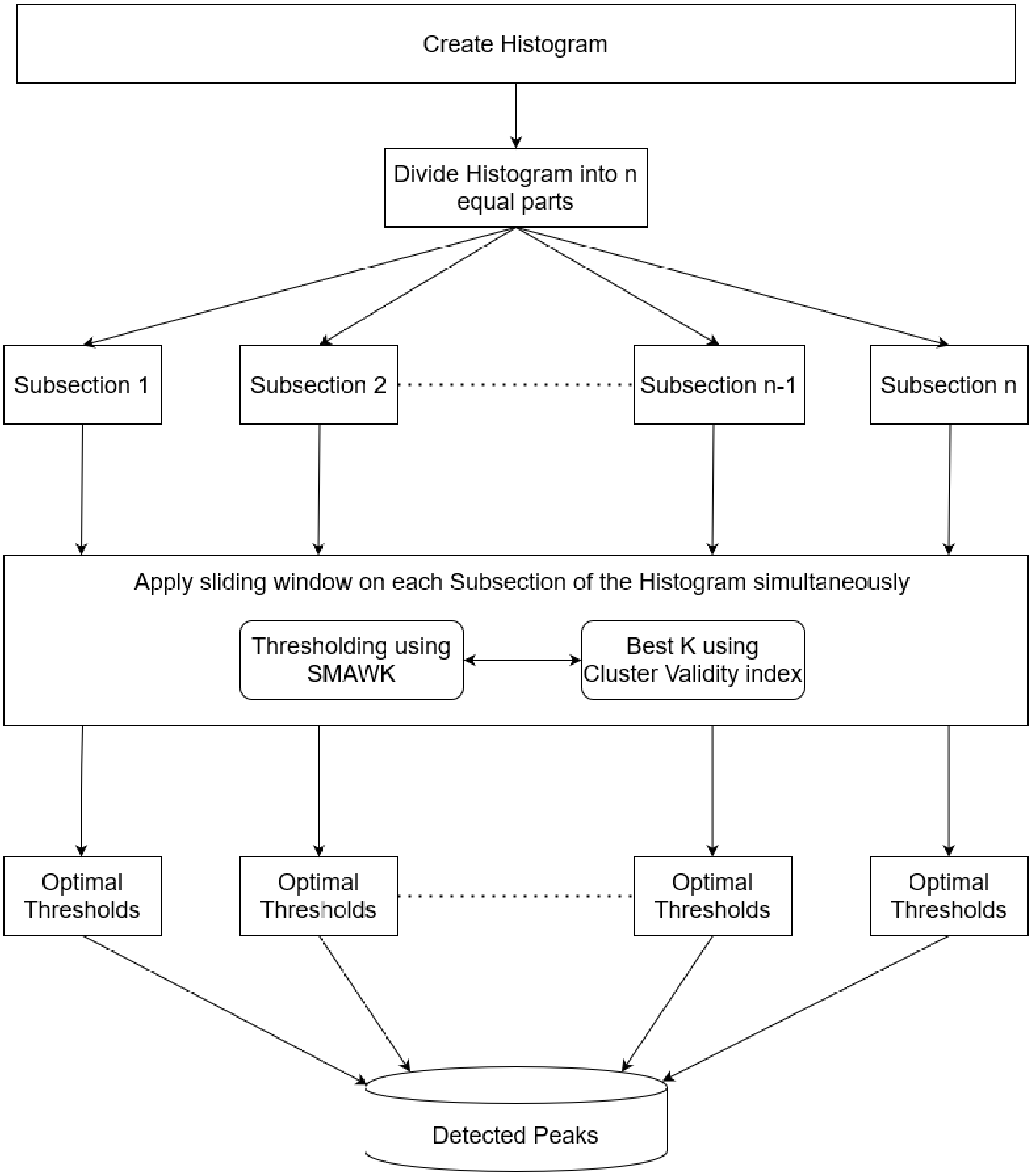

Fig. 1.8.1: A block diagram of the proposed method.

1.8.1 Contributions

The contributions of this thesis can be summarized as follows:

- Proposed a very fast method for the detection of binding sites in data generated by ChIP-Seq experiments.
- Provided a mechanism to find optimal location as well as optimal number of thresholds in large histograms in linear-time complexity.
- Envisioned and implemented a mechanism to take advantage of modern-day multi-core CPU architecture.
- Developed a Python package for the proposed method [24].
- Provided a framework for optimal and fast one-dimensional clustering of discrete points, which can be extended to real-numbered points in one dimension and applied to other fields beyond image segmentation and bioinformatics.

References

[1] Alok Aggarwal et al. “Geometric applications of a matrix-searching algorithm”. In: *Algorithmica* 2 (Nov. 1987), pp. 195–208. DOI: `10.1007/bf01840359`.

[2] Bruce Alberts et al. *Molecular biology of the cell.* Garland Science, 2015.

[3] Mohamed Abd El Aziz et al. “Whale Optimization Algorithm and Moth-Flame Optimization for multilevel thresholding image segmentation”. In: *Expert Systems with Applications* 83 (Oct. 2017), pp. 242–256. DOI: `10.1016/j.eswa.2017.04.023`.

[4] Timothy Bailey et al. “Practical Guidelines for the Comprehensive Analysis of ChIP-seq Data”. In: *PLoS Computational Biology* 9 (Nov. 2013). Ed. by Fran Lewitter, e1003326. DOI: `10.1371/journal.pcbi.1003326`.

[5] Xiaoli Bao et al. “A Novel Hybrid Harris Hawks Optimization for Color Image Multilevel Thresholding Segmentation”. In: *IEEE Access* 7 (2019), pp. 76529–76546. DOI: `10.1109/ACCESS.2019.2921545`.

[6] Ashish K. Bhandari et al. “Modified artificial bee colony based computationally efficient multilevel thresholding for satellite image segmentation using Kapur’s, Otsu and Tsallis functions”. In: *Expert Systems with Applications* 42 (Feb. 2015), pp. 1573–1601. DOI: `10.1016/j.eswa.2014.09.049`.

[7] Ashish Kumar Bhandari. “A novel beta differential evolution algorithm-based fast multilevel thresholding for color image segmentation”. In: *Neural Computing and Applications* 32 (May 2020), pp. 4583–4613. DOI: `10.1007/s00521-018-3771-z`.

[8] Ashish Kumar Bhandari et al. “Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur’s entropy”. In: *Expert Systems with Applications* 41 (June 2014), pp. 3538–3560. DOI: `10.1016/j.eswa.2013.10.059`.

[9] Vijay Kumar Bohat et al. “A new heuristic for multilevel thresholding of images”. In: *Expert Systems with Applications* 117 (Mar. 2019), pp. 176–203. DOI: `10.1016/j.eswa.2018.08.045`.

[10] Terence A Brown. *The Human Genome*. Nih.gov, 2016. URL: `https://www.ncbi.nlm.nih.gov/books/NBK21134/` (visited on 12/22/2019).

[11] Michael J Buck et al. “ChIP-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments”. In: *Genomics* 83 (Mar. 2004), pp. 349–360. DOI: `10.1016/j.ygeno.2003.11.004`.

[12] Rainer E. Burkard et al. “Perspectives of Monge properties in optimization”. In: *Discrete Applied Mathematics* 70 (Sept. 1996), pp. 95–161. DOI: `10.1016/0166-218x(95)00103-x`.

[13] ENCODE Project Consortium. “An integrated encyclopedia of DNA elements in the human genome”. In: *Nature* 489 (Sept. 2012), pp. 57–74. DOI: `10.1038/nature11247`.

[14] Alexander Dobin et al. “STAR: ultrafast universal RNA-seq aligner”. In: *Bioinformatics* 29 (Oct. 2012), pp. 15–21. DOI: `10.1093/bioinformatics/bts635`.

[15] K. Erciyes. “Introduction to Molecular Biology”. In: *Distributed and Sequential Algorithms for Bioinformatics*. Cham: Springer International Publishing, 2015, pp. 11–25. ISBN: 978-3-319-24966-7. DOI: `10.1007/978-3-319-24966-7_2`. URL: `https://doi.org/10.1007/978-3-319-24966-7_2`.

[16] M. J. Fullwood et al. “Next-generation DNA sequencing of paired-end tags (PET) for transcriptome and genome analyses”. In: *Genome Research* 19 (Apr. 2009), pp. 521–532. DOI: `10.1101/gr.074906.107`.

[17] D. S. Gilmour et al. “Detecting protein-DNA interactions in vivo: distribution of RNA polymerase on specific bacterial genes.” In: *Proceedings of the National Academy of Sciences* 81 (July 1984), pp. 4275–4279. DOI: `10.1073/pnas.81.14.4275`.

[18] David S. Johnson et al. “Genome-Wide Mapping of in Vivo Protein-DNA Interactions”. In: *Science* 316 (June 2007), pp. 1497–1502. DOI: `10.1126/science.1141319`.

[19] Jagat N. Kapur et al. “A new method for gray-level picture thresholding using the entropy of the histogram”. In: *Computer Vision, Graphics, and Image Processing* 29 (Mar. 1985), pp. 273–285. DOI: `10.1016/0734-189x(85)90125-2`.

[20] Josef Kittler et al. “Minimum error thresholding”. In: *Pattern Recognition* 19 (Jan. 1986), pp. 41–47. DOI: `10.1016/0031-3203(86)90030-0`.

[21] Hongnan Liang et al. “Modified Grasshopper Algorithm-Based Multilevel Thresholding for Color Image Segmentation”. In: *IEEE Access* 7 (2019), pp. 11258–11295. DOI: `10.1109/access.2019.2891673`.

[22] Marcel Margulies et al. “Genome sequencing in microfabricated high-density picolitre reactors”. In: *Nature* 437 (2005), pp. 376–80. DOI: `10.1038/nature03959`.

[23] Allan M. Maxam et al. “[57] Sequencing end-labeled DNA with base-specific chemical cleavages”. In: *Nucleic Acids Part I* (1980), pp. 499–560. DOI: `10.1016/s0076-6879(80)65059-9`.

[24] Musab Naik. *Github Repo: LinMLTBS*. `https://github.com/MusabNaik/LinMLTBS`. 2020.

[25] *NovaSeq 6000 System*. www.illumina.com. URL: `https://www.illumina.com/systems/sequencing-platforms/novaseq.html` (visited on 08/27/2020).

[26] Nobuyuki Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9 (Jan. 1979), pp. 62–66. DOI: `10.1109/tsmc.1979.4310076`.

[27] Peter J. Park. “ChIP–seq: advantages and challenges of a maturing technology”. In: *Nature Reviews Genetics* 10 (Sept. 2009), pp. 669–680. DOI: `10.1038/nrg2641`.

[28] Ho Sung Rhee et al. “Comprehensive Genome-wide Protein-DNA Interactions Detected at Single-Nucleotide Resolution”. In: *Cell* 147 (Dec. 2011), pp. 1408–1419. DOI: `10.1016/j.cell.2011.11.013`. (Visited on 01/29/2020).

[29] Luis Rueda. “An Efficient Algorithm for Optimal Multilevel Thresholding of Irregularly Sampled Histograms”. In: *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. SSPR amp; SPR ’08. Orlando, Florida: Springer-Verlag, 2008, pp. 602–611. ISBN: 9783540896883. DOI: `10.1007/978-3-540-89689-0_64`. URL: `https://doi.org/10.1007/978-3-540-89689-0_64`.

[30] F Sanger et al. “A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase”. In: *Journal of Molecular Biology* 94 (1975), pp. 441–8. DOI: `10.1016/0022-2836(75)90213-2`.

[31] Wikimedia Commons: Sponk. *File:Difference DNA RNA-DE.svg: Sponk / *translation: Sponk / CC BY-SA*. [Online; accessed 26-August-2020]. 2010. URL: `https://commons.wikimedia.org/wiki/File:Difference_DNA_RNA-EN.svg`.

[32] Cole Trapnell et al. "TopHat: discovering splice junctions with RNA-Seq". In: *Bioinformatics* 25 (Mar. 2009), pp. 1105–1111. DOI: `10.1093/bioinformatics/btp120`.

CHAPTER 2

Finding Binding Sites in ChIP-Seq Data via a Linear-time Multi-level Thresholding Algorithm

2.1 Introduction

Chromatin immunoprecipitation followed by high-throughput sequencing (ChIP-Seq), first described in 2007 by Johnson et al. [13], is a technique that maps genome-wide protein-DNA interactions. Determining how the protein-DNA interactions regulate gene expression, is a crucial step in understanding many biological processes. Using ChIP-Seq, it is possible to identify unique regions on the genome with which a target protein interacts with [5].

ChIP-Seq data is extracted by first cross-linking the Chromatin with the target protein, followed by its fragmentation, usually by sonication. Immunoprecipitation is then performed on the fragments using an antibody-specific to the target protein, after which the fragments are filtered such that only the immunoprecipitated fragments are left behind. These fragments, after reverse cross-linking, are sequenced and aligned to the reference genome. A signal or histogram can be created using these aligned fragments, where the x-axis represents the genome coordinate and the y-axis represents the number of unique reads at each genome coordinate. Figure

2.1.1 shows the process of creating a histogram by aligning the immunoprecipitated fragments over the reference genome. Each point on the x-axis corresponds to a nucleotide. Fragments that align over the same region in the reference genome are stacked over each other. The count of unique matches at each point on the x-axis is represented by the y-axis.

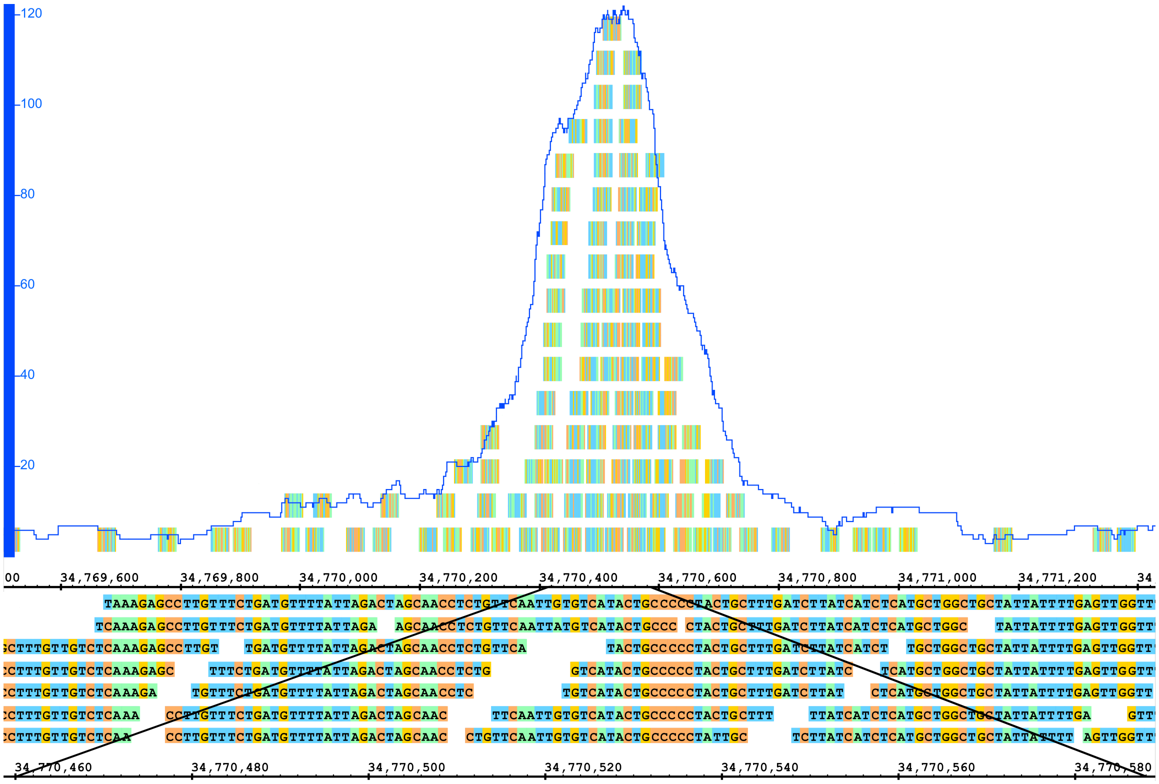

Fig. 2.1.1: Alignment of the immunoprecipitated fragments over the reference genome resulting in a continuous signal where the x-axis represents the genome coordinate and the y-axis corresponds to the number of reads at each coordinate. A small section is zoomed-in to help visualize the aligned fragments.

From the millions of short reads aligned over the reference genome, putative identification of the protein binding sites (peaks) is truly a bioinformatic challenge that requires considerable computational resources [3]. A computational technique known as *Peak Calling* is used to solve the aforementioned problem. Numerous peak finding algorithms have been developed in order to satisfy the ever growing popularity of ChIP-Seq, though identifying peaks of varying size with accuracy and in respectable time is still a challenge.

One such widely used peak finding method is *Model-based analysis of ChIP-Seq* (MACS), which analyzes data generated by short read sequencers [36]. Over the years MACS has become a standard for detection of binding sites in next generation sequencing. Although MACS can produce results in a respectable time, it is not guaranteed to find all the peaks in one run, and may require the user to search over a large set of parameters to obtain optimal results. MACS models the length of the sequenced ChIP fragments and uses it to improve the spatial resolution of predicted binding sites. It utilizes a local Poisson model that calculates a λ_{local} parameter at each local genomic position. A few other methods that work on a similar principle are SICER [35], MOSAiCS [18], ZINBA [25], and BCP [33].

In [12], Hocking et al. have proposed the use of a supervised machine learning algorithm to analyze the ChIP-Seq data and identify the peaks. The main idea is to have a data set labeled by experts, learn a model thought it and then make peak predictions on the rest of the genome. The success of this approach heavily relies on the quality of the labeled data. Manually labeling data is no doubt a labour intensive task and even the experts may not agree whether a given peak is significant or not and also on small details such as where a peak starts and ends. Some of the more recent works make use of machine learning concepts such as recurrent neural networks [32], adversarial networks [19], and convolutional neural networks [23], though they are very slow and need extra data to identify relevant peaks, where such data may not be available. A constrained multi-level thresholding (CMT) is a method that makes use of multi-level thresholding algorithm to find relevant peaks, proposed in [26, 27]. Here, the problem of peak finding is converted into a problem of finding ”valleys” in a continuous signal generated by the alignment of fragments over the reference genome. It makes use of Otsu's method along with dynamic programming in order to find the optimal thresholds.

On the other hand, multi-level thresholding is a field that has emerged in the past decades and has been applied to image segmentation, primarily, among other domains. Multi-level thresholding is essentially a one-dimensional clustering problem where the objective, as in any other clustering problem, is to produce clusters in such a way that the within-cluster scatter is minimized and the between-cluster distance is maximized. In multi-level thresholding, a cluster is define as the region between two thresholds. When the results are visualized, it can be seen that the thresholds are placed in such a way that they lie at an optimal position in the valleys between the clusters. The earliest works, such as Otsu et al. [24], Kapur et al. [14], and Kittler et al. [17], proposed algorithms to find bi-level thresholds. However, for optimal multi-level thresholds, exhaustive search is required which results in exponential increase of computational complexity with the increase in the number of thresholds. In [30], an algorithm was proposed for large histograms, which makes use of dynamic programming to achieve polynomial-time complexity. Large histograms are still a problem as the computational time increases quadratically with the increase in the length of the histogram. Over the years various metaheuristic algorithms have been proposed as a solution to the problem, including those published in [8, 6, 2, 9, 4, 20, 7], just to mention a few, though they all lead to sub-optimal solutions. In [22, 21], making use of the *SMAWK* algorithm, a fast, linear-time multi-level approach was proposed; the SMAWK algorithm was originally proposed by [1]. The result is an algorithm that runs in linear time in both, the number of thresholds and the length of the histogram. Multi-level thresholding has also been successfully used in biomedical fields, including segmentation of biofilm images, obtaining very good results due to the nature of those images, which are obtained from confocal microscopy [29, 28], and also in automating the process of gridding microarray images [31].

In this work, we introduce a method based on the *SMAWK* algorithm, which we call LinMLTBS, used to identify binding sites in ChIP-Seq experiments. Since existing methods require the user to finesse with a number of parameters of the method in order to obtain optimal results, minimizing the number of adjustable parameters while retaining high accuracy is one of the goals achieved by the proposed method. The proposed method has been shown to outperformed previous approaches when testing on six different cell-lines for the H3K27ac histone modification protein. The rest of the chapter is organized as follows. Section 2.2 discusses the main concepts used for the proposed method. Section 2.3 includes the datasets used, data pre-processing required, procedures followed for testing and comparison, and results obtained.

2.2 Methods

The main goal is to find relevant peaks corresponding to regions of binding sites for the targeted protein, and, ideally, rank these peaks by relevance. As previously mentioned, thresholding can be seen as one-dimensional clustering where the peaks are clusters separated by thresholds placed at optimal positions in the valleys separating the clusters.

To achieve this objective the *SMAWK* algorithm is used, which optimizes the Otsu's between-class variance criteria. To take advantage of modern-day multi-core processors, the input data is divided into equally-sized chunks and worked on simultaneously by all of the available cores. As Multi-level thresholding is essentially an unsupervised approach where the optimal number of thresholds or clusters are not known, the concept of *cluster validity indices* is used to determine the best number of thresholds. Each of the aforementioned concepts is further discussed in the following subsections.

2.2.1 Multi-level Thresholding

Multi-level thresholding is a process originally proposed for segmentation of images. This method proceeds by setting thresholds such that all pixels of an image, depending on their intensity value are separated into $k + 1$ classes, where k is the number of thresholds. To further understand the underlying concepts, let us consider the following definitions.

Definition 1 (Histogram): A histogram H is an ordered set $\{1, 2, ..., n-1, n\}$, where the i^{th} value corresponds to the i^{th} bin and has a probability, p_i.

Definition 2 (Probabilities): Let $F = \{f_1, f_2, ...f_n\}$ be the frequencies of the histogram H. The probabilities of H are given by the set $P = \{p_1, p_2, ..., p_n\}$, where $p_i \geq 0, \sum_{i=1}^{n} p_i = 1$, and $p_i = f_i/N$ with $N = \sum_{i=1}^{n} f_i$

Definition 3 (Threshold set): A threshold set T is defined as an ordered set $T = \{t_0, t_1, ..., t_k, t_{k+1}\}$, where $0 = t_0 < t_1 < ... < t_k < t_{k+1} = n$ and $t_i \in \{0\} \cup H$.

For an ordered subset of T, the following notation is used: $T_{i,j} = \{t_i, t_{i+1}, ..., t_j\}$, where $i, j = 0, 1, ..., k, k+1, i < j$, and $T = T_{0,k+1}$.

Thresholding Problem: The problem of multilevel thresholding consists of finding a threshold set T, in such a way that a function $f : H^k \times [0,1]^n \to \mathbb{R}^+$ is maximized/minimized. Using this threshold set, H is divided into $k+1$ classes: $\zeta_1 = \{1, 2, ..., t_1\}, \zeta_2 = \{t_1 + 1, t_1 + 2, ..., t_2\}, ..., \zeta_k = \{t_{k-1} + 1, t_{k-1} + 2, ..., t_k\}, \zeta_{k+1} = \{t_k + 1, t_k + 2, ..., n\}$.

The between-class variance criterion aims to maximize the sum of the between-class variances for each class, as follows:

$$\Psi_{BC}(T) = \sum_{j=1}^{k+1} \omega_j \mu_j^2, \tag{1}$$

where $\omega_j = \sum\limits_{i=t_{j-1}+1}^{t_j} p_i, \mu_j = \frac{1}{\omega_j} \sum\limits_{i=t_{j-1}+1}^{t_j} i p_i$.

The selection of the thresholds is based on optimizing the objective function. Optimal thresholds are the ones that either maximize or minimize the said objective function. The most straightforward approach is to evaluate the objective function for every possible combination of thresholds, which is essentially an exhaustive search with a time complexity that is exponential in the number of thresholds $O(n^k)$, where n is the number of bins in the histogram and k is the number of thresholds.

Using dynamic programming, it is possible to reduce the time complexity to $O(n^2)$. If the objective function meets certain criterion, it is possible to combine dynamic programming with divide-and-conquer to achieve a time complexity of $O(n \log n)$, or even $O(n)$ when dynamic programming is combined with the *SMAWK* algorithm. For the sake of brevity, only *SMAWK* algorithm and related concepts are further discussed. A more detailed explanation of the former approaches can be found in [30, 22, 21].

Definition 4 (function Ψ): Let $T = \{t_0, t_1, ..., t_k, t_{k+1}\}$ be a threshold set, where $k \geq 1$. The function Ψ is defined for all m, where $1 \leq m \leq k+1$, as a function

$\Psi : H^m \times [0,1]^n \to \mathbb{R}^+ \cup \{0\}$ as follows:

$$\Psi(T_{0,m}) = \Psi(\{t_0, t_1, ..., t_m\}) \triangleq \sum_{j=1}^{m} \psi_{t_{j-1}+1,t_j}, \tag{2}$$

where $1 \leq m \leq k+1$, and $\psi_{t_{j-1}+1,t_j}$ is the class cost of the class ζ_k.

The optimal solution for multi-level thresholding can be defined as optimal solutions to smaller sub-problems. This can be observed in the decomposition of $\Psi(T)$ as the sum of independent terms.

$$\Psi(T_{0,m}) = \Psi(\{t_0, t_1, ..., t_m\}) + \psi_{t_{m-1}+1,t_m} \tag{3}$$

Which means that if the optimal solution for $T_{0,j-1} = \{t_0, t_1, ..., t_{j-1}\}$, $\Psi^*(T_{0,j-1})$, is known. Then, the optimal solution for $T_{0,j}$, $\Psi^*(T_{0,j})$, is computed as follows:

$$\Psi^*(T_{0,j}) = \begin{cases} 0 & \text{if } j = 0 \\ \max_{\min\{t_{j-1}\} \geq t_{j-1} \geq \max\{t_{j-1}\}} & \\ \quad \Psi^*(T_{0,j-1}) + \psi_{t_{j-1}+1,t_j} & \text{if } 1 \leq j \leq k+1 \end{cases} \tag{4}$$

where

$$\min\{t_j\} = \begin{cases} j & \text{if } 0 \leq j \leq k \\ n & \text{if } j = k+1 \end{cases} \tag{5}$$

and

$$\max\{t_j\} = \begin{cases} 0 & \text{if } j = 0 \\ n-k+j-1 & \text{if } 1 \leq j \leq k+1 \end{cases} \tag{6}$$

The trellis structure, shown in Figure 2.2.1, helps better understand the task of finding the thresholds and how the algorithm is implemented. The x-axis represents the bin number, i, where $i = 0, ..., n$, while the current stage of the algorithm, m,

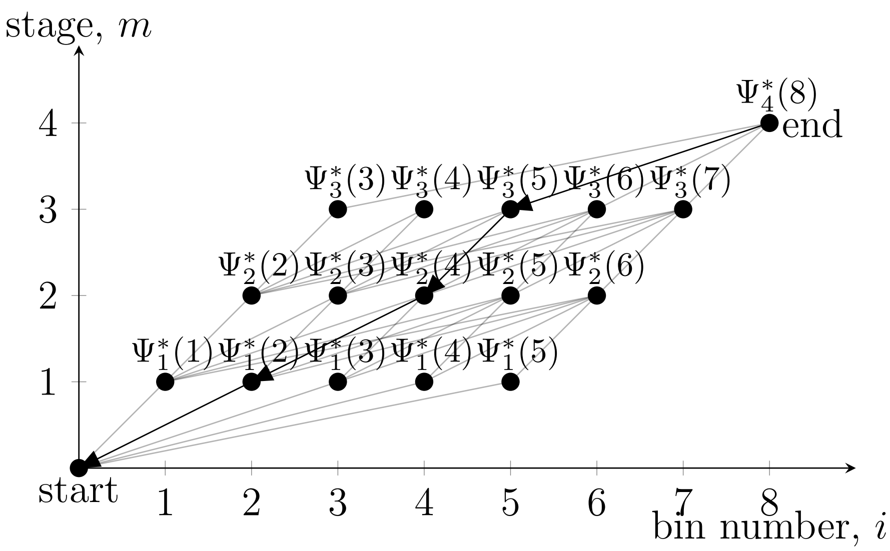
stage, m
4
3
2
1
start
1
2
3
4
5
6
7
8
bin number, i
$\Psi_4^*(8)$
end
$\Psi_3^*(3)$ $\Psi_3^*(4)$ $\Psi_3^*(5)$ $\Psi_3^*(6)$ $\Psi_3^*(7)$
$\Psi_2^*(2)$ $\Psi_2^*(3)$ $\Psi_2^*(4)$ $\Psi_2^*(5)$ $\Psi_2^*(6)$
$\Psi_1^*(1)$ $\Psi_1^*(2)$ $\Psi_1^*(3)$ $\Psi_1^*(4)$ $\Psi_1^*(5)$

Fig. 2.2.1: An example of a trellis structure corresponding to the steps to identify the optimal thresholds.

is represented by the y-axis. At stage m, m thresholds divide the interval $[1, i]$ into $m+1$ classes. At each node, two critical pieces of information are stored, the optimal partial sum up to that node, and a pointer that points back to the best node to come from. When the algorithm reaches the end node, the optimal solution is found by backtracking to the start node.

The problem of finding the optimal paths to all the nodes in one stage of the trellis, is equivalent to the problem of finding the row-wise maxima of a lower-triangular matrix [21]. The definition of this matrix, hereby known as the search matrix, can be derived from Equation (4) as follows:

$$M(r, c) = \begin{cases} -\infty & \text{if } c > r, \\ \Psi_{j-1}^*(c+j-2) & \\ \quad +\psi(c+j-2, r+j-1) & \text{if } c \leq r \end{cases} \tag{7}$$

where j denotes the stage in the trellis, r the row index and c the column index.

2.2.2 SMAWK Algorithm using Otsu's Criterion

The *SMAWK* algorithm [1] is a recursive algorithm capable of finding the row-wise maxima of an $m \times n$ matrix in $O(n)$ time. This is possible only when the matrix is a totally monotone. The search matrix will be totally monotone if the objective function satisfies the *convex quadrangle inequality*, which can be explained as follows. Assuming that for some objective function, Ψ, the class cost function ψ is of the form:

$$\psi_{p,q} = \omega_{p,q} \times f\left(\frac{\sum_{p<i\leq q} p(i).\gamma(i)}{\omega_{p,q}}\right), \tag{8}$$

where $\omega_{p,q}$ is the weight (probability) of the class, $f(x)$ is a convex function in the interval $[\gamma(1), \gamma(N)]$, and function $\gamma(x)$ is either monotonically increasing or decreasing in the interval $[1, N]$, then, the class cost function ψ fulfills the *convex quadrangle inequality.* The resulting matrix (7) is a totally monotone matrix. The proof of the above argument can be found in [21].

It has been shown that Otsu's between-class variance criterion (1), is of the form given in equation (10). Therefore, it fulfills the quadrangle inequality, which means that the resulting search matrix is totally monotone and the row-wise minima or maxima can be found in $O(n)$ using the *SMAWK* algorithm.

The pseudocode of the *SMAWK* algorithm is shown in Algorithm 1. The algorithm is composed of three functions, viz SMAWK, REDUCE, and MFILL. It starts with the call of the SMAWK function. The REDUCE function is the central part of the algorithm as it reduces an $m \times n$ matrix to an $m \times m$ matrix by removing $n - m$ columns that do not contain row maxima. This can be done in $O(n)$ time. The time complexity of the SMAWK algorithm has been derived in [1]. After the reduction, even-numbered rows of the reduced matrix are selected and the SMAWK function recursively calls itself until the REDUCE function returns a 1×1 matrix containing a row maxima. The MFILL function finds the maxima in the odd-numbered rows very efficiently, since the position of the maxima in even-numbered rows is already known. The optimal thresholds can be found in $O(n)$ by combining the dynamic programming approach and the SMAWK algorithm.

Algorithm 2.2.1 SMAWK (M)

```
$A \leftarrow$ REDUCE($M$)
if size of $A$ is $1 \times 1$ then
    $A$ is a maximum in the initial matrix, store position
    return
end if
$B \leftarrow$ matrix with only even-numbered rows of ($A$)
SMAWK($B$) {recursive call}
MFILL($A, B$) {find the maxima in odd rows of$A$}
REDUCE($M$):
$[m, n] \leftarrow$ size of $M$ {rows, columns}
$k \leftarrow 1$
while $M$ has more columns than rows do
    if M($k, k$) $\geq$ M($k, k + 1$) and $k < m$ then
        $m \leftarrow m + 1$
    else if M($k, k$) $\geq$ M($k, k + 1$) and $k = m$ then
        Delete column $k + 1$ of $M$
    else if M($k, k$) $<$ M($k, k + 1$) then
        Delete column $k$ of $M$
        if $k > 1$ then
            $k \leftarrow k - 1$
        end if
    end if
end while
MFILL($A, B$):
$[m, n] \leftarrow$ size of $A$ {rows, columns}
MPOS $[2, 4, ..., 2[m/2]] \leftarrow$ position of maxima in even-numbered rows of A
MPOS $[0] \leftarrow 1$; MPOS $[m + 1] \leftarrow n$
for $i \leftarrow 1$ to $[m/2]$ do
    $r \leftarrow 2i - 1$
    $max \leftarrow -\infty$
    for $c =$ MPOS$[r - 1]$ to MPOS$[r + 1]$ do
        if $A(r, c) > max$ then
            $max \leftarrow A(r, c)$
            MPOS$[r] = c$
        end if
    end for
end for
```

2.2.3 Cluster Validity Indices

The process of multi-level thresholding cannot be fully automated until there is a way to find the best number of thresholds without any input from the user. Multi-level thresholding is essentially equivalent to one-dimensional unsupervised clustering and it too suffers from the same problem faced by many other methods in the same category. In this context, the problem is to determine the best number of clusters such that the natural partitions of the clusters in a dataset are represented appropriately. Clustering algorithms generally require that the number of clusters be determined before clustering; in this sense, multi-level thresholding is no different.

The solution to this problem is to execute the clustering algorithm several times with different numbers of clusters, and based on a pre-defined criterion function select the number of clusters that gives the best results. *Cluster Validity Indices* are methods used to determine the goodness of the clustering. They are usually based on an analysis of variance, where the intra and inter-cluster variability are compared. A good clustering is expected to possess a small intra-cluster variance and a large inter-cluster separation at the same time.

In the proposed method, the $\alpha(k)$ index, which is a combination of a simple index, $A(k)$ and the well-known $I(k)$ index, is used and defined as follows:

$$\alpha(k) = \sqrt{k}\frac{I(k)}{A(k)} = \frac{(\frac{E_1}{E_k} \times D_K)^2}{\sqrt{k}\sum_{i=1}^{k} p(t_i)}\,, \tag{9}$$

where $E_k = \sum_{i=1}^{k}\sum_{k=1}^{n_i} p_k \|k - z_i\|$, $D_k = \overset{k}{\underbrace{\max}_{i,j=1}} \|Z_i - Z_j\|$, n is the total number of bins in the window, k is the number of clusters, Z_k is the center of the k^{th} cluster, t_I is the i^{th} threshold found by multi-level thresholding and $p(t_i)$ is the corresponding number of reads in the histogram.

For maximizing $I(k)$ and minimizing $A(k)$, the value of $\alpha(k)$ must be maximized. Thus, the best number of thresholds k^* based on the α index is given by:

$$k^* = \underset{1\leq k\leq\delta}{\operatorname{argmax}}\,\alpha(k) = \underset{1\leq k\leq\delta}{\operatorname{argmax}}\,\frac{(\frac{E_1}{E_k} \times D_k)^2}{\sqrt{k}\sum_{i=1}^{k} p(t_i)}\,, \tag{10}$$

where δ is the maximum number of clusters. This parameter needs to be determined depending on the application. When finding peaks in ChIP-Seq data, δ is determined based on the number of binding sites, which is specific to the type of protein being detected.

To find the optimal number of clusters (thresholds), we compute and compare values of $\alpha(k)$ over all possible numbers of clusters (thresholds) from 2 to $\sqrt{n}/2$, where n is the size of window. The one with the maximum value of $\alpha(k)$ corresponds to the best number of clusters (thresholds).

2.2.4 Concurrent Multi-window Approach

The proposed method makes use of a sliding window approach, where a window of a predetermined size is slid over the entire histogram. The window can be seen as a self-contained unit, which when applied to a region of the histogram, finds not only the optimal thresholds but also the optimal number of thresholds for that region of the histogram. The window is initially set to the start of the histogram, always maintaining the predetermined size. For the window at each position multi-level thresholding is applied, using the *SMAWK* algorithm for maximizing the objective function Ψ defined in Equation (1) to find a set of thresholds T_k, where k is the number of thresholds. To find the best number of thresholds, the concept of cluster validity index is used, by optimizing the formula of Equation (9). Once the optimal thresholds are determined, the peaks along with other relevant information are recorded. The window then slides forward in such a way that its start coincides with the position of the optimal threshold learned in the previous window. The entire process is repeated until the entire histogram has been scanned.

The proposed method also takes advantage of the modern-day multicore CPU architectures by deploying multiple windows that simultaneously and independently process different parts of the histogram. This is made possible by assigning each window a dedicated core from the CPU. Depending on the number of CPU cores available, the entire histogram is divided into equal length fragments, and each fragment is processed by a sliding window. A dataset containing the aligned fragments from the ChIP-Seq experiment, organized as a set of 23 chromosomes, along with the window-size is given as input. The detected peaks are the output. The pseudocode of the entire process is shown in Algorithm 2.

2.3 Results and Discussion

For the evaluation and comparison of the proposed method, tests were run on six different datasets obtained from ChIP-Seq experiments. The datasets used are publicly available on ENCODE portal, maintained by ENCODE Project Consortium [11] and can be downloaded as discussed in the following subsection. The results obtained by LinMLTBS are compared against MACS2 and CMT. Comparison is made in terms of speed with which the results are obtained, number of unique binding-sites found, and the biological significance of the discovered binding sites.

In Section 2.3.1, we describe the dataset used, reason for choosing this dataset, where and how the dataset can be obtained so that the results shown can be replicated, and pre-processing done on the dataset. Section 2.3.2 describes the testing environment, parameters used, the output obtained and the process used for biological validation. Finally, the results obtained by different methods are described and compared in Section 2.3.3.

2.3.1 Datasets

Histones are proteins with long chains of DNA molecules tightly wrapped around them, in order to contain approximately 1.8 metres of DNA in the nucleus of each cell in our bodies. These basic proteins (H2A, H2B, H3, and H4) not only provide structural support, but also play an important role in controlling the activities of the genes. Modifications to these proteins, a process also known as histone modification, can impact the gene expression by altering chromatin structure or recruiting histone modifiers, thus, the need to study them and predict the correlation between histone modification and gene expression. From the many known histone modifications, H3K27ac is one of the most studied case of histone modification. It is the

Algorithm 2.2.2 Concurrent Multi-window

```
Input: Dataset, $windowSize$ Output: Detected Peaks
  for Chr $\in$ DATASET do
    $H \leftarrow$ createHistogram(Chr) {Chr is a chromosome}
    $N_c \leftarrow$ no. of available CPUs
    $L \leftarrow$ length of Chr
    $FL \leftarrow L/N_c$ {Fragment Length}
    $H_F = \{H_1, ..., H_{N_c}\} \leftarrow$ DIV $H$ into $N_c$ equal fragments of length $FL$
    $W \leftarrow windowSize$
    for all fragments $\in H_F$ do in parallel
      while start of window < end of fragment do
        if slidingWindow not yet created then
          $start \leftarrow$ start of fragment
        else
          $start \leftarrow$ location of last threshold
        end if
        $end \leftarrow start + W$
        slidingWindow $\leftarrow$ makeWindow($start, end$)
        $k_{best} \leftarrow 0$ {best number of thresholds}
        $valScore_{best} \leftarrow 0$ {cluster validity score of best $k$}
        $T_{best} \leftarrow= \{\}$ {best thresholds found so far}
        for $k \leftarrow 2$ to $\sqrt{FL}/2$ do
          $T_k \leftarrow$ multilevelThresholding($k, W$)
          $valScore_k \leftarrow$ clusterValidity($T_k, k$)
          {assign validity score using Eq. 9}
          if $valScore_k > valScore_{best}$ then
            $valScore_{best} \leftarrow valScore_k$
            $k_{best} \leftarrow k$; $T_{best} \leftarrow T_k$
          end if
        end for
        $Peaks \leftarrow T_{best}, peakLen(T_{best}), peakVol(T_{best})$
        {Store peak position, length & volume}
      end while
    end for
  end for
  procedure multilevelThresholding($k, W$: integer)
  $trellis \leftarrow$ empty trellis {with $k$ stages and $W$ bins}
  $M \leftarrow$ empty $W \times W$ matrix
  for $stage : i \leftarrow 1$ to $k$ do
    $M \leftarrow$ fill search matrix according to objective func. $\Psi$
    {$\Psi$ defined in Eq. 1, $M$ defined in Eq. 7}
    $rowMAX \leftarrow$ SMAWK($M$)
    $trellis \leftarrow$ fill stage $i$ with $rowMAX$
  end for
  $T \leftarrow$ backTrack($trellis$); return $T$
```

modification of the H3 histone protein, more specifically, the acetylation of the H3 histone at the 27^{th} lysine residue. This specific modification is associated with gene activation.

ChIP-Seq has proven to be a robust and comprehensive technique for genome-wide studies of histone modification. For testing the proposed method, datasets of ChIP-Seq experiments performed on six different cell lines using antibody targeting the H3K27ac protein have been used. The cell lines and their respective accession number are depicted in 2.3.1. These datasets can be downloaded from the ENCODE Portal [15]. For the purpose of peak annotation, the Bioconductor package *TxDb.Hsapiens.UCSC.hg19.knowngene* version 3.2.2 [10], which contains the genomic features of the entire human genome, generated from the UCSC database [16], has been used.

Cell line	ENCODE Accession no.	GEO Accession no.
GM12878	ENCSR000AKC	GEO:GSM733771
HSMM	ENCSR000ANF	GEO:GSM733755
HUVEC	ENCSR000ALB	GEO:GSM733691
K562	ENCSR000AKP	GEO:GSM733656
NHEK	ENCSR000ALK	GEO:GSM733674
NHLF	ENCSR000AMR	GEO:GSM733646

Table 2.3.1: Accession no. & cell line for ChIP-Seq experiments targeting the H3K27ac histone modification.

2.3.2 Testing and Biological Validation

Tests were run on all three methods, LinMLTBS, CMT and MACS2, using the same procedure and datasets on a system that includes a Xenon 12 core processor and 32 Gigabytes of RAM. For all methods, default parameters were used throughout the test. For LinMLTBS, the only adjustable parameter is the size of the sliding window; the default size of the window is 10,000 bp. The number of CPU cores available are automatically detected and the maximum number of cores are used by default. However, the user may choose to limit the number of cores used.

The output from all three methods is in the form of a list containing the detected peaks, the genomic coordinates of the peaks, and a score for each peak, based on some criteria to rank the peaks. LinMLTBS, along with the genomic coordinates of the detected peaks, provides the volume of each peak. The peaks were ranked based on volume.

The detected peaks are biologically validated by a process known as peak annotation. For this ChIPseeker [34], an R/Bioconductor package for ChIP peak annotation, comparison and visualization, has been used. Based on the distance from the peak to the Transcriptional Start Site (TSS) of the nearest known gene, ChIPseeker assigns a genomic annotation such as Promoter, 5' UTR, 3' UTR, Exon, Intron, Downstream, or Intergenic to the peak.

2.3.3 Comparison with Other Methods

Since both LinMLTBS and CMT work using the same underlying principle, *optimal* multi-level thresholding, they produce similar results in terms of the of the peaks detected. However, they differ in the speed with which they produce the results. This is due to the difference in time complexity of the thresholding algorithms used. CMT uses a dynamic programming approach for its thresholding algorithm, which results in a time complexity of $O(n^2)$, where n is the number of bins in the histogram (or the number of base pairs in the genome). This means that the execution time increases quadratically with the increase in the length of the histogram. This is not a big problem for thresholding of images, where the total number of pixels is usually in the range of 10-20 million. However, with a count of 3 billion base pairs in the case of the human genome, the use of a quadratic-time algorithm becomes very inconvenient. LinMLTBS on the other hand, due to the combination of dynamic programming and the *SMAWK* algorithm, is able to achieve linear time complexity, namely $O(n)$, for its thresholding algorithm. Even though the difference in time complexities says it all, to further contextualize the difference in performance, both the methods were run on the same dataset using the same computational environment. CMT took about 120 minutes to produce its results, whereas LinMLTBS took only 20 minutes. Figure 2.3.1 shows yet another experiment performed to visualize the performance difference between quadratic and linear-time algorithms. Here, the log-log plots depict the execution time for both methods over a range of the input sizes. The times account of processing a single window. The trend in the plots show that LinMLTBS is able to process a window in a fraction of a second, while finding the optimal number of seconds. In contrast, the quadratic time algorithm takes thousands of seconds for a similar window size. As observed in the plot the different will grow asymptotical with the size of the window.

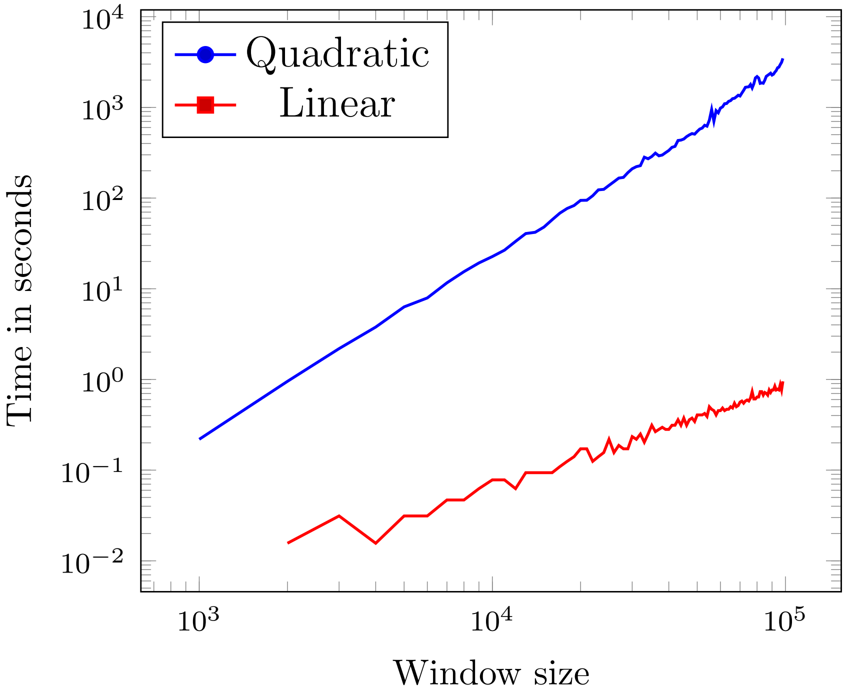

Fig. 2.3.1: Log-log graph of window size vs CPU time for quadratic and linear time algorithms.

Comparing LinMLTBS and MACS2, Table 2.3.2 shows the number of peaks detected by each method for each of the six ChIP-Seq datasets used. It can be seen that LinMLTBS detects significantly higher number of peaks compared to MACS2. Even when only the top 1% of the peaks (ranked by volume) detected by LinMLTBS are taken into consideration, in 4 of 6 cases, LinMLTBS still has higher number of peaks.

Owing to the large disparity in the number of peaks detected, it was decided that a similar number of peaks from both methods should be considered for further testing. After weighing a few options, it was determined that, the number of ranked peaks taken from LinMLTBS should be equal to 1.5 times the number of peaks detected by MACS2. This makes it a fair comparison against MACS2 while not forgetting the fact that LinMLTBS detected a larger number of peaks.

Table 3 shows the results of the annotation of all the peaks detected by MACS2 and the top ranked peaks detected by LinMLTBS, where the number of peaks selected from LinMLTBS is 1.5 times the number of peaks detected by MACS2. For annotation of the peaks, ChIPseeker with default parameters was used [34]. It can be seen that LinMLTBS exhibits better performance in all but a few cases. From the annotation results, it can be further derived that LinMLTBS not only detects a higher percentage of biologically significant regions such as Promoters, 5' UTR, 3' UTR, and Exons, but also manages to capture a lesser percentage of Introns and Intergenic regions, which have little to no biological significance. Out of the regions detected by LinMLTBS an average of 11% of them are promoters, 0.7% 5' UTR, 2.3% 3' UTR, and 6.5% exons. In contrast, in case of MACS2, 9% are promoters, 0.09% 5' UTR, 1.2% 3' UTR, and 1.6% are exons.

Table 2.3.2: Number of peaks detected by LinMLBTS and MACS2.

Cell line	MACS2	LinMLTBS	LinMLTBS_Top1%	LinMLTBS_1.5X	LinMLTBS_2X
GM12878	40,100	10,867,950	108,679	60,150	80,200
HSMM	60,650	8,226,860	82,268	90,975	121,300
HUVEC	57,453	4,164,444	41,644	86,179	114,906
K562	35,123	6,479,268	64,792	52,684	70,246
NHEK	61,878	4,323,250	43,232	92,817	123,756
NHLF	40,821	4,210,062	42,100	61,231	81,642

Table 2.3.3: H3K27ac - (LinMLTBS with 1.5 times the number of peaks)

Gene Classification	Cell Type	GM12878	HSMM	HUVEC	K562	NHEK	NHLF
	Method						
Promoter(<=1kb) %	**LinMLTBS**	🟩 5.07	🟩 4.12	🟩 4.42	🟩 5.62	🟩 4.76	🟩 5.11
	MACS2	🟥 3.59	🟥 3.07	🟥 3.1	🟥 3.72	🟥 3.17	🟥 3.27
Promoter(1-2kb) %	**LinMLTBS**	🟩 3.09	🟩 2.73	🟥 2.74	🟩 3.63	🟩 3.01	🟩 3.24
	MACS2	🟥 3	🟥 2.64	🟩 2.76	🟥 3.33	🟥 2.78	🟥 2.89
Promoter(2-3kb) %	**LinMLTBS**	3.01	🟩 2.61	🟩 2.78	🟩 3.41	🟩 2.92	🟩 3
	MACS2	3.01	🟥 2.53	🟥 2.64	🟥 3.35	🟥 2.63	🟥 2.8
5' UTR %	**LinMLTBS**	🟩 0.65	🟩 0.59	🟩 0.63	🟩 0.75	🟩 0.71	🟩 0.83
	MACS2	🟥 0.1	🟥 0.08	🟥 0.1	🟥 0.09	🟥 0.1	🟥 0.08
3' UTR %	**LinMLTBS**	🟩 2.49	🟩 2.08	🟩 2.15	🟩 2.58	🟩 2.22	🟩 2.32
	MACS2	🟥 1.27	🟥 1.14	🟥 1.19	🟥 1.36	🟥 1.1	🟥 1.15
1st Exon %	**LinMLTBS**	🟩 0.24	🟩 0.24	🟩 0.26	🟩 0.25	🟩 0.26	🟩 0.29
	MACS2	🟥 0.13	🟥 0.1	🟥 0.09	🟥 0.13	🟥 0.12	🟥 0.11
Other Exon %	**LinMLTBS**	🟩 6.84	🟩 5.77	🟩 5.7	🟩 6.62	🟩 5.86	🟩 6.47
	MACS2	🟥 1.53	🟥 1.28	🟥 1.41	🟥 1.69	🟥 1.38	🟥 1.56
1st Intron %	**LinMLTBS**	🟩 10.36	🟩 10.88	🟩 10.93	🟩 10.15	🟩 11.11	🟩 10.52
	MACS2	🟥 11.48	🟥 11.35	🟥 11.79	🟥 11.01	🟥 11.58	🟥 11.46
Other Intron %	**LinMLTBS**	🟩 24.91	🟩 26.74	🟩 27.07	🟩 25.19	🟩 26.61	🟩 26.61
	MACS2	🟥 32.23	🟥 32.1	🟥 32.3	🟥 32.02	🟥 31.78	🟥 32.42
Downstream %	**LinMLTBS**	🟥 1.05	🟥 0.9	🟥 0.83	🟥 0.83	🟩 0.95	🟥 0.91
	MACS2	🟩 1.07	🟩 0.91	🟩 0.93	🟩 1.01	🟥 0.94	🟩 0.94
Distal Intergenic %	**LinMLTBS**	🟩 42.27	🟩 43.33	🟩 42.5	🟩 40.93	🟩 41.6	🟩 40.7
	MACS2	🟥 42.59	🟥 44.81	🟥 43.69	🟥 42.28	🟥 44.42	🟥 43.33

🟩 Better Performance 🟥 Weaker Performance

References

[1] Alok Aggarwal et al. “Geometric applications of a matrix-searching algorithm”. In: *Algorithmica* 2 (Nov. 1987), pp. 195–208. DOI: `10.1007/bf01840359`.

[2] Mohamed Abd El Aziz et al. “Whale Optimization Algorithm and Moth-Flame Optimization for multilevel thresholding image segmentation”. In: *Expert Systems with Applications* 83 (Oct. 2017), pp. 242–256. DOI: `10.1016/j.eswa.2017.04.023`.

[3] Timothy Bailey et al. “Practical Guidelines for the Comprehensive Analysis of ChIP-seq Data”. In: *PLoS Computational Biology* 9 (Nov. 2013). Ed. by Fran Lewitter, e1003326. DOI: `10.1371/journal.pcbi.1003326`.

[4] Xiaoli Bao et al. “A Novel Hybrid Harris Hawks Optimization for Color Image Multilevel Thresholding Segmentation”. In: *IEEE Access* 7 (2019), pp. 76529–76546. DOI: `10.1109/ACCESS.2019.2921545`.

[5] Artem Barski et al. “Genomic location analysis by ChIP-Seq”. In: *Journal of Cellular Biochemistry* 107 (May 2009), pp. 11–18. DOI: `10.1002/jcb.22077`.

[6] Ashish K. Bhandari et al. “Modified artificial bee colony based computationally efficient multilevel thresholding for satellite image segmentation using Kapur’s, Otsu and Tsallis functions”. In: *Expert Systems with Applications* 42 (Feb. 2015), pp. 1573–1601. DOI: `10.1016/j.eswa.2014.09.049`.

[7] Ashish Kumar Bhandari. “A novel beta differential evolution algorithm-based fast multilevel thresholding for color image segmentation”. In: *Neural Computing and Applications* 32 (May 2020), pp. 4583–4613. DOI: `10.1007/s00521-018-3771-z`.

[8] Ashish Kumar Bhandari et al. “Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur’s entropy”. In: *Expert Systems with Applications* 41 (June 2014), pp. 3538–3560. DOI: `10.1016/j.eswa.2013.10.059`.

[9] Vijay Kumar Bohat et al. “A new heuristic for multilevel thresholding of images”. In: *Expert Systems with Applications* 117 (Mar. 2019), pp. 176–203. DOI: `10.1016/j.eswa.2018.08.045`.

[10] Marc Carlson. *TxDb.Hsapiens.UCSC.hg19.knownGene: Annotation package for TxDb object(s)*. R package version 3.2.2. 2015.

[11] ENCODE Project Consortium. “An integrated encyclopedia of DNA elements in the human genome”. In: *Nature* 489 (Sept. 2012), pp. 57–74. DOI: `10.1038/nature11247`.

[12] Toby Dylan Hocking et al. “Optimizing ChIP-seq peak detectors using visual labels and supervised machine learning”. In: *Bioinformatics* 33 (Oct. 2016), pp. 491–499. DOI: `10.1093/bioinformatics/btw672`.

[13] David S. Johnson et al. “Genome-Wide Mapping of in Vivo Protein-DNA Interactions”. In: *Science* 316 (June 2007), pp. 1497–1502. DOI: `10.1126/science.1141319`.

[14] Jagat N. Kapur et al. “A new method for gray-level picture thresholding using the entropy of the histogram”. In: *Computer Vision, Graphics, and Image Processing* 29 (Mar. 1985), pp. 273–285. DOI: `10.1016/0734-189x(85)90125-2`.

[15] Anastasiya Kazachenka et al. “Identification, Characterization, and Heritability of Murine Metastable Epialleles: Implications for Non-genetic Inheritance”. In: *Cell* 175 (Nov. 2018), 1259–1271.e13. DOI: `10.1016/j.cell.2018.09.043`.

[16] W. James Kent et al. “The Human Genome Browser at UCSC”. In: *Genome Research* 12 (May 2002), pp. 996–1006. DOI: `10.1101/gr.229102`.

[17] Josef Kittler et al. “Minimum error thresholding”. In: *Pattern Recognition* 19 (Jan. 1986), pp. 41–47. DOI: `10.1016/0031-3203(86)90030-0`.

[18] Pei Fen Kuan et al. “A Statistical Framework for the Analysis of ChIP-Seq Data”. In: *Journal of the American Statistical Association* 106 (Sept. 2011), pp. 891–903. DOI: `10.1198/jasa.2011.ap09706`.

[19] Gongqiang Lan et al. "Cross-Cell-Type Prediction of TF-Binding Site by Integrating Convolutional Neural Network and Adversarial Network". In: *International Journal of Molecular Sciences* 20 (July 2019), p. 3425. DOI: `10.3390/ijms20143425`.

[20] Hongnan Liang et al. "Modified Grasshopper Algorithm-Based Multilevel Thresholding for Color Image Segmentation". In: *IEEE Access* 7 (2019), pp. 11258–11295. DOI: `10.1109/access.2019.2891673`.

[21] Martin Luessi et al. "Framework for efficient optimal multilevel image thresholding". In: *Journal of Electronic Imaging* 18 (Jan. 2009), p. 013004. DOI: `10.1117/1.3073891`.

[22] Martin Luessi et al. "New Results on Efficient Optimal Multilevel Image Thresholding". In: *2006 International Conference on Image Processing.* Oct. 2006, pp. 773–776. DOI: `10.1109/ICIP.2006.312426`.

[23] Dongpin Oh et al. "CNN-Peaks: ChIP-Seq peak detection pipeline using convolutional neural networks that imitate human visual inspection". In: *Scientific Reports* 10 (May 2020). DOI: `10.1038/s41598-020-64655-4`.

[24] Nobuyuki Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9 (Jan. 1979), pp. 62–66. DOI: `10.1109/tsmc.1979.4310076`.

[25] Naim U Rashid et al. "ZINBA integrates local covariates with DNA-seq data to identify broad and narrow regions of enrichment, even within amplified genomic regions". In: *Genome Biology* 12 (2011), R67. DOI: `10.1186/gb-2011-12-7-r67`.

[26] Iman Rezaeian et al. "A new algorithm for finding enriched regions in ChIP-Seq data". In: *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine.* BCB '12. Orlando, Florida: Association for Computing Machinery, 2012, pp. 282–288. ISBN: 9781450316705. DOI: `10.1145/2382936.2382972`. URL: `https://doi.org/10.1145/2382936.2382972`.

[27] Iman Rezaeian et al. "CMT: A Constrained Multi-Level Thresholding Approach for ChIP-Seq Data Analysis". In: *PLoS ONE* 9 (Apr. 2014). Ed. by Jindan Yu, e93873. DOI: `10.1371/journal.pone.0093873`.

[28] Dario Rojas et al. "Image segmentation of biofilm structures using optimal multi-level thresholding". In: *International Journal of Data Mining and Bioinformatics* 5 (2011), p. 266. DOI: `10.1504/ijdmb.2011.040384`.

[29] Darío Rojas et al. "Biofilm Image Segmentation Using Optimal Multi-level Thresholding". In: *International Conference on Bioinformatics and Biomedicine* (Nov. 2009). DOI: `10.1109/bibm.2009.69`.

[30] Luis Rueda. "An Efficient Algorithm for Optimal Multilevel Thresholding of Irregularly Sampled Histograms". In: *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. SSPR amp; SPR '08. Orlando, Florida: Springer-Verlag, 2008, pp. 602–611. ISBN: 9783540896883. DOI: `10.1007/978-3-540-89689-0_64`. URL: `https://doi.org/10.1007/978-3-540-89689-0_64`.

[31] Luis Rueda et al. "A fully automatic gridding method for cDNA microarray images". In: *BMC Bioinformatics* 12 (Apr. 2011). DOI: `10.1186/1471-2105-12-113`.

[32] Zhen Shen et al. "Recurrent Neural Network for Predicting Transcription Factor Binding Sites". In: *Scientific Reports* 8 (Oct. 2018). DOI: `10.1038/s41598-018-33321-1`.

[33] Haipeng Xing et al. "Genome-Wide Localization of Protein-DNA Binding and Histone Modification by a Bayesian Change-Point Method with ChIP-seq Data". In: *PLoS Computational Biology* 8 (July 2012). Ed. by Ilya Ioshikhes, e1002613. DOI: `10.1371/journal.pcbi.1002613`.

[34] Guangchuang Yu et al. "ChIPseeker: an R/Bioconductor package for ChIP peak annotation, comparison and visualization". In: *Bioinformatics* 31 (Mar. 2015), pp. 2382–2383. DOI: `10.1093/bioinformatics/btv145`.

[35] Chongzhi Zang et al. "A clustering approach for identification of enriched domains from histone modification ChIP-Seq data". In: *Bioinformatics* 25 (June 2009), pp. 1952–1958. DOI: `10.1093/bioinformatics/btp340`.

[36] Yong Zhang et al. "Model-based Analysis of ChIP-Seq (MACS)". In: *Genome Biology* 9 (2008), R137. DOI: `10.1186/gb-2008-9-9-r137`.

CHAPTER 3

Conclusion and Future Work

3.1 Conclusion

We have proposed a method that introduces the use of a linear-time multi-level thresholding algorithm to identify relevant peaks corresponding to binding sites in ChIP-Seq data. The SMAWK algorithm paired with the use of indices of validity for clustering ensure that not only the thresholds obtained are optimal in the between-class sense, but also that the number of thresholds is also optimized. The use of the concurrent multi-window approach makes sure that a modern-day multi-core CPU architecture is used to it fullest potential.

When compared to CMT, the proposed method promises similar results in one sixth of the execution time. The proposed framework has been applied on a dataset of the well-known histone modification protein H3K27ac. Running tests on six randomly-chosen cell types shows that LinMLTBS outperforms MACS2. LinMLTBS has detected more regions of promoters, 5' UTR, 3' UTR and exons, while detecting less introns and intergenic regions.

3.1.1 Contributions

The contributions of this thesis can be summarized as follows:

- Proposed a very fast method for the detection of binding sites in data generated by ChIP-Seq experiments.
- Provided a mechanism to find optimal location as well as optimal number of

thresholds in large histograms in linear-time complexity.

- Envisioned and implemented a mechanism to take advantage of modern-day multi-core CPU architecture.
- Developed a Python package for the proposed method [3].
- Provided a framework for optimal and fast one-dimensional clustering of discrete points, which can be extended to real-numbered points in one dimension and applied to other fields beyond image segmentation and bioinformatics.

3.2 Future Work

This work can be further extended as follows:

- Improving the performance of the proposed method through the use of a different objective function. As the objective function is at the core of the proposed method, a more efficient objective function may improve the performance. A number of different objective functions can be studied to find the most cost efficient function. In [2], it is suggested that the use of Kittler's criterion [1] may have certain advantages.
- The proposed method relies upon the use of cluster validity indices for optimal number of thresholds. A more efficient clustering validity index that does not require the testing of all possibilities, is yet another avenue for improvement.
- This method can also be extended to the use of other next generation sequencing (NGS) and high throughput sequencing (HTS) technologies, such as RNA-Seq to identify transcriptomic features, Ribo-seq to determine which mRNA transcripts are actively being translated, among many others.

References

[1] Josef Kittler et al. “Minimum error thresholding”. In: *Pattern Recognition* 19 (Jan. 1986), pp. 41–47. DOI: `10.1016/0031-3203(86)90030-0`.

[2] Martin Luessi et al. “Framework for efficient optimal multilevel image thresholding”. In: *Journal of Electronic Imaging* 18 (Jan. 2009), p. 013004. DOI: `10.1117/1.3073891`.

[3] Musab Naik. *Github Repo: LinMLTBS*. `https://github.com/MusabNaik/LinMLTBS`. 2020.

VITA AUCTORIS

NAME: Musab Naik

PLACE OF BIRTH: Mumbai, Maharashtra, India

EDUCATION: B.E. Computer Engineering, M.H. Saboo Siddik College of Engineering, Mumbai, Maharashtra, India, 2017

M.Sc. Computer Science, University of Windsor, Windsor, Ontario, Canada, 2020