UNIVERSITY OF CALIFORNIA
RIVERSIDE

Computational Methods for Exploring Nucleosome Dynamics

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Anton Polishko

June 2014

The Dissertation of Anton Polishko is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I was very lucky to join PhD Program at University of California, Riverside. For the past five years I met a lot of awesome people and had great experiences that contributed to my personality.

I am infinitely grateful to my advisor, Stefano Lonardi, without whose help, I would not have been here. His kind and unobtrusive guidance was allowing me to have all the research freedom I needed and, at the same time, steered my focus on the most important things.

I am very thankful to Dr. Karine Le Roch for the opportunity to work on the cutting edge projects in the field of computational biology and to collaborate with great researches from Dr. Le Roch's lab: Nadia Points, Evelien Bunnik and Xueqing Lu.

My thanks goes to all the professors that greatly contributed to my training, especially to Christian Shelton, Neal Young, Stefano Lonardi, Tao Jiang, Marek Chrobak, Eamon Keogh and Michalis Faloutsos. Their enthusiasm in teaching and doing research encouraged me to pursue much higher goals than I would without them.

Chapter 2 appeared in *Bioinformatics*, in the publication "NOrMAL: accurate nucleosome positioning using a modified Gaussian mixture model.", co-authored with N. Ponts, K. G. Le Roch and S. Lonardi [54]

Portions of Chapter 3 are based on the paper "PuFFIN - A Parameter-free Method to Build Nucleosome Maps from Paired-end Reads", co-authored with E. M. Bunnik, K. Le Roch and S. Lonardi. Some of the results appeared in *BMC Genomics*, in paper "DNA-encoded nucleosome occupancy regulates transcriptional levels in the human malaria parasite Plasmodium falciparum", co-authored with E. Bunnik, J. Prudhomme, N. Ponts, S. S. Gill, S. Lonardi and K. G. Le Roch.

To all my friends and family.

Great thanks for support to my parents Ludmila and Sergey Polishko.

ABSTRACT OF THE DISSERTATION

Computational Methods for Exploring Nucleosome Dynamics

by

Anton Polishko

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2014
Dr. Stefano Lonardi , Chairperson

Nucleosomes are the basic elements of DNA chromatin structure. Not only they control DNA packaging but also play a critical role in gene regulation by allowing physical access to transcription factors. In addition to providing the positions of nucleosomes and the occupancy level it is becoming more and more important to resolve possible overlaps, extract additional information about nucleosomes like the probability of placement, and determine whether they are well-positioned or fuzzy in the sequenced cell sample.

In this dissertation, we address some of the computational issues associated with the analysis of sequencing data enriched for nucleosomes. We propose two novel algorithms to create nucleosome maps, for single- and paired-end sequencing data respectively. Then, we study the problem of aligning these maps.

The first method, called NOrMAL, is based on a novel parametric probabilistic model of a nucleosome. Expectation maximization is used to learn the parameters of a Gaussian mixture model. Extensive experiments on real and synthetic data shows that our method can produce very accurate maps, and can detect a larger number of nucleosomes than published tools.

The second method, called PuFFIN, takes advantage of paired-end short reads to build genome-wide nucleosome maps. In contrast to other approaches that require users to optimize several parameters according to their data (e.g., the maximum allowed nucleosome overlap or legal ranges for the fragment sizes) our algorithm can accurately determine a genome-wide set of non-overlapping nucleosomes without any user-defined parameter. On the real data PuFFIN detects stronger associations between nucleosome occupancy and gene expression levels compared to other tools, which indicates that our tool extracts more biologically-relevant features from the data.

Finally, we then study the problem of aligning nucleosome maps, which is NP-complete when the number of maps is three or more. We use effective bounding tricks to limit the size of the problem and use linear programming to solve it. Our evaluations on the synthetic data shows that our aligning tool consistently outperforms the naive (greedy) approach and it is faster than dynamic programming.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the central problems in molecular biology is to characterize all cellular processes controlling gene regulation. The complex interaction between DNA chromatin structure and transcription factors is one of these key processes. To study these interactions it is crucial to know the exact location of nucleosomes and how the position of nucleosomes is changing over time.

The *nucleosome* is the basic unit of chromatin structure. It is composed of DNA wrapped around a protein complex of eight histones (see Figure 1.1). The size of the DNA wrapped around the histone complex is about 146bp, and is called *nucleosomal DNA*. The portion of DNA between nucleosomes is called the *linker*. In some organisms, linkers have a typical size (i.e., for *yeast* is about 20bp), while for others (i.e., *Plasmodium falciparum*) linker regions can be of any length. Nucleosome-free regions (NFR) play an important role in transcription regulation, because they allow access to special DNA binding proteins called *transcription factors* that control the activation of gene transcription. As a consequence, nucleosome positioning not only allows DNA packaging but influences gene expression (see, e.g., [77]). Loosely speaking, the more compact is

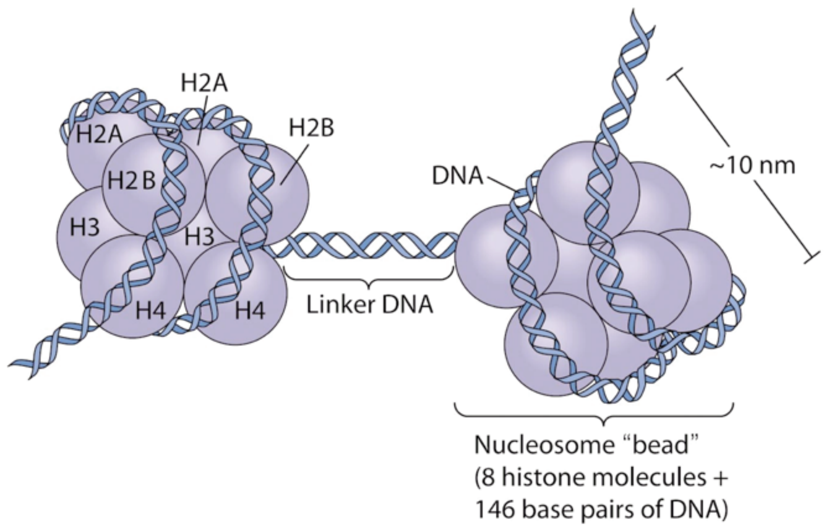

Figure 1.1: Nucleosome structure. (Resource: Pearson Education, Inc 2009)

the chromatin, the harder it is for transcription factors and other DNA binding proteins to access DNA and trigger transcription. Nucleosome positions also affects a variety of cellular and metabolic processes like centromere formation, recombination, replication, and DNA repair. Thus, to elucidate the role of interactions between chromatin and transcription factors, it is crucial to determine the location of all nucleosomes along the genome by building genome-wide nucleosome maps and to study how these maps change over time.

Several wet-lab protocols have been developed to build genome-wide nucleosome maps. For instance, one can enrich for genomic regions that are either bound to histones (typically via chromatin immuno-precipitation or ChIP) or for genomic regions that are free of nucleosomes (i.e., linkers). For instance, MAINE (MNase Assisted Isolation Nucleosomal Elements) [83] isolates the portions of the DNA that are attached to nucleosomes, because MNase preferentially digests linker regions. Then, tiling microarrays (ChIP-chip) or sequencing (ChIP-Seq/MNase-Seq) are applied to the enriched DNA. A complementary approach, called FAIRE (Formaldehyde- Assisted Isolation of

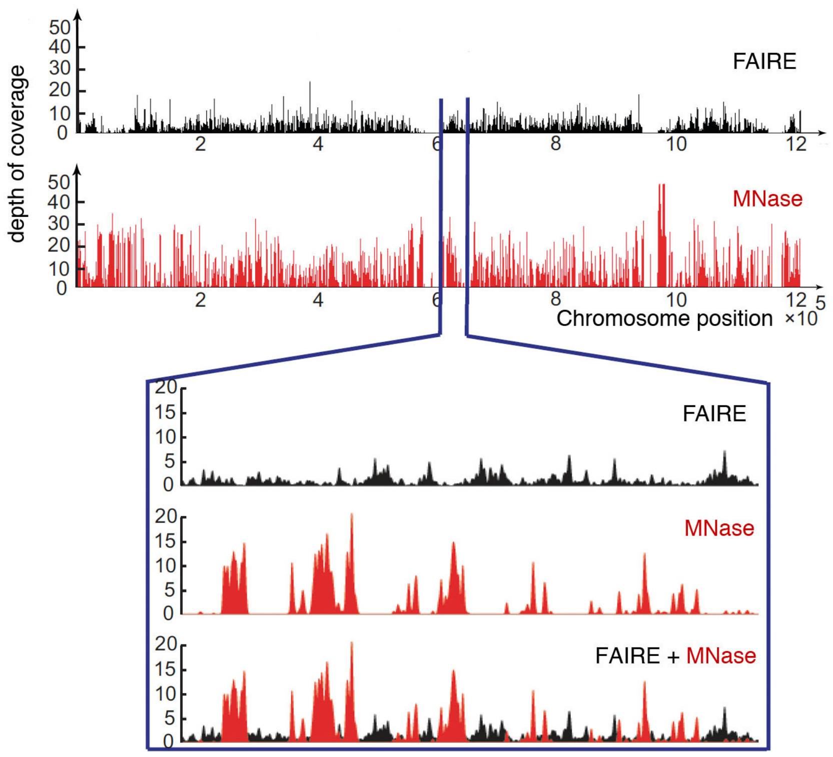

Figure 1.2: Example of combining MAINE and FAIRE raw coverage functions. Resource: [56]

Regulatory Elements) [23, 24], isolates sequences from linker DNA rather than nucleosomes. The procedure is similar to MAINE, the only difference is the resulting sequences originates from linker DNA rather than nucleosomes. We should also note that MAINE and FAIRE can be used as complementary approaches to improve the accuracy of nucleosome mapping (see Figure 1.2). Since experimental results from these complementary approaches are not always available, in this dissertation we focus on analyzing data from the more popular MAINE-Seq protocol.

We can classify computational solutions to the problem of detecting nucleosome positions depending on the type of technology used to obtain the data. The first type of technology are *tiling microarrays*. Tiling microarray technology allows one to sample specific location in the genome, i.e., the location for which a probe is represented on the chip. Due to the fact that we sampling a subset of locations, nucleosome maps obtained from microarrays are *low resolution*. For microarray data, the first challenge is to "extrapolate" the available information genome-wide. In order to achieve this generalization, different strategies have been proposed. For instance, one could build a probability model based on the available data. Then, one can either use a thermodynamic model [67], or hidden Markov Model (HMM) [81] or a HMM in combination with wavelet transformation [80] to place the nucleosomes. These approaches have good accuracy but are limited only to nucleosomes that are strongly positioned (i.e., they have a strong signal).

When the cost of high-throughput sequencing dropped significantly, it quickly became the main tool for studying chromatin structure. Enriched DNA products obtained from MAINE and FAIRE protocols can be sequenced and corresponding reads can be mapped to the reference genome. Because reads can potentially map to any location in the genome, nucleosome maps obtained from MAINE-Seq and FAIRE-Seq are high-resolution (i.e., single base pair resolution). Nowadays high-resolution analysis has became the main tool for study chromatin structure in a large number of organisms (see, e.g., [1, 19, 20, 35, 45, 46, 64, 74, 77, 1, 69, 46, 55, 20, 88, 85, 66]). These latter methods generally outperform low-resolution approaches and potentially can detect nucleosome dynamics [65].

In this dissertation, we assume that the sequencing data is either MNase-Seq or ChIP-Seq, which are currently the most popular approaches to study nucleosome and histone modifications. More specifically, we describe algorithmic methods to analyze single- and paired-end sequencing data from MNase-Seq.

In Chapter 2, we describe a novel tool to build nucleosome maps, called NOrMAL, that uses a modified Gaussian mixture model. NOrMAL was specifically designed to deal with the single-end reads. It uses *Expectation Maximization* (EM) to infer the size of nucleosome enriched fragments to improve the accuracy of the positioning.

In Chapter 3, we present another method, called PuFFIN, which is a parameter-free method to build nucleosome maps from paired-end reads. PuFFIN exploits the length information available for paired-end data to avoid the inference of the size of nucleosome enriched fragments. Our approach uses a multi-scale approach for the analysis of the data, which eliminates the need to specify parameters.

In Chapter 4, we discuss the problem of studying dynamics of nucleosomes. We formulate the problem of aligning maps of genomic features (i.e., nucleosomes) as a multi-target tracking problem and present a novel tool, called ThIEF, to track nucleosomes across time points.

Chapter 2

NOrMAL: Accurate Nucleosome Positioning using a Modified Gaussian Mixture Model

In this Chapter we concentrate on the analysis of single-end sequencing data, given the prevalence of MNase/ChiP-Seq experiments in the recent literature. The computational analysis of sequencing data usually consists of two steps: (1) a nucleosome occupancy coverage is computed from the process of mapping nucleosome-enriched sequenced reads to a reference genome, followed by some normalization steps; (2) nucleosomes are placed according to the peaks of the coverage profile.

Approaches based on *peak-calling* are computationally fast and quite accurate in resolving isolated (*stable*) nucleosomes, however they are not entirely reliable when more complex nucleosome configurations are present. Observe that while it is physically impossible for two nucleosome to be "overlapping" on the same location on a DNA strand, it is quite common that the population of cells from which the enriched DNA

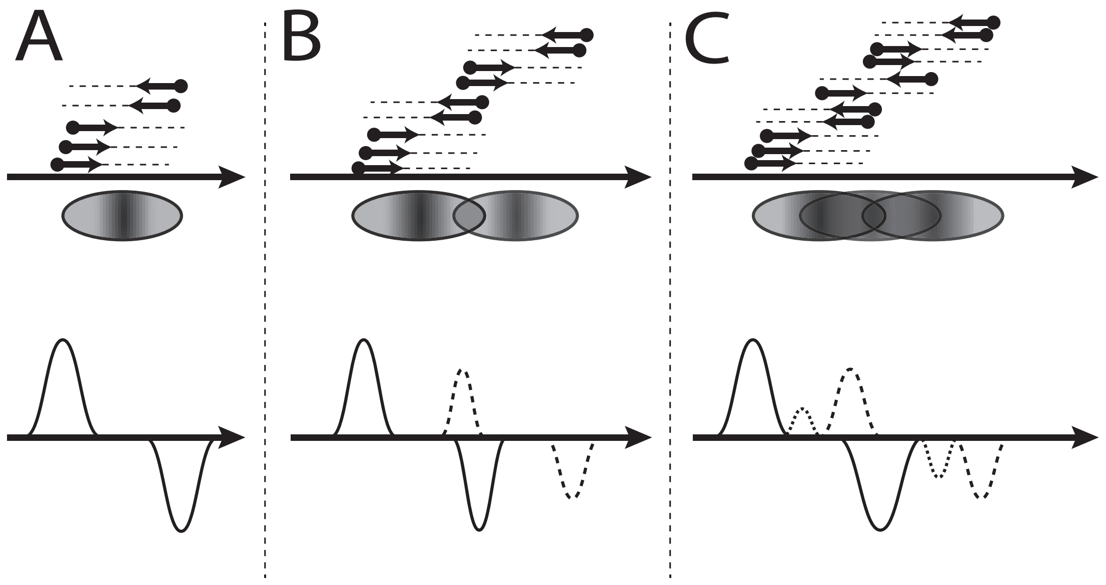

Figure 2.1: Nucleosomes are represented by ovals, mapped reads by arrows (which correspond to 5'→3' prefixes of nucleosome-bound DNA). Coverage profiles are represented as time series for forward (*top*) and reverse (*bottom*) strands: the line style (solid, dotted, dashed) indicates peaks originating from distinct nucleosomes. **A** represents a stable nucleosome; **B** illustrates overlapping nucleosomes; **C** represents "fuzzy" nucleosomes.

was obtained had nucleosomes slightly "off-sync" at a given genomic coordinate. As a consequence, the resulting coverage profile will exhibit a "blurring" of the peaks.

Molecular biologists distinguish the case of "overlapping" nucleosome from "fuzzy" nucleosomes or "fuzzy" regions (see Figure 2.1). For overlapping nucleosomes, the overlap is relatively small; in the "fuzzy" case, several nucleosomes are mutually overlapping for a significant fraction of their size. This definition can be made precise by introducing a user-defined threshold parameter on the allowed overlap.

Another shortcoming of peak-calling approaches is that they can only report nucleosome positions and/or occupancy level, but molecular biologists need additional information about nucleosomes. For instance, they are interested in the level of "fuzziness" in certain genomic locations with respect to coding regions (i.e., well-positioned for all the cells in the sample, or "blurred"), how strong is the binding between nucleosomes and DNA, etc. To address these shortcomings we propose a method that determines the accurate position of nucleosomes independently from the amount of overlaps in the nucleosomes and that can extract other important statistics about nucleosomes, e.g., the probability that a nucleosome is actually present, a measure of nucleosome "fuzziness", and the expected size of DNA fragments enriched for nucleosomes.

Here we propose a parametric probabilistic model for nucleosome positioning, which we called NOrMAL (NucleOsome Mapping ALgorithm). NOrMAL uses Expectation Maximization to infer its parameters. To demonstrate the performance of our method, we report experimental results on MAINE-Seq data for *Plasmodium falciparum* [56], and *Saccharomyces cerevisiae* [77]. We compare the performance of our method against the Template Filtering algorithm [77], which is considered the current state-of-the-art in terms of accuracy and ability to estimate sizes of the DNA fragments bound to nucleosomes. We also discuss a fundamental limitation of greedy peak-calling approaches in the case of overlapping nucleosomes and how our method addresses this issue.

2.1 Previous work

Several landmark studies have been published in the last few years on the chromatin structure of model organisms based on the analysis of genome-wide nucleosome maps (see, e.g., [1, 69, 74, 46, 55, 20, 45, 88]). Existing methods in the literature are based on the analysis of the peaks in the nucleosome occupancy coverages estimated by mapping nucleosome-enriched reads to the reference genome. The *coverage occupancy profile* is an integer-valued function defined for all genomic locations: given a position i in a chromosome the function is equal to the number of sequenced reads that are mapped to location i. From a probabilistic point of view, the coverage profile represents a non-parametric distribution of nucleosome positions. At the time of writing, the length of the reads obtained by second-generation sequencing (*e.g.*, Illumina Genome Analyzer) are limited to about 100 bases and the sequencing occurs in the 5'$\rightarrow$3' direction. In the case of ChIP-Seq/MAINE-Seq, sequenced reads that can be uniquely mapped to the positive strand originate from the left boundary of nucleosome DNA fragments, while reads uniquely mapped to the negative strand originate from the right boundary (see Figure 2.1). Recall that nucleosomes are composed of about 146bp of DNA, so if reads are single-end and shorter than 146 bases, we expect to observe a peak in the forward and a peak in the reverse coverage profiles at a distance consistent with the nucleosome size.

The problem of associating a peak in the forward strand with the correct peak in the negative strand can be difficult in the case of a large number of complex nucleosome configurations. Some authors artificially extend the reads in the 5'$\rightarrow$3' direction or they shift the positions of the mapped read position of the forward and reverse towards the middle of potential nucleosomes. Then, they combine (e.g., sum) the forward and reverse modified coverages to build a score function. In both cases, they need to determine the amount of the extension or the size of the shift. In the former case, the extension should account for the expected length of the DNA fragments enriched for nucleosomes; in the latter the shift should be about half of the DNA fragment size. The problem of this approach is that no extension or shift that will work equally well for all nucleosomes in the genome. While one should expect DNA fragments enriched for nucleosomes to be about 146 bp, the reality is that the digestion process can either leave non-nucleosome-bound DNA in the sample or "over-digest" the ends of nucleosome-bound DNA. What complicates the matter further is that the rate of digestion is sequence-dependent, so nucleosomes in different genomic locations will end up with different DNA fragment size. For this reason, it is advantageous to "learn" this information from the input data. Template Filtering [77] is one of the first methods that can handle variable fragment sizes in a specified range, whereas other methods require users to decide this value in advance.

As said, a variety of peak-calling algorithms have been also developed (see, e.g., [1, 19, 20, 35, 45, 46, 64, 74]). Most of these methods have been proposed for the analysis of ChIP-chip or ChIP-Seq data to determine the position and strength of transcription factor binding to DNA. The problem of detecting transcription factor binding sites is similar to nucleosome positioning: in both cases we need to infer position of proteins binding to DNA from the coverage profiles. However the size of nucleosomes is significantly bigger than transcription factor binding sites, as a consequence the resulting configurations of nucleosomes can be more complex.

To summarize our experience with existing methods on the genome-wide nucleosome study of human malaria parasite [55, 56], peak-calling approaches suffer from a variety of problems. First, the coverage profile function has to be cleaned of high-frequency noise, typically via a kernel density estimation method [53]. The type of kernel and the amount of smoothing can drastically affect the results: too much can merge adjacent peaks, too little can leave too many noisy artifacts that can be interpreted as individual peaks. Second, peak finding algorithms have parameters (like the extension and the shift discussed above) that are difficult to optimize: a set of parameter can work for a region of a chromosome but not for another. Third, peak-calling do not properly resolve overlapping nucleosomes. For instance, Template Filtering [77] uses a greedy strategy: nucleosomes are placed according to the "best" matching peaks in the score function. Once these strong-positioned nucleosome are assigned, Template

Filtering ignores any nucleosome that overlaps with previous ones. It is relatively easy to show that for overlapping nucleosomes the greedy strategy does not always return the best overall placement (see Section 2.3 for details).

2.2 Methods

Next we propose a parametric probabilistic model to find the most likely set of nucleosome that best "explain" the mapped reads. We cast this problem in a *modified Gaussian mixture model* framework. The problem of positioning nucleosomes is then reduced to the problem of learning the parameters of the model and finding the distribution of mixture components, which is achieved via Expectation Maximization.

2.2.1 A Probabilistic Model for Nucleosomes

We employ a probabilistic model for nucleosome positioning that is described by a set of hidden and observed variables. We use N to denote the number of DNA fragments obtained after MNase digestion. For any DNA fragment $i \in [1, N]$, let x_i be the starting position of the 5' end of fragment i (obtained by mapping a corresponding sequenced read), and let variable $d_i \in \{+1, -1\}$ be the strand on which fragment i was mapped (+1 for the positive strand, and −1 for the negative strand). Also, let z_i be the length of fragment i. If we use variable m_i to denote the position of the center of fragment i, then we have $m_i = x_i + (d_i z_i)/2$.

We denote with X_i, D_i, Z_i and M_i the random variables associated with variables x_i, d_i, z_i and m_i, respectively. Since the sequencing process is 5'→3', the value of X_i is observable by means of mapping a read originating from fragment i. Similarly, the strand variable D_i is also observable. Variables Z_i and M_i can be observed directly only if sequencing produces paired-end reads, otherwise these variables are hidden. In order to consider the most general case, we only deal with the latter case (single-end reads).

We assume for the time being that the number K of nucleosomes is given. We will discuss how to choose K in Section 2.2.3. For each DNA fragment i, we use a hidden variable $C_i \in [1, K]$ to represent which nucleosome it belongs. Each nucleosome $j \in [1, K]$ is described by a set of six variables $(\mu_j, \sigma_j, \Delta_j, \delta_j^{+1}, \delta_j^{-1}, \pi_j)$, where μ_j denotes the center position of nucleosome j, σ_j is the *fuzziness* associated with the position of nucleosome j, Δ_j describes the length of DNA fragments associated with nucleosome j, δ_j^{+1} and δ_j^{-1} represent the variation on fragment sizes for positive and negative strands respectively, and π_j is the probability of nucleosome j. The degree of *fuzziness* captures the variation of the position of a particular nucleosome in the population of sampled cells. *Well-positioned* nucleosomes have very low degree of fuzziness. We introduce two variables δ_j^{+1} and δ_j^{-1} to model the variation of the fragment size because MNase does not only digest nucleosome-free DNA. Given enough time, it can also digest the ends of the fragments bounds to nucleosomes, but the rate of digestion is sequence-dependent (see, e.g., [77]). Since the sequence composition of the 5' end of a DNA fragment can be quite different from the 3' end, we need to have two different variables. The value of C_i is drawn from $(1, 2, \ldots, K)$ with corresponding probabilities $(\pi_1, \pi_2, \ldots, \pi_K)$. Parameter π_j models the contribution of j-th nucleosome to the occupancy level, i.e., what portion of the mapped reads belong to nucleosome j.

Our nucleosome model assumes that our random variables are distributed according to a normal distribution. For convenience of notation, we set $\Theta_j = (\mu_j, \sigma_j, \Delta_j, \delta_j^{+1}, \delta_j^{-1}, \pi_j)$ for all $j \in [1, K]$, and $\Theta = (\Theta_1, \ldots, \Theta_K)$. First, we assume that variable M_i associated with the center of fragment i for a particular nucleosome j is distributed

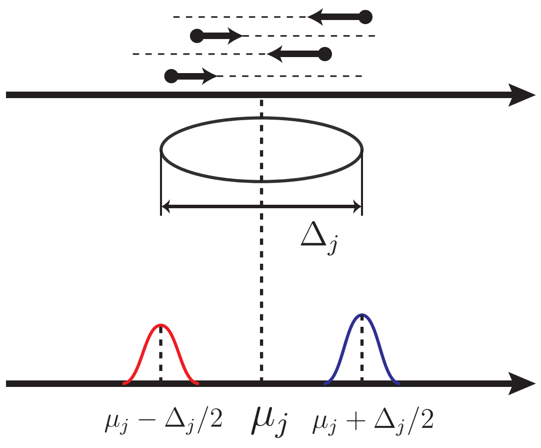

Figure 2.2: Proposed probabilistic model for a single nucleosome

as follows

$$P(M_i|C_i = j, \Theta) \sim N(\mu_j, \sigma_j^2) \tag{2.1}$$

where μ_j represents the center of nucleosome j and σ_j is its fuzziness. Second, we assume that the length Z_i of fragment i for a particular nucleosome j is distributed as follows

$$P(Z_i|D_i = d_i, C_i = j, \Theta) \sim N(\Delta_j, (\delta_j^{d_i})^2) \tag{2.2}$$

where Δ_j represents the expected size of the fragments for nucleosome j, and δ_j^{+1} and δ_j^{-1} represent the variation of fragment sizes for positive and negative strands, respectively.

Combining Equations (2.1) and (2.2) and relation $x_i = m_i - (d_i z_i)/2$, and then applying the rule of linear combination of independent Gaussians we obtain

$$P(X_i|D_i{=}d_i, C_i{=}j, \Theta){\sim}N\Big(\mu_j - (d_i\Delta_j)/2, \sigma_j^2 + (\delta_j^{d_i}/2)^2\Big) \tag{2.3}$$

Equation (2.3) allows one to compute the probability of a given data point x_i given the parameters of a nucleosome.

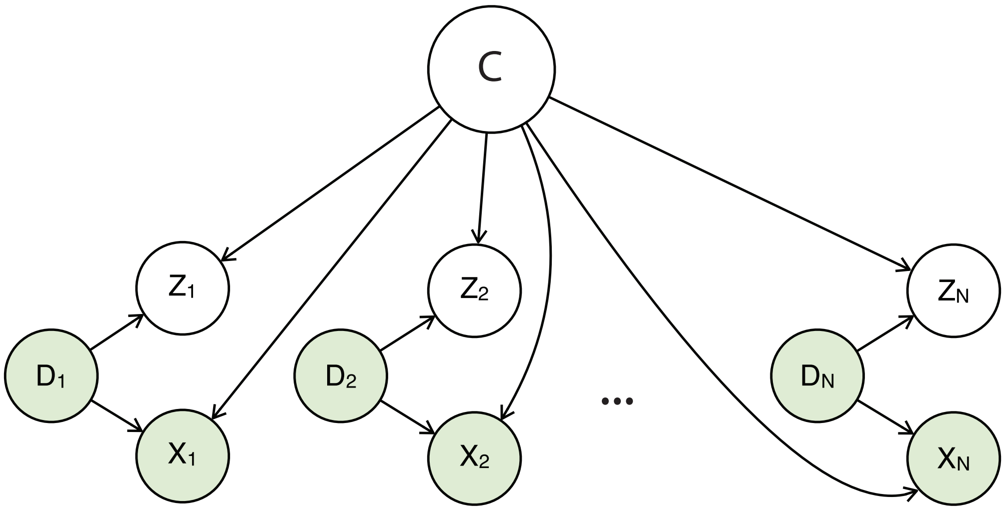

Figure 2.3: The proposed graphical mixture model: shaded nodes correspond to observed variables, white nodes correspond to hidden variables

Figure 2.2 illustrates our proposed model for a single nucleosome model. On the top, we show the location and the direction of the reads that belong to a nucleosome at location μ_j of size Δ_j. On the bottom, we drew the corresponding Gaussian distributions (Equation (2.3)) that model the forward and reverse reads (red and blue curves, respectively).

Next we describe the model for multiple nucleosomes.

2.2.2 Mixture model

Next, we introduce a generative mixture model to describe the likelihood of input data points $X = (x_1, \dots, x_N)$. Figure 2.3 shows a graphical representation of the mixture model. In Equation (2.3) the only hidden random variable is C because we already excluded variables z_i from the computation. By grouping variables X_i, D_i we can use an approach similar to a *naive Bayes classifier*. Variable C represents the nucleosome to which the points belong. Thus, we can describe the likelihood of point (x_i, d_i) given the parameters of our model as a mixture of distributions. Using the

Bayesian rule we obtain

$$
\begin{aligned}
P(X_i|D_i{=}d_i,\Theta) &= \sum_{j=1}^{K} P(C_i=j,\Theta)P(X_i|D_i=d_i,C_i=j,\Theta)\\
&= \sum_{j=1}^{K} \pi_j f\Big(x_i,\mu_j-d_i\Delta_j/2,\sigma_j^2+(\delta_j^{d_i}/2)^2\Big)
\end{aligned}
\tag{2.4}
$$

where $f(x,a,b)=\frac{1}{\sqrt{2\pi b}}e^{-\frac{(x-a)^2}{2b}}$ is the Gaussian density function.

Using Equation (2.4) we can obtain the log-likelihood of observed data points X given parameters Θ as

$$
\begin{aligned}
l(X|\Theta) &= \sum_{i=1}^{N} \log P(X_i=x_i|D_i=d_i,\Theta)\\
&= \sum_{i=1}^{N} \log\left[\sum_{j=1}^{K} P(C_i{=}j,\Theta)P(X_i|D_i{=}d_i,C_i=j,\Theta)\right]\\
&= \sum_{i=1}^{N} \log\left[\sum_{j=1}^{K} \pi_j f\Big(x_i,\mu_j-d_i\Delta_j/2,\sigma_j^2+(\delta_j^{(d)}/2)^2\Big)\right]
\end{aligned}
\tag{2.5}
$$

Given Equation (2.5) and input data points $X=(x_1,\ldots,x_N)$ we can find an estimate the parameters of the model Θ via maximum likelihood

$$
\widehat{\Theta} = argmax_{\Theta} l(X|\Theta) \tag{2.6}
$$

Recall that $\Theta=(\Theta_1,\ldots,\Theta_K)$ is a vector whose components are the nucleosome parameters $\Theta_j=(\mu_j,\Delta_j,\sigma_j,\delta_j^{(+1)},\delta_j^{(-1)},\pi_j)$ for all $j\in[1,K]$. The presence of parameters π_j that correspond to hidden variables C_i prevents us from solving Equation (2.6) directly. We estimate $\widehat{\Theta}$ via Expectation Maximization (EM). In our case, the E step requires computing the posterior probabilities $P(C_i=j|X_i=x_i;\Theta)$ of data points $x_i, i\in[1,N]$ with respect to the distribution of C_i given the current estimate of parameters $\Theta^{(t)}$

$$Q(\Theta|\Theta^{(t)}) = E_{C|X,\Theta^{(t)}} l(X|\Theta) \tag{2.7}$$

During the E step we supplement the missing data in Equation (2.6) with the expected values under the current parameter estimates $\Theta^{(t)}$. In the M step, we find new parameter estimation $\Theta^{(t+1)}$ by maximizing Equation (2.7)

$$\Theta^{(t+1)} = argmax_{\Theta} Q(\Theta|\Theta^{(t)}) \tag{2.8}$$

It is relatively straightforward to bound the variation parameters $(\sigma, \delta^{+1}, \delta^{-1})$ during the iterative EM process to converge to a solution with "reasonable" parameters. We can also easily introduce prior distribution for some of the parameters. For instance, we can specify an expected distribution for DNA fragment sizes Δ_j, which can be obtained via gel electrophoresis prior to the sequencing experiment.

2.2.3 Choosing the number of nucleosomes

The method described above assumes that the number of clusters K is known. The problem of selecting the best value for K is as challenging as selecting the optimal number of clusters in k-means clustering. One can estimate the number of clusters by looking at the support area of the occupancy coverage, but this will be quite inaccurate because "fuzzy" nucleosomes correspond to wider peaks, and the support area is bigger for them.

Here we propose a simple but effective heuristic to find K. We start by (1) placing the maximum possible number of non-overlapping nucleosomes uniformly dis-

```
**Data**: a set of input data points $X$
**Result**: a set of model parameters $\Theta = (\Theta_1, ..., \Theta_{k_{res}})$
$\mu_0 \leftarrow (\mu_1, \mu_2, \ldots, \mu_K)$ , where $\mu_i$ is uniformly distributed
$\pi_0 \leftarrow (\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}) \in R^K$
$t \leftarrow 0, \Theta^{(t)} \leftarrow (\mu_0, \sigma_0, \Delta_0, \delta_0^{+1}, \delta_0^{-1}, \pi_0)$
*Soft Learning*
**repeat**
    **while** *not converged* **do**
        $Q(\Theta|\Theta^{(t)}) \leftarrow E_{C|X,\Theta^{(t)}} l(X|\Theta)$
        $\Theta^{(t+1)} \leftarrow argmax_\Theta Q(\Theta|\Theta^{(t)})$
        $t \leftarrow t + 1$
    **end**
    **for** $\forall j \in [1, K-1]$ **do**
        **if** $|\mu_j - \mu_{j+1}| \leq$ threshold **then**
            Merge clusters $i$ and $i+1$
        **end**
    **end**
**until** *no clusters were merged*;
*Hard learning*
**for** $\forall i \in [1, N]$ **do**
    $C_i \leftarrow argmax_j(T_{ij})$
**end**
Recompute cluster parameters $\Theta$
return $\Theta$
```

Figure 2.4: Sketch of the proposed NORMAL algorithm

tributed on the chromosome, that is K = (size of the chromosome)/(expected size of a nucleosome), where the expected size of nucleosomes is underestimated. Then, (2) we run our EM algorithm until convergence ("soft learning"). We will then (3) check the distance between the clusters, and merge those that have too much overlap (above a specified threshold). In case of multiple overlaps for a nucleosome, we merge it with the closest one. We repeat (2) and (3) until no additional clusters are merged. After a few cycles we will obtain a set of non-overlapping clusters that best explain the given data points. Overlapping nucleosomes are merged into new ones and then the position of new nucleosomes are learned from the data.

This procedure will give us a good estimate on the number of clusters and a rough estimate of nucleosome positions as well. To further improve accuracy for other model parameters we perform one iteration of "hard learning" by assigning each data point x_i its maximum probable cluster. Nucleosome clusters will partition the set of input points, which in turns will allow us to compute their parameters more accurately. The pseudo-code of the algorithm is shown on Figure 2.2.2. The running time of NORMAL is dominated by the running time of *Learning* step (Algorithm 2.2.2, line 6). Observe that the probability that a data point belongs to far-away nucleosomes is close to zero, so one can avoid unnecessary computations by computing updates only for clusters in close vicinity of each point.

The bottleneck in the running time is the heuristics to find the number of nucleosomes K. In order for the algorithm to scale to eukaryotic genomes additional optimization steps will have to be implemented. During the early stage of soft-learning (i.e., active cluster merging) the algorithm could be applied to small "chunks" of chromosomes. Then, when the number of merges reduces substantially, the nucleosome maps for each chunk can be combined and algorithm continues to the hard-learning step.

2.2.4 Practical considerations

Our method requires users to specify three parameters, namely the threshold for allowed overlap between adjacent nucleosomes, the prior Δ on nucleosome sizes and the prior λ on nucleosome weights.

The threshold for allowed overlap can significantly affect the output: the more overlap is allowed the more nucleosomes can be placed. This parameter has to be specified by the user, and cannot be inferred from the data. The prior Δ on the nucleosome size and the prior λ on the nucleosome weight control the propagation of "knowledge" from data points on forward strand to data points on the reverse strand, and vice versa. Based on our experience if the prior size Δ is within 30bp of the "true" fragment size, then the algorithm is consistent in its output.

Our implementation has some additional internal parameters that we are not expecting users to change. While inferring the parameters of our mixture models some clusters will tend to cover most of the data points using large variances: a common trick to avoid this from happening is to introduce hard limits on such parameters. Our implementation has range limits for the nucleosome sizes and variance to force the method to converge to "reasonable" nucleosome sizes/variances in the early stage of the iterative process. These parameters have been chosen loose enough so by the end of the iterative process the limits for nucleosome size and variance are rarely hit, and the output is not significantly affected. The hard learning step completely ignores those upper limits.

2.3 Experimental Results

We carried out extensive benchmarking between our proposed method NORMAL and TEMPLATE FILTERING (TF) [77]. We selected TF because it is considered the current state-of-the-art. It is the only method that in addition to nucleosome positions can extract nucleosome fragment sizes and binding scores. TF differs from the traditional peak-calling algorithms because it does not look for peaks in the coverage profiles, but it places nucleosomes at the peaks of a correlation score matrix. Due to its greedy strategy, TF has significant limitations when dealing with overlapping nucleosome, as explained next.

The setup for the comparison is as follows. The input parameters for NORMAL are the prior size of the nucleosome fragments and the allowed amount of overlap between nucleosomes. For TF we used default parameters, unless specified otherwise.

Parameter	True value	TEMPLATE FILTERING	NORMAL
μ_1	210	208	206
Δ_1	130	125	134
μ_2	300	301	300
Δ_2	150	149	148

Table 2.1: The parameters used to generate the reads in Figure 2.5, and the corresponding output results from TEMPLATE FILTERING and NORMAL

The default allowed range of nucleosome size for TF is [100,200], which centered around the expected nucleosomes size of about 146bp.

Synthetic Data. First, we want to illustrate the challenge for existing nucleosome positioning methods to deal with the placement of overlapping nucleosomes. The problem derives from the difficulty in distinguishing two overlapping nucleosomes from the "fuzzy" case. To define precisely this problem we need to introduce a threshold parameter: when the percentage of overlap between two nucleosomes exceeds the threshold they should be considered "fuzzy", otherwise they should be treated as separate overlapping nucleosomes. It is relatively easy to show that the greedy strategy does not always give the optimal nucleosome placement in case of overlapping nucleosomes. To do so, we have created a small synthetic dataset that contains reads corresponding to two overlapping nucleosomes. Although our own parametric model could be used to generate the synthetic data, we have employed a different approach to avoid the possibility of giving an advantage to our method. We generated the input data according to the template function described in [77]. The parameters for nucleosome positions (μ_1 and μ_2) and nucleosome sizes (Δ_1 and Δ_2) that we used to generate the reads are reported in Table 2.1. Figure 2.5-A illustrates the coverage profile for mapped reads. Observe that there are two peaks on forward and reverse strands, which indicates the presence of two nucleosomes. The percentage of overlap is roughly 35%. In the first case we allowed such amount of overlap in both TF and NOrMAL (Figure 2.5-B for TF and Figure 2.5-C for NOrMAL). Both methods correctly reported two overlapping nucleosomes. Observe the error-bars attached to the boundaries of nucleosomes reported by NOrMAL that indicate the positional variance of the corresponding boundary (each bar has length $3\sqrt{\sigma_i^2 + {\delta_i^{d_i}}^2}$). TF does not provide such information.

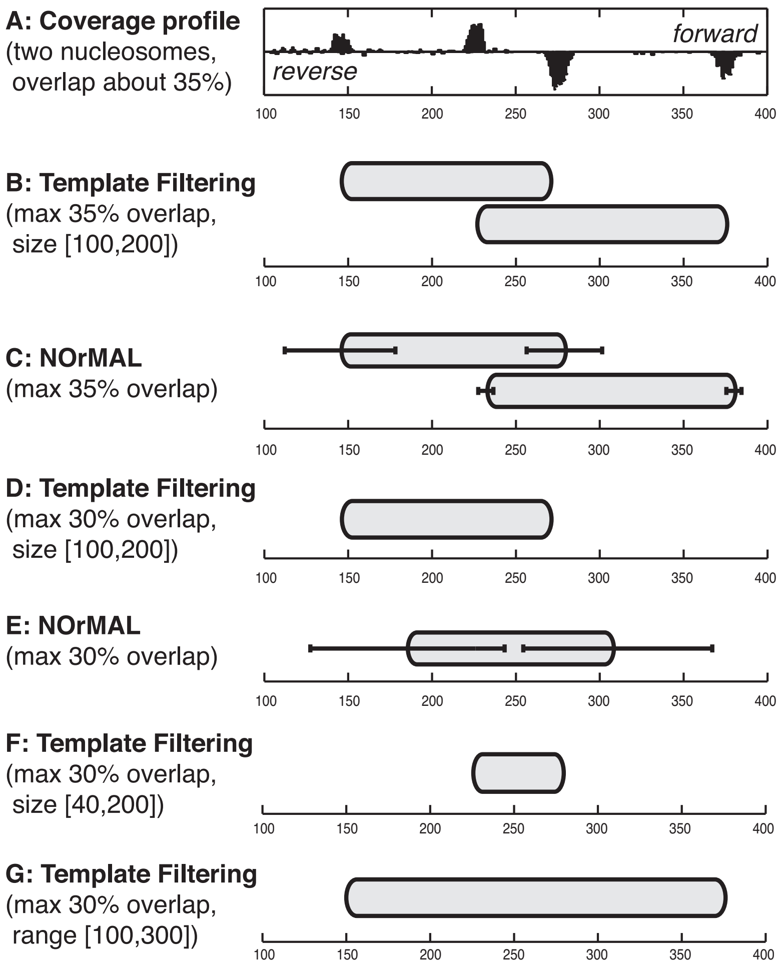

Figure 2.5: An example of two overlapping nucleosomes. **(A)** Coverage profile from synthetic data (forward on strand top, reverse strand on bottom), where nucleosomes overlap about 35%; nucleosomes detected using TEMPLATE FILTERING allowing overlap (**B**), NOrMAL allowing overlap (**C**), TEMPLATE FILTERING disallowing overlap (**D**), and NOrMAL disallowing overlap (**E**); nucleosome reported by TEMPLATE FILTERING with nucleosome size range [40,200] (**F**) and [100,300] (**G**)

In the second case, when the parameters are set so nucleosomes are not allowed to overlap more than 30%, only one nucleosome should be reported. Figure 2.5-D and

Figure 2.5-E illustrates the output of TF and NORMAL, respectively. Observe that now there is a fundamental difference: TF's greedy strategy reports the presence of the first nucleosome, but then it completely ignores the data corresponding to the second nucleosome. This is an entirely arbitrarily choice and the user won't be even aware of this. In contrast, NORMAL reports one nucleosome positioned near the centroid of the data points and correctly indicates that the variance of the nucleosome boundaries in this case is very high, indicating that this nucleosome should be considered "fuzzy".

In addition, TF's positioning results can be very sensitive to its main input parameter, namely the allowed range for nucleosome fragment sizes. With the default size range [100,200], the output of TF for the input in Figure 2.5 is shown in Figure 2.5-D. When we change the range to [40,200], TF's output is shown on Figure 2.5-E. If we extend the range to [100,300] the output is shown in Figure 2.5-G. Even if we allow a larger overlap, TF will produce the output shown in Figure 2.5-G. These results are not intended to prove that TF is flawed, but to warn users that the range parameter has be chosen carefully to produce good results. NORMAL can also produce unsatisfactory results if the prior size specified is very far from the true size. The main difference, however, is that range for TF is a hard boundary, while the prior distribution for the fragment sizes in NORMAL is "soft" and it will adapt to the data.

Real data. The challenge for nucleosome position inference is that the true positions of nucleosomes are unknown. The lack of a "ground-truth" makes it very hard to benchmark existing computational methods. To compare between methods we can only use conservative indicators. We argue that a valuable indicator is the number of reported nucleosomes, however it is difficult to argue about performance in objective terms. That is why the first dataset we considered is from *S. cerevisiae* [77]. This was the original dataset for which TF was designed. The results of TF on this dataset are assumed to be accurate.

To compare NORMAL and TF we used the following setup. The main range parameter for TF was set to $[80, 200]$, which is slightly wider than the default parameters. We did not want to penalize TF, since NORMAL does not have any hard limits for the nucleosome sizes. For NORMAL the main parameter is the prior expected value for the nucleosome sizes: we used 140bp to hit the middle of the specified range of TF. The threshold value of allowed overlap for both methods was set at 35%. All other parameters were left to default values. The results of both methods are reported in Table 2.2. Observe that NORMAL is slower, but it returns on average 2.6% more nucleosomes than TF.

If we compare the distribution of reported nucleosome sizes (see Figure 2.6), both method provide consistent results. Note that while the prior size for NORMAL was set to 140bp, but the model learned a new value for the data. To compare how reported nucleosomes are related to each other we performed a matching procedure. We built a bipartite graph, where a node corresponds to a reported nucleosome (each part corresponds to one of the two methods). The bipartite graph is fully connected, and the weight on edge (u, v) is the squared distance between nucleosome u and v: when the distance between u and v exceeded 50bp, we set the weight to ∞. Then we solved the *weighted assignment* problem between the two sets using the Hungarian method. The distribution of pairwise distances between matching nucleosomes is shown in Figure 2.7. Observe that the distribution is a unimodal bell-shaped curve with mean and mode having near zero value. The number of matched (common) nucleosomes is 81.44% of

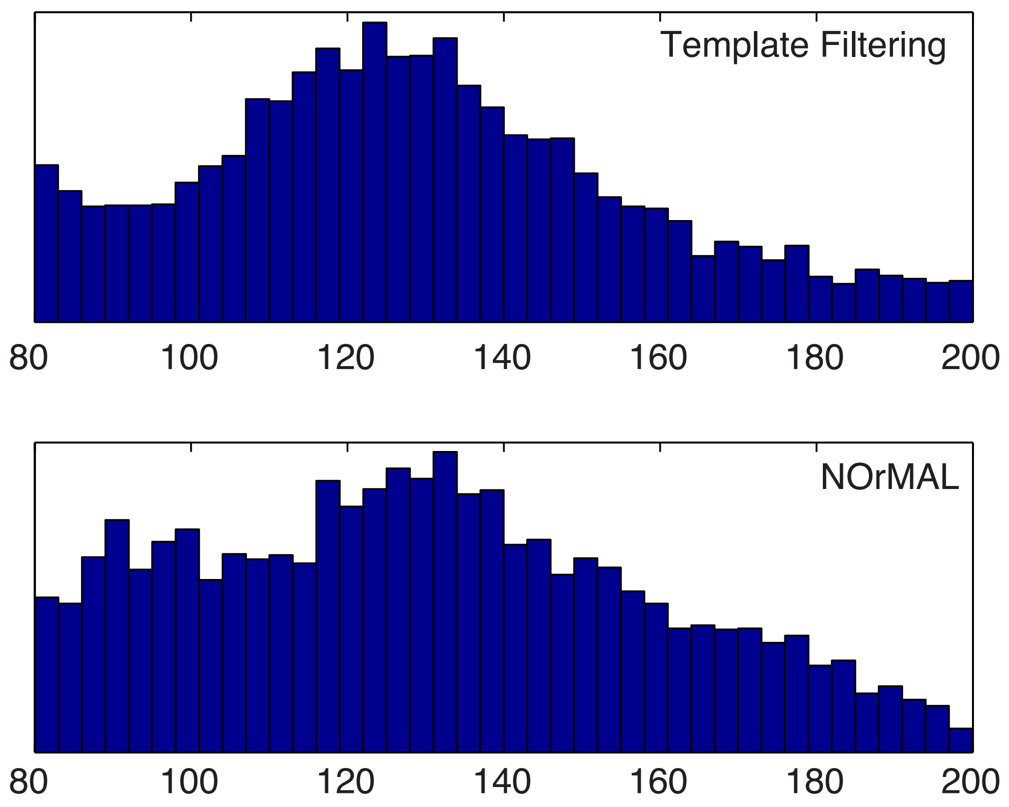

Figure 2.6: Size distribution of reported nucleosomes for *S. cerevisiae*

the total: 8.29% and 10.27% are unique to NORMAL and TF, respectively.

While the dataset for *S. cerevisiae* is considered to have relatively stable set of nucleosomes [77], the dataset for the human malaria parasite *P. falciparum* has very dynamic nucleosomes [55]. The considered dataset consists of seven time-points (namely, 0, 6, 12, 18, 24, 30, 36 hours), each related to a different stage on the cell cycle [61]. The experiment assumes that cells are "synchronized" at each time-point, but obviously the synchronization is not perfect. As a consequence, we expect a large number to nucleosomes to exhibit a "fuzzy" behavior.

First, we performed nucleosome placement with the same setup as with the yeast dataset. The distribution of fragment sizes is quite different in this case (see Figure 2.9-TOP). NORMAL reports fragment sizes with a mean value of 105bp and

Chr	# reads	TEMPLATE FILTERING	Time (*sec*)	NORMAL	Time (*sec*)
1	16,688	1,033	1.38	**1,078**	6.86
2	78,543	4,284	7.37	**4,394**	84.56
3	30,589	1,583	4.43	**1,618**	8.49
4	138,801	7,975	16.36	**8,014**	369.11
5	55,601	2,986	4.02	**3,101**	38.80
6	26,141	1,403	1.63	**1,453**	4.45
7	101,981	5,727	9.84	**5,817**	126.34

Table 2.2: Experimental results on the *S. cerevisiae* dataset: number of nucleosome detected by TEMPLATE FILTERING and NORMAL and corresponding execution time

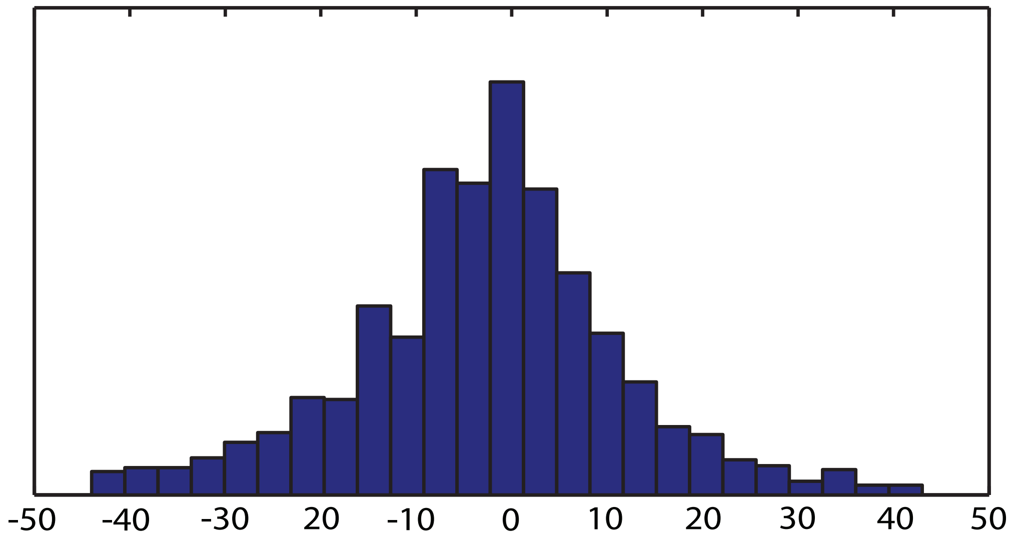

Figure 2.7: Distribution of pairwise distances between corresponding nucleosomes reported by TF and NORMAL for *S. cerevisiae*

mode value of about 120bp, whereas TF reports a distribution with mean and mode of about 84bp. If we perform the matching of the reported nucleosomes the distribution of distances is much wider than for yeast (see Figure 2.8). Now only 50% of all detected nucleosomes are in common between two methods. A total of 32,38% and 17,62% are unique to NORMAL and TF, respectively. As expected, the disagreement between NORMAL and TF is much higher on this dataset, due to presence of a much higher fraction of overlapping/fuzzy nucleosomes.

Two examples of the disagreement between TF and NORMAL are shown in Figure 2.10. Forward and reverse coverage profiles with extension to 35bp are shown on

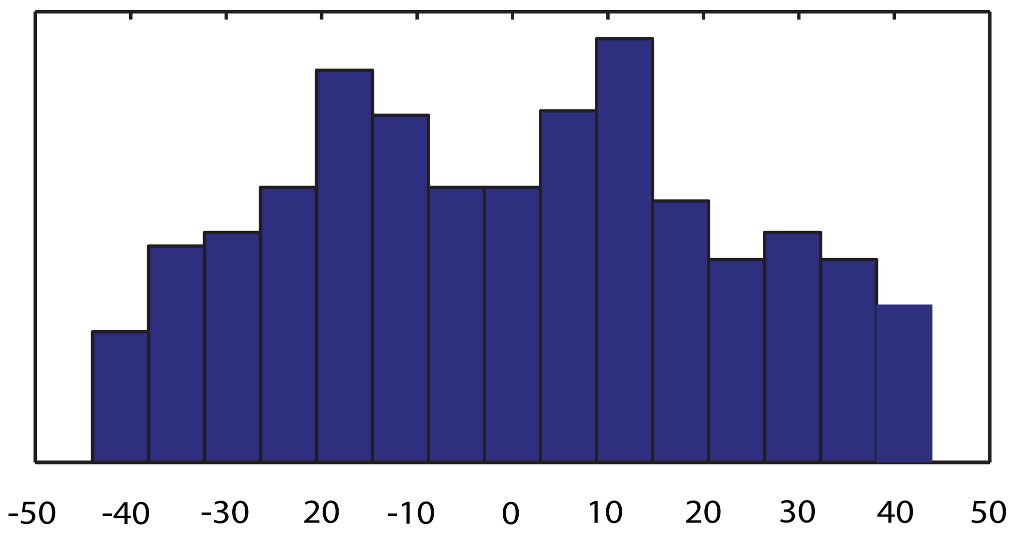

Figure 2.8: Distribution of distances between corresponding nucleosomes reported by TF (range [80,200]) and NORMAL for *P. falciparum* (chromosome 1) across all seven time points

top. Nucleosomes are represented by ovals, where the height of each oval represents the confidence score. NORMAL also reports the variance associated with the left and the right boundary, represented with error-bars.

In Figure 2.10-TOP, we have labeled corresponding nucleosomes **1**–**6**. Some observations are in order. First, nucleosome **3** is an incarnation of the synthetic example in Figure 2.5-DE. The coverage profile around coordinate 800 shows two heavily overlapping nucleosomes that should be reported as one "fuzzy" nucleosome. However, TF reports the position of the nucleosome using the stronger pair of forward/reverse peaks and completely ignores the other pair of peaks. As a consequence, the coordinate of the reported nucleosome is shifted compared to the centroid of the four peaks. NORMAL instead correctly places one nucleosome at the centroid with a relatively high fuzzyness score. Nucleosome **3** is also quite fuzzy, and it is better placed by NORMAL. Some disagreement exists on nucleosome **6** as well. The left boundary of that nucleosome detected by TF correspond to a very weak peak. This is due to the fact that TF's

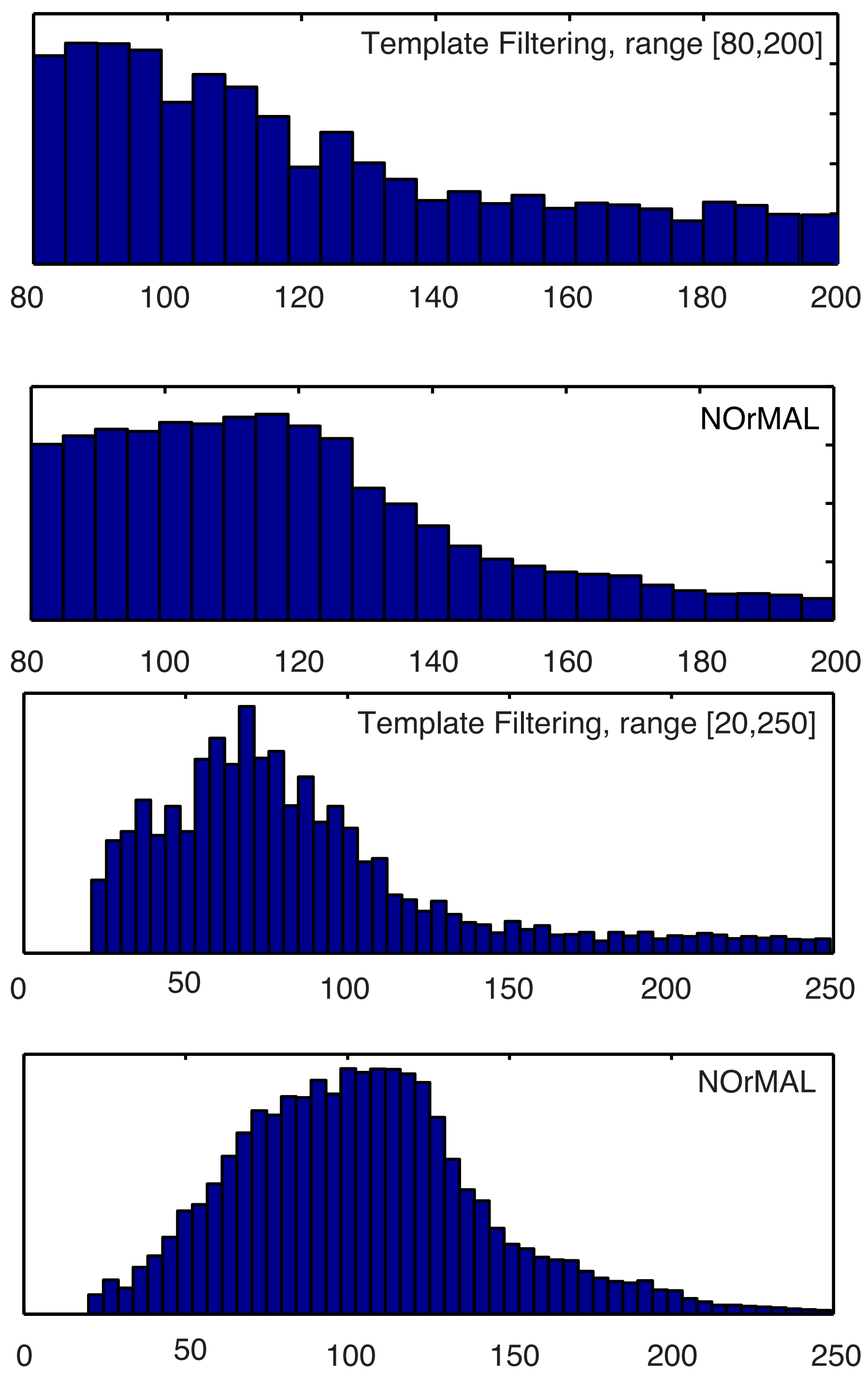

Figure 2.9: Size distribution of reported nucleosomes for *P. falciparum* (chromosome 1) across all seven time points

placement is based on the correlation score rather than the intensity of the peak. In fairness, both methods assign nucleosome **6** a very low confidence score. Finally, TF detects additional nucleosomes **a** and **b**, whereas NORMAL reports additional nucleosome **c**. All these nucleosomes have low confidence scores. Our method did not report **a** and **b** because it explained the data using fuzzy nucleosome **2**. NORMAL should have merged nucleosome **c** to nucleosome **1**, but because the overlap did not exceed the chosen threshold those two nucleosomes were not merged.

Figure 2.10-BOTTOM illustrates a more complex example of coverage profile: even for trained experts placing nucleosomes here would be very challenging. The output of NORMAL and TF are quite consistent for nucleosomes associated to strong peaks. In the regions with high density of peaks, TF tends to place nucleosomes of small sizes (see also Figure 2.9-BOTTOM) and pack them as tight as possible according to allowed overlapping threshold.

In order to increase the agreement between TF and NORMAL we tried to extend the range of nucleosome sizes for TF to [20,250]. Observe that by comparing Figure 2.9-TOP and Figure 2.9-BOTTOM, the size distribution for NORMAL are the same (only truncated in Figure 2.9), while the distribution for TF has changed completely, again pointing out how this range parameter can drastically change the results. Using the extended range, TF was allowed to place smaller nucleosomes so the mode and the mean of the size distribution shifted to smaller values. According to the authors of TF such small nucleosomes can be due to problems in the experimental procedure, namely overexposing the sample to the MNase digestion. However, we have hard evidence from the gel electrophoresis analysis that the expected size of sequenced fragments in our samples after digestion was about 130bp (without adapters). We speculate that TF's approach might have a problem when the range of admissible nucleosome sizes is too wide, and the algorithm confuses boundaries of neighboring nucleosomes.

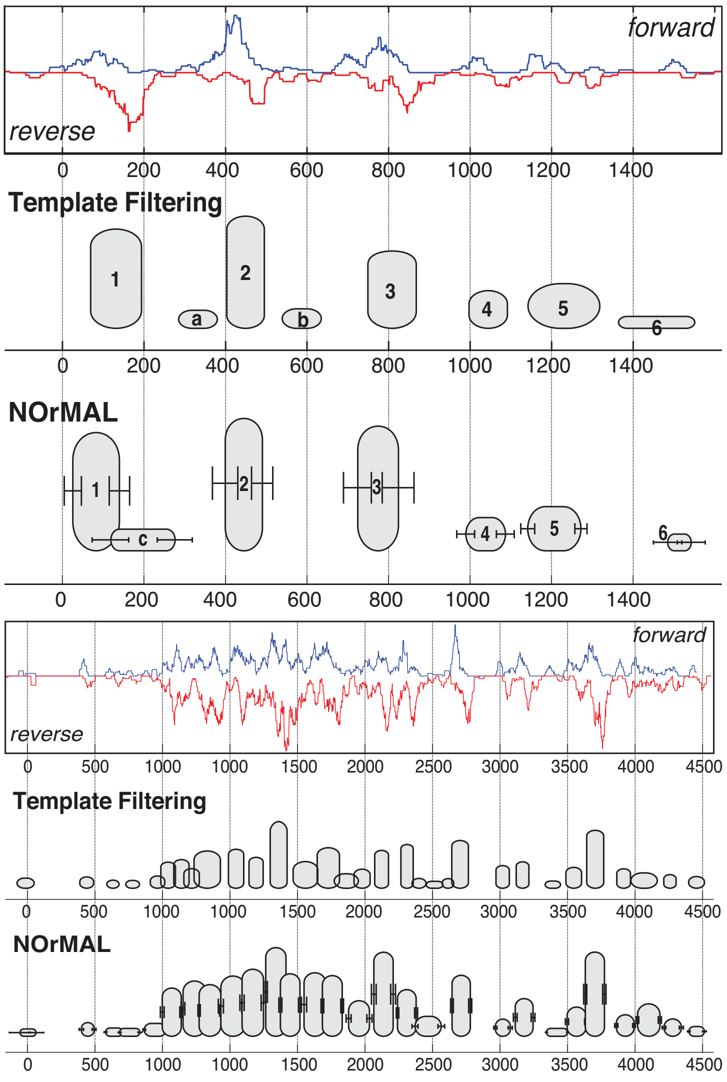

Figure 2.10: Two examples of nucleosome maps for chromosome 1 of *P. falciparum* (TOP: “0” is location 148,500bp, BOTTOM: “0” is location 111,500bp): forward and reverse coverage profiles are shown on top; nucleosomes are represented by ovals where the height of each nucleosome represents the confidence score

The number of reported nucleosomes for all time points is shown in Table 2.3. Observe that for time points 0-24 hours, the number of nucleosomes reported by NOrMAL is between TF with range [80,200] and [20,250]. However, recall that the extended range [20,250] is likely to be unreliable. For time point 30h and 36h, NOrMAL identifies a higher number of nucleosomes. The 30h and 36h marks correspond to the schizont stage of the *P. falciparum* life cycle [61]. During this stage the parasites divide and a

Time	TF [80-200]	TF [20-250]	NOrMAL
0h	1,720	**2,031**	1,934
6h	1,491	**1,826**	1,720
12h	1,461	**2,158**	2,043
18h	1,185	**1,665**	1,537
24h	1,440	**1,910**	1,766
30h	1,723	2,229	**2,443**
36h	1,701	2,514	**2,788**

Table 2.3: Number of nucleosomes reported for *P. falciparum* (chromosome 1) for different time-points (TF: Template Filtering, parameter in square brackets is the range of admissible nucleosome sizes)

large number of nucleosome are added.

2.4 Conclusion

In this Chapter we presented a parametric probabilistic model for nucleosome positioning framed in the context on a modified Gaussian mixture model. Our method directly addresses the challenges imposed by overlapping and fuzzy nucleosomes, their detection and the inference of their characteristics. We demonstrated with a synthetic example that the current state-of-the-art method does not properly handle complex overlapping configurations. On real data, our method detects a higher number of nucleosomes with higher quality detection of nucleosome sizes.

Chapter 3

PuFFIN - A Parameter-free Method to Build Nucleosome Maps from Paired-end Reads

In this Chapter, we introduce a novel method, called PuFFIN, that takes advantage of paired-end short reads to build genome-wide nucleosome maps with larger numbers of detected nucleosomes and higher accuracy than existing tools. In contrast to other approaches that require users to optimize several parameters according to their data (e.g., the maximum allowed nucleosome overlap or legal ranges for the fragment sizes) our algorithm can accurately determine a genome-wide set of non-overlapping nucleosomes without any user-defined parameter. This feature makes PuFFIN significantly easier to use and prevents users from choosing the "wrong" parameters and obtain sub-optimal nucleosome maps.

3.1 Previous work

An analysis of the literature reveals that the majority of nucleosome maps have so far been produced from single-end reads (which are less expensive to obtain than paired-end reads). As a consequence, nearly all computational methods available assume that the input data are single-end reads. Nucleosome positioning from single-end reads is, however, more computationally challenging and much less precise than if paired-end data was available. Paired-end reads allow one to determine both ends of nucleosome-enriched DNA fragments, whereas with single-end reads one either obtains one "boundary" or the other. In the latter case, the problem of associating a peak in the forward strand with the correct peak in the negative strand can be difficult, in particular for complex nucleosome configurations.

Existing methods for single-end reads either rely on the assumption that nucleosome-enriched DNA fragments are expected to be of a size compatible with the nucleosome ($\approx$ 146 bp), or use probabilistic models to estimate these sizes from the data. From our experience, the first approach can lead to poor results because there is no fragment size that will work equally well for all nucleosomes in the genome. While one would expect nucleosome-enriched DNA fragments to be about 146 bp, in MNase-Seq the digestion process can either leave nucleosome-free DNA in the sample, or "over-digest" the ends of nucleosome-bound DNA. Furthermore, the rate of digestion is sequence-dependent [2, 77], so nucleosomes in different genomic locations can end up with different DNA fragment sizes.

Despite these challenges, the majority of so-called "peak-calling" approaches usually rely on the assumption that the data is derived from nucleosome-sized DNA fragments and consist of following steps: (1) a nucleosome occupancy score function is obtained from mapping nucleosome-enriched reads to the reference genome, followed by counting, smoothing and normalization; (2) candidate nucleosomes are placed according to the peaks of the score function; (3) the final set of nucleosomes is selected to satisfy additional constraints (which are tool-dependent). To compute the occupancy score, different techniques have been proposed, ranging from simply computing the number of reads covering each genomic location, to sophisticated statistics to estimate the false discovery rate. For instance, nucleR, [21] uses the raw coverage with extensive "profile cleaning" based on the Fourier transform, whereas NSeq [50] employs a triangle statistic based on read counts within a sliding window. From a probabilistic point of view, the occupancy score represents a non-parametric distribution of nucleosome positions. Despite being defined non-parametric, building such a score function relies heavily on a user-defined parameters (e.g., window sizes, smoothing parameters, etc).

A second group of methods is based on probabilistic models. Our tool NOrMAL [54] uses a modified Gaussian mixture model to infer nucleosome-enriched fragment sizes. The parametric probabilistic model allows to deal with the problem of overlapping and complex configurations of nucleosomes. In previous chapter, we showed that additional information about nucleosome fragment size could significantly improve the accuracy of nucleosome mapping for the organisms, which have complex nucleosome patterns. Developed in parallel with NOrMAL, Ping [85] employs a similar probabilistic model. Both tools provide a clear advantage over algorithms that rely on the user to provide estimated DNA fragment sizes. Even-though these models are parametric, these approaches are much more robust against wrong choices in user-defined parameters, because they can adapt to the data. One should also keep in mind that being based on iterative procedures to infer parameters, the performance of these methods also depends from (several) internal parameters.

Finally, a distinct group of positioning methods depend on the availability of a control track (i.e., "naked" DNA), e.g., NucleoFinder [4], while others have been designed to perform differential nucleosome positioning, e.g., Danpos [9] and DiNuP [22].

In this chapter, we focus on the problem of determining nucleosome positions based on the availability of paired-end reads (without a control track). To the best of our knowledge, NucPosSimulator [66] is the only published tool specifically designed to take advantage of paired-end reads: to place nucleosomes it solves the optimization problem of selecting the subset of peaks which maximizes the total score, under the constraint that these peaks are located at the expected nucleosome distance from each other. Our tool PuFFIN (Positioning for Fuzzy and FIxed Nucleosomes) instead uses a novel multi-resolution approach: while its algorithm is relatively simple, our approach introduces some novel ideas that have the potential to be useful in other domains of genome analysis.

3.2 Methods

Our method consists of three steps: (A) we build a set of nucleosome profiles and nucleosome "landscapes"; (B) we detect candidate nucleosome locations on each profile; (C) we select a "consensus" set of nucleosomes that satisfies non-overlapping constraints. We discuss these steps in detail in the next subsections.

3.2.1 Computing Nucleosome Profiles and Nucleosome Landscapes

We first map sequenced reads to the reference genome and then compute a *nucleosome profile* that represents the likelihood that a genomic location is occupied by a nucleosome. Candidate nucleosomes are detected at the peaks of the nucleosome profile. In order to reduce false positives, profiles have to be cleaned from their high frequency component. Choosing the best smoothing method (and its parameters) is, however, not easy. For instance, in [21] the authors show that the kernel density estimation method [53] works significantly better than moving average-based smoothing. The choice of kernel parameters is also important: too much smoothing can merge adjacent peaks, too little can leave noisy artifacts that can be interpreted as peaks and thus introduce spurious nucleosomes. To address the challenges of choosing the "right" kernel and smoothing parameters, we follow an alternative (novel) procedure to construct nucleosome profiles.

First, we replace each mapped paired-end read i with a function $f_i^{\alpha w_i}$ distributed as a Gaussian with mean μ_i and standard deviation αw_i, i.e.,

$$f_i^{\alpha w_i}(x) = \frac{1}{\alpha w_i \sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{(\alpha w_i)^2}}$$

where μ_i is the genomic center location of read i, w_i is the length of read i (i.e., the distance between the leftmost nucleotide in the left mate and rightmost nucleotide of the right mate), and α is a smoothing parameter. Replacing each mapped read with a gaussian distribution allows us to model probabilistically the uncertainty in the paired-end mapping. For instance, when the left and right mate are mapped far from each other, the mass of the gaussian will be distributed on a longer interval because of its large variance. If instead the left and right mate are close to each other, the gaussian will have its mass concentrated at the center of the read, indicating a higher confidence in the nucleosome position.

Then, we compute the *nucleosome profile* S_α as the weighted sum of functions

f_i^{α} for all the mapped reads in the input

$$S_{\alpha}(x) = \sum_{i=1}^{n} \beta_i f_i^{\alpha w_i}(x)$$

where n is the number of mapped reads in input, and β_i is the weight of the read i. If we had employed a uniform weighting scheme ($\beta_i = \frac{1}{n}$), paired-end reads with very short insert would dominate the profiles. To reduce the effects of short DNA fragments, we use a non-uniform weighting scheme. For paired-end reads that are shorter than 146bp, we assign a penalty factor $\gamma(w) < 1$, such that the shorter the read is, the less the weight is (i.e., $\beta_i = \frac{1}{n}\gamma(w_i)$). Additionally, one could use the weights β_i to account for sequence quality of individual reads, mappability biases, etc.

As said, parameter α controls the smoothness of function S_{α}. The bigger is α, the smoother is S_{α} (peaks will be wider), and vice versa. When α is large, we capture nucleosome binding preferences at a lower resolution scale; when α is small we can detect nucleosomes at a high resolution scale (but noisier). In the limit $\alpha \to 0$, function $S_{\alpha}(x) \to \sum_{i=1}^{n} \chi(x - \mu_i)$, where $\chi(x) = \begin{cases} 1, x = 0 \\ 0, x \neq 0 \end{cases}$ is the indicator function. In this case, $S_0(x)$ represents how many read centers cover location x in the genome.

One might think that one could obtain the same profiles by computing the read coverage function smoothed by a Gaussian kernel. There is, however, a significant difference: the size of each mapped read *independently* influences the shape of S_{α} (no matter what smoothing parameter is chosen), while in the case of kernel smoothing the impact of read sizes becomes less and less important as the smoothing strength increases.

Since we do not know the appropriate value for α for the data, in this step we generate a family of functions for several choices of α. Formally, we create a set of m functions $\{S_{\alpha_k}\}_{k=1,2,\ldots,m} = \{S_{\alpha_1}, S_{\alpha_2}, \ldots, S_{\alpha_m}\}$, where $\alpha_1 < \alpha_2 < \cdots < \alpha_m$ are m distinct choices for α. The value m is hard-coded in our implementation (we used $m = 40$ for all the experiments).

The set of functions $\{S_{\alpha_k}\}_{k=1,2,\ldots,m}$ enables our algorithm to detect candidate locations for nucleosomes at different resolution scales, thus eliminating the need to specify in advance the parameters for the range of nucleosome-enriched fragments. In other words, our algorithm can "adapt" to the local properties of the input data by processing the same location at different resolutions (corresponding to the choices of α).

Finally, we compute a set of *nucleosome landscapes* $\{N_{\alpha_k}\}_{k=1,2,\ldots,m}$ by normalizing each function S_{α_k} by the lowest resolution function S_A, as follows

$$N_{\alpha_k}(x) = \log\left(\frac{S_{\alpha_k}(x) + \epsilon}{S_A(x) + \epsilon}\right)$$

where $\epsilon > 0$ is a small constant to avoid a division by zero, and $A > \max_{k=1,2,\ldots,m} \alpha_i$. In our implementation we pick α to range from 0.05 to 0.63 and $A = 1.5$. Since mappability biases affect each function $\{S_{\alpha_k}\}$, we can effectively reduce these biases by taking the log ratio of high-resolution and low-resolution function. Another reason to carry out this normalization step is to reduce the differences in the peak heights.

To illustrate the multi-resolution approach in our algorithm, we created a small synthetic dataset with four nucleosomes shown in Figure 3.2. Panel A shows the raw coverage obtained by mapping synthetic paired-end reads to the reference genome. Observe that nucleosomes I,III and IV are strongly positioned, while nucleosome II is "fuzzy". Fuzzy nucleosomes are quite common and occur when a subset of the cells in the sample has a nucleosome at one location, while in the other subset the same nucleosome is slightly shifted. Nucleosome I is isolated, while nucleosomes III and IV are located very close to each other. Panel B shows the family of functions $\{S_{\alpha_k}\}$ for three choices of α; panel C illustrates the set of nucleosome landscape functions $\{N_{\alpha_k}\}$. Observe in Figure 3.2C that the transformation amplifies candidate peaks in areas with low coverage and reduces the amplitude of peaks in regions with high coverage.

3.2.2 Detecting Candidate Nucleosomes

By construction, a nucleosome landscape N_{α_k} represents a non-parametric distribution of nucleosomes at resolution α_k. The presence of a peak in any nucleosome landscape indicates a candidate nucleosome. The reads that form corresponding peak belong to that candidate.

A peak is defined by a pair (q, s) where q is the center of the peak and s is the width of the peak. We say that (q, s) is a *peak* for function N when $N(q)$ is local maximum for N and $s = \min_z(|q - z|)$ where z is any local minimum for function N.

Detecting peaks on each function N_{α_k} can be easily computed in linear time along the length of the genome. As a result, for every choice of α_k, $k = 1, 2, \ldots, m$ we have a set of peaks $P_{\alpha_k} = \{p_{k_1}, p_{k_2}, \ldots, p_{k_l}\}$, where p_{k_j} is a pair (center, width) representing the peak, and l is the number of peaks.

Peaks are however not guaranteed to have a symmetric shape. We therefore recompute the location of every nucleosome candidate as the centroid location of its read midpoints. This additional step ensures that candidate nucleosome locations properly represent the corresponding input reads.

3.2.3 Building the Final Solution

We now explain how to build the final set of non-overlapping nucleosomes from the family of peak sets $\{P_{\alpha_k}\}_{k=1,\ldots,m}$. We say that two peaks (q_1, s_1) and (q_2, s_2) *overlap* if $|q_1 - q_2| < 146$ (the size of a nucleosome). Observe that by construction, the number of peaks detected at lower resolution (i.e., for large α) will be smaller than or equal to the number of peaks detected at higher resolution, i.e., $|P_\alpha| \leq |P_\beta|$ when $\alpha > \beta$. As we increase the smoothing parameter α, the total number of peaks decreases: while some peaks are preserved, others are merged. In other words, for every peak in P_α we can find at least one corresponding peak in P_β if $\alpha > \beta$.

Based on this observation, we build the final set of non-overlapping nucleosomes C as follows. Given a family of peak sets $\{P_{\alpha_k}\}_{k=1,\ldots,m}$ where $\alpha_1 < \alpha_2 < \cdots < \alpha_m$, we process each peak set P_{α_k} in increasing order for α. We add a peak p from the current set P_{α_k} to the final solution C if p does not overlap with any other peak in the set P_{α_k} and if p does not overlap with any other peak already in C. A sketch of the algorithm can be found in Figure 3.1.

Let us consider again our example in Figure 3.2. Detected peaks are marked with circles in panel C. The algorithm first processes the set of peaks on the blue function ($\alpha = 0.07$). Since there are no peaks on that curve that are located at a distance greater than 146bp from each other, the final set C remains empty. Next, the algorithm processes the green curve ($\alpha = 0.21$): here there are three peaks that satisfy the non-overlapping constraint. Thus, the algorithm adds those peaks (marked with solid circles) to C. Then, the algorithm considers the red curve ($\alpha = 0.62$): all four peaks are non-overlapping with each other, however only one peak (marked with the solid circle) can be added to C. As a result, the final solution C consists a set of four peaks that match the original nucleosomes. Observe that strongly positioned nucleosomes I, III and IV are detected earlier in the algorithm ($\alpha = 0.21$) than fuzzy nucleosome II ($\alpha = 0.62$).

```
Data: $\{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$ - a set of $n$ paired-end input reads,
        $x_i$ is leftmost and $y_i$ - rightmost coordinate
$(\alpha_1,\alpha_2,...,\alpha_m)$ - a set of choices for smoothing parameter $\alpha$ and a
predefined constant $A > \alpha_i, i = 1..m$
Result: $C$ - resulting set of nucleosomes
for $\forall i = 1..n$ do
    $\mu_i \leftarrow \frac{x_i+y_i}{2}$ - location of the midpoint for read $i$
    $w_i \leftarrow (y_i - x_i)$ - size of read $i$
end
for $\forall \alpha \in \{\alpha_1,...,\alpha_m\}$ do
    $S_\alpha(x) = \sum_{i=1}^{n} \beta_i f_i^{\alpha w_i}(x)$ where $f_i^{\alpha w_i}(x) = \frac{1}{\alpha w_i \sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{(\alpha w_i)^2}}$
    $N_\alpha(x) = \log\left(\frac{S_\alpha(x)+\epsilon}{S_A(x)+\epsilon}\right)$, where $\epsilon > 0$
end
for $\forall N_\alpha \in N = \{N_1,...,N_{\alpha_m}\}$ do
    $P_\alpha \leftarrow$ a set of peaks for $N_\alpha$, each peak corresponds to some candidate
    nucleosome
end
$C \leftarrow \emptyset$ - a consensus set of nucleosomes to report
for $k \in 1,\dots,m$ do
    for $\forall$ peak $(q,s) \in P_{\alpha_k}$ do
        if $|q-\phi| > 146, \forall \phi \neq q$ such that $(\phi,\psi) \in P_{\alpha_k} \cup C$ then
            $C \leftarrow C \cup (q,s)$
        end
    end
    $P_{\alpha_k} \leftarrow P_{\alpha_k} \setminus C$
end
return $C$
```

Figure 3.1: Sketch of the proposed PuFFIN algorithm

3.2.4 Running Time

To compute a set of profile functions S_α we use a precomputed set of curves $f_i^{\alpha w}$ for every choice α and w in a predefined range. As a result, it takes $\Theta(nm)$ operations, where n is the number of reads and m is the number of curves. In our implementation we used $m = 40$ choices of equally distributed values for $\alpha \in [0.05, 0.63]$.

Finding peaks on each curve S_α takes $\Theta(l)$ time, where l is the length of the processed region. Thus, the total time to find candidate nucleosomes (Figure 3.1, lines 1-3) is $\Theta(m(n + l))$. Building the resulting set of non-overlapping nucleosomes is determined by the number of candidates that is at most $\Theta(ml)$. Given that m is predefined, it follows that the total running time is linear in the region size and number of input reads.

3.3 Experimental Results

To evaluate the performance of PuFFIN, we performed extensive benchmarking against NucPosSimulator, Template Filtering and NOrMAL. NucPosSimulator is the only published tool designed to deal with paired-end reads [66]. As said, it solves the optimization problem of selecting the subset of peaks which maximizes the total score, under the constraint that these peaks are located at the expected nucleosome distance from each other. Template Filtering is one of the first algorithms developed to infer the size of the fragments from single-end reads [77]. NOrMAL uses a modified Gaussian model to cluster input single-reads such that every cluster represents a nucleosome [54]. Some of the recently published tools that use a control sample to solve the nucleosome positioning problem, e.g., Danpos and NucleoFinder, are not included in this comparison.

We used default parameters for each tool except for the following provisions. For Template Filtering and NOrMAL we set to zero the allowed overlap between adjacent nucleosomes to allow for a fair comparison with PuFFIN and NucPosSimulator.

Arguably the major challenge for nucleosome position inference is that the true positions of nucleosomes are unknown. The lack of a "ground-truth" makes it very hard to benchmark existing computational methods. For this reason we made extensive use of synthetic data, as explained next.

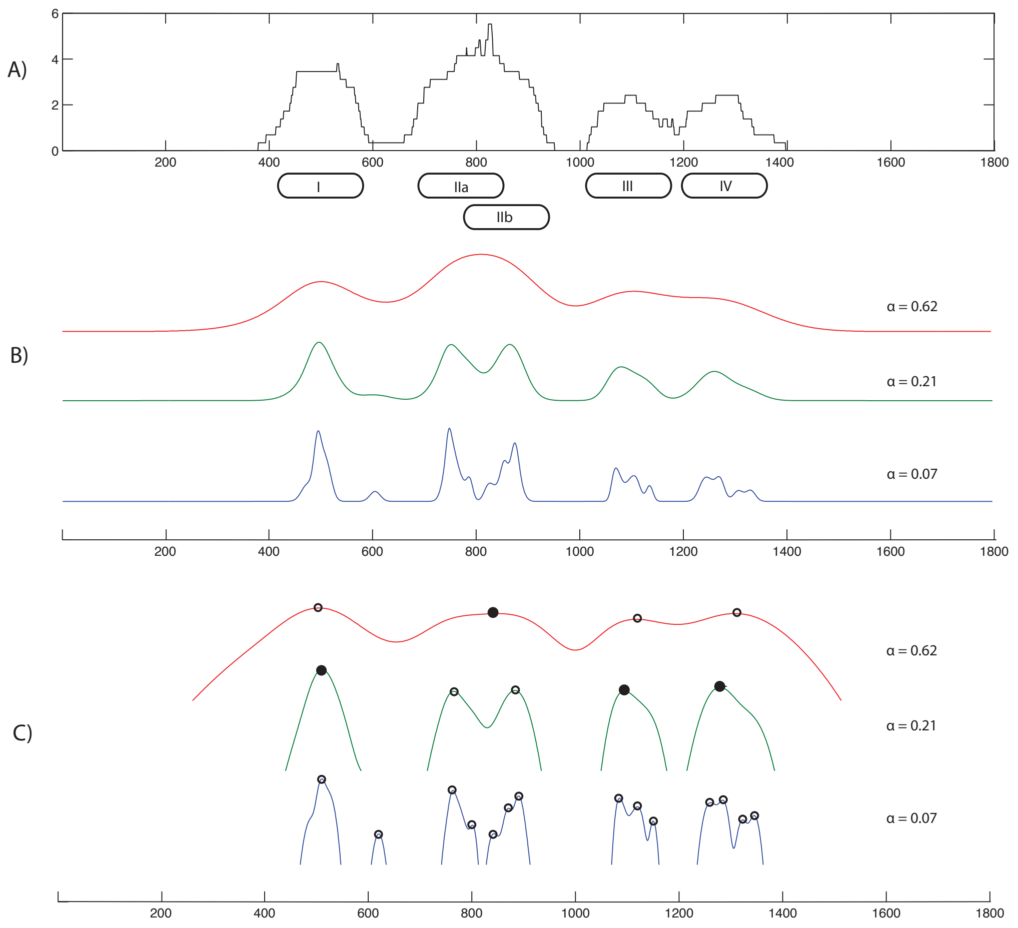

Figure 3.2: Synthetic illustrative example. A) read coverage; B) Nucleosome profiles $\{S_{\alpha_k}\}$ for $\alpha_1 = 0.07$ (blue), $\alpha_2 = 0.21$ (green), $\alpha_3 = 0.62$ (red) C) Corresponding nucleosome landscapes $\{N_{\alpha_k}\}$ (see text for detailed explanation)

3.3.1 Results on Synthetic Data

We started by producing a small dataset of reads corresponding to DNA-enriched fragments for only one nucleosome (Figure 3.3). This allowed us to investigate the behavior of these various tools in the scenario of low sequence coverage in a region containing a fuzzy nucleosome. Nucleosome I is centered at 300bp and the paired-end reads of size 146bp were generated with midpoints distributed according to Gaussian with mean 300, and standard deviation 40. To simulate a low coverage scenario, we generated only twenty sequence reads (20-fold coverage). PuFFIN, Template Filtering

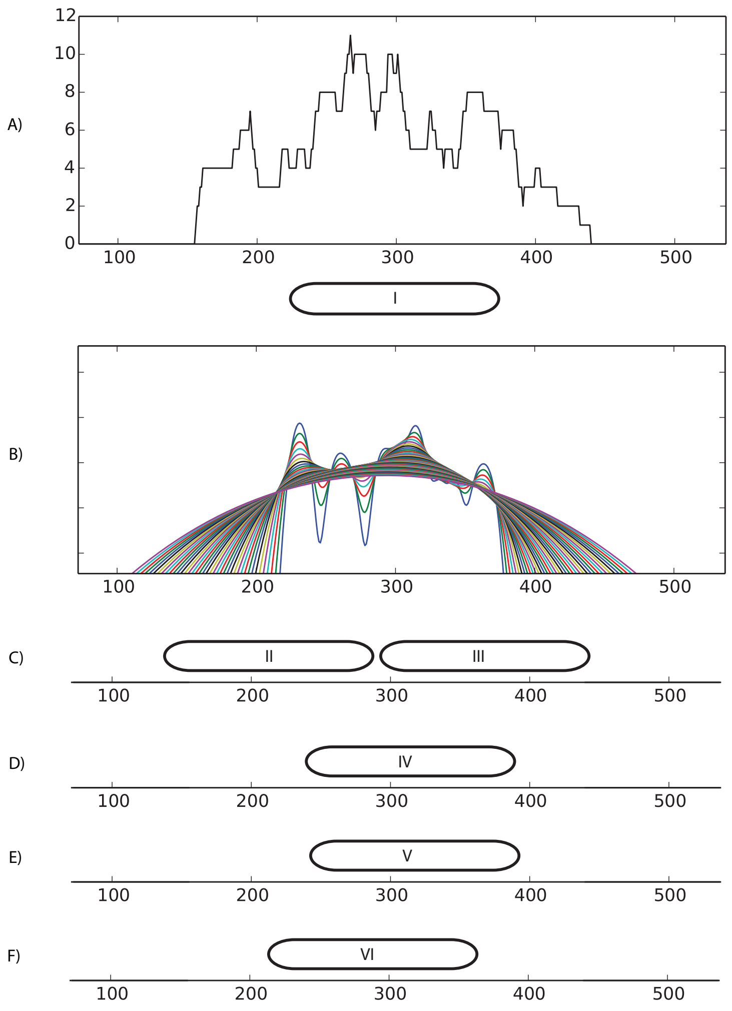

Figure 3.3: A "toy" example. A) Raw coverage; B) Nucleosome landscapes for different choices of $\alpha \in [0.05, 0.63)$; C) NucPosSimulator result; D) PuFFIN result; E) Template Filtering result; F) NOrMAL result;

and NOrMAL report one nucleosome located at 308bp, 311bp and 292bp, respectively, while NucPosSimulator reports two nucleosomes positioned at 221bp and 369bp. The slight difference of the reported locations for the first three tools could be explained by the small sample size that is insufficient to recover the true location. Interestingly, the first two methods, which are based on peak-detection, produced a similar close right shift, while the nucleosome detected by NOrMAL showed a small left shift. NucPosSimulator detected two distinct nucleosomes, probably because the objective of this tool is to maximize the total score of reported nucleosomes. We believe that maximizing this quantity has the undesirable effect to over-report nucleosomes (i.e., increase false positives). Decreasing the smoothing parameter in NucPosSimulator from 20.0 (default) to 2.0 reduces the output to a single nucleosome, again demonstrating how the choice of smoothing parameters can have significant effects on the results.

Next, we performed a more realistic comparison on *in silico* reads for larger synthetic nucleosome maps. We used the nucleosome map generator `syntheticNucMap` from nucleR [21]. This tool allows users to specify the number of well-positioned and fuzzy nucleosomes, as well as the variance for the location of synthetic reads and the coverage level. Well-positioned nucleosomes are placed along the chromosome regularly spaced with a fixed linker size (we used linkers of 20bp, which introduces a periodicity of $\approx$ 167bp). For fuzzy nucleosomes, locations are picked at random and independently from other nucleosomes already on the chromosome. As a consequence, fuzzy nucleosomes can overlap with other nucleosomes. For the variance parameter we choose 30 bases for well-positioned and 50 bases for fuzzy nucleosomes.

Our objective was to investigate the accuracy of nucleosome detection as a function of the fraction of fuzzy nucleosomes: we expected the detection problem to become increasingly harder as the number of fuzzy/overlapping nucleosomes increases. For each percentage level of fuzzy nucleosomes (0%, 10% ..., 100%) we generated ten datasets of synthetic reads for a map containing 1,000 synthetic nucleosomes. To build these datasets, we used the following command: `syntheticNucMap(wp.num=1100, wp.del=(100+r*100), wp.var=30, fuz.num=(r*100), fuz.var=50, max.cover=70, nuc.len=147, lin.len=20)`, where r controls the fraction of fuzzy nucleosomes ($r = 0$ is 0%, $r = 1$ is 10%, ..., $r = 10$ is 100%).

For each group of ten datasets we measured the number of reported nucleo-

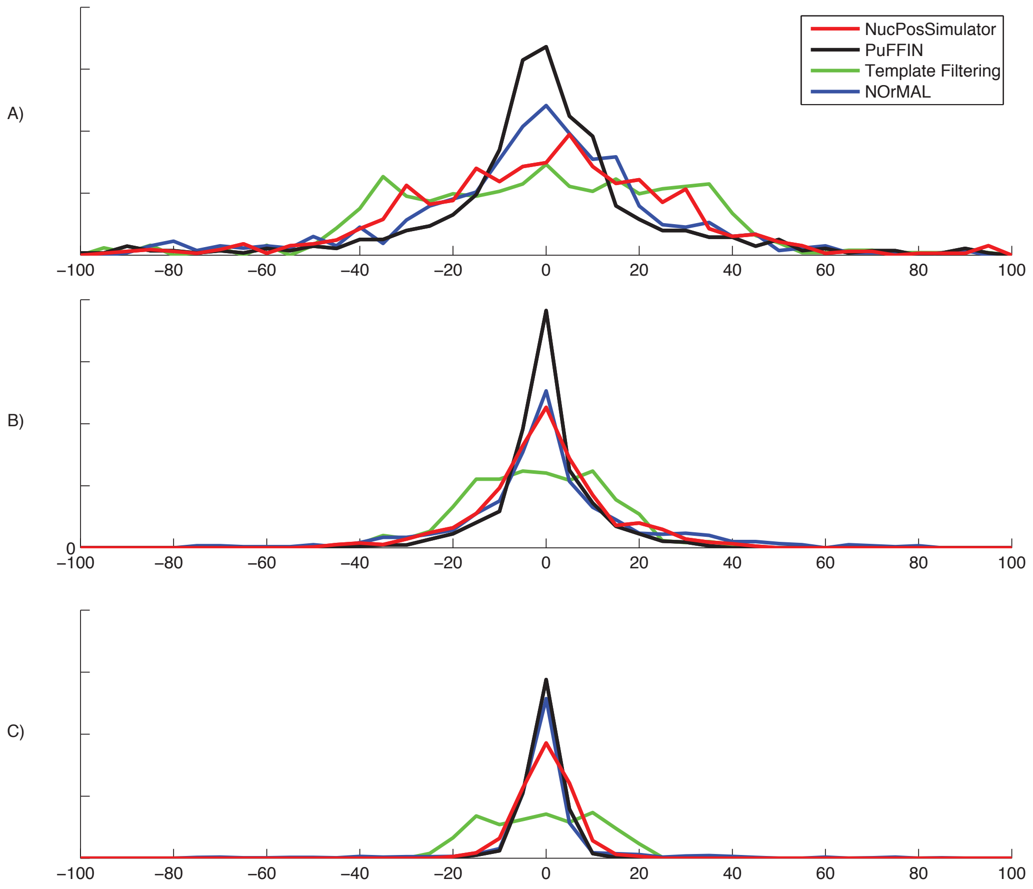

Figure 3.4: Distribution of the distances between detected and true locations for A) 100%, B) 60% and C) 0% fuzzy dataset

somes and the accuracy of each tool, and reported the average and standard deviation over the ten sets. To measure the accuracy, we calculated the distances between the true nucleosome location and the center of the corresponding detected nucleosome. Results in Figure 3.4 show that PuFFIN reports nucleosome positions more accurately in datasets with larger proportions of fuzzy nucleosomes. In addition, Figure 3.5 shows the average number of nucleosomes detected by the various tools for increasing percentages of fuzzy nucleosome (the error bar represents the standard deviation over the ten datasets). First observe that although each dataset is expected to have synthetic reads for exactly 1,000 nucleosomes, this is only true for datasets with no fuzzy nucleosomes. Since fuzzy nucleosomes may overlap other nucleosomes, we expect to detect a decreasing numbers of nucleosomes as the percentage of fuzzy nucleosomes increases (which is reflected in Figure 3.5). Also observe in Figure 3.5 that in datasets with more than 20% of fuzzy nucleosomes, NucPosSimulator detects the highest number of nucleosomes compared to other tools. However, as we demonstrated earlier in Figure 3.3, NucPosSimulator can over-report nucleosomes. To explore whether this was true on these larger datasets, we computed the distribution of distances between adjacent nucleosomes (Figure 3.6). In the group of datasets with no fuzzy nucleosomes, both NucPosSimulator and PuFFIN have strong peak at around 167bp location and 334bp. This is expected, because all nucleosomes are well-positioned and are located at multiples of 167bp. However, as we increase the percentage of fuzzy nucleosomes in the datasets, NucPosSimulator reports more and more nucleosomes exactly 148 bp apart from each other, which suggests that its strategy to maximize the total score for reported nucleosomes has the effect of reporting too many nucleosomes.

To eliminate the effects of over-reporting in NucPosSimulator, we discarded from the counts nucleosomes that are located 148 bases or less from each other, such that every pair of tightly placed nucleosomes is count as one nucleosome. In Figure 3.5, curves marked "filtered" shows the results of this cleaning step. Observe that the number of nucleosomes reported by NucPosSimulator drops significantly, while only a small number of PuFFIN nucleosomes are affected. In fact, using this cleaning step, PuFFIN reports a larger numbers of nucleosomes than NucPosSimulator. All together, these experimental results on synthetic data show that PuFFIN generates more accurate nucleosome maps, without over-reporting nucleosomes.

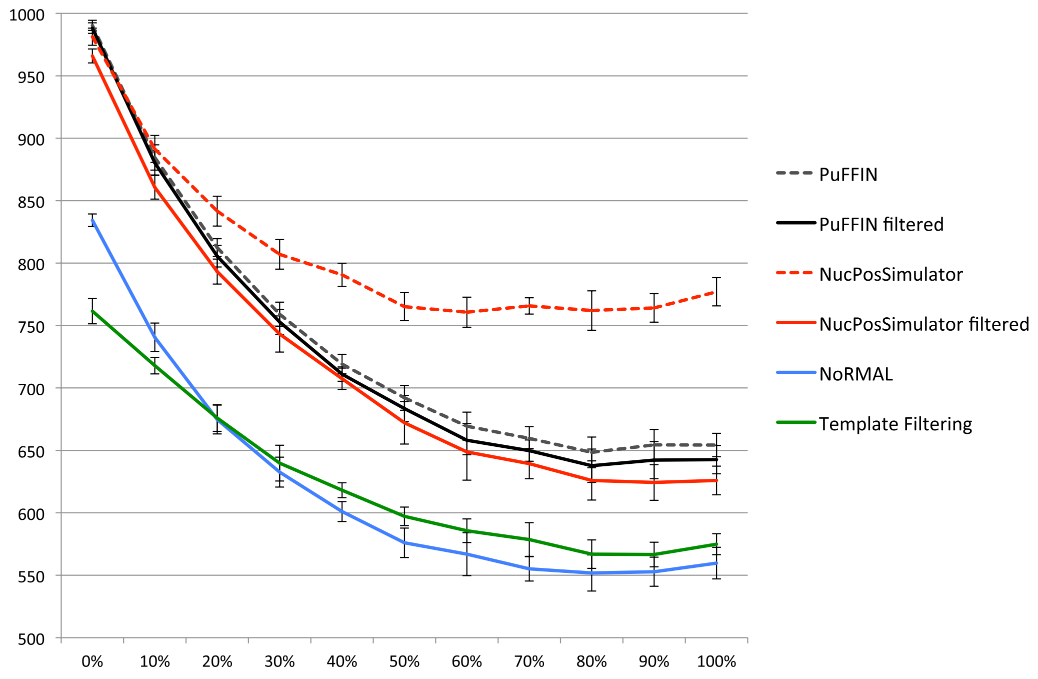

Figure 3.5: Dependency between the percentage of fuzzy reads in the sample (X axis) and the number of detected nucleosomes (Y axis) for synthetic dataset

3.3.2 Results on Real Data

For the comparison of nucleosome positioning tools, we used a publicly available dataset for *S. cerevisiae* (NCBI SRA SRR094649, [13]) and our dataset for *P. falciparum* (NCBI SRA SRS453761). All datasets contain paired-end reads produced by an Illumina sequencing instrument. Reads were mapped to their corresponding reference genomes using Bowtie2 [40] with `--very-fast-local --no-discordant` flags. We removed reads that were not mapped uniquely or had a distance between the left and right mates smaller than 40bp or bigger than 1,000bp.

Experimental results are summarized in Table 3.1, which include the number of reported nucleosomes and the execution time. Nucleosome positioning in *S. cerevisiae* is extensively studied and the majority of the tools perform well on this organism. Also, nucleosomes in yeast are well-positioned and not many overlaps are present. The results in Table 3.1 show that the number of nucleosome reported in yeast by these tools are

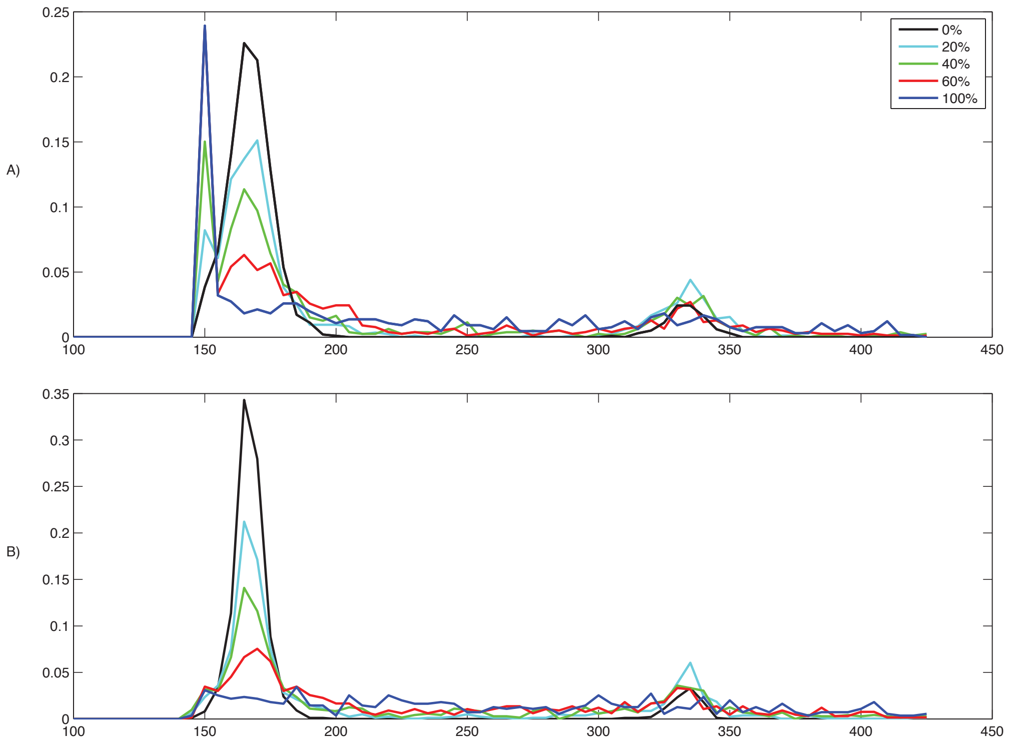

Figure 3.6: Distribution of the distances between adjacent nucleosomes for A) NucPosSimulator B) PuFFIN

quite similar, except for NucPosSimulator that reports a significantly larger number. These results possibly again suggest the over-reporting behavior of this tool.

Our previous work [54] has demonstrated that the *P. falciparum* genome has a greater complexity of nucleosomes configurations. As expected, experimental results show much greater variance in the number of nucleosomes in the malaria dataset reported by the various tools. PuFFIN reports a similar number of nucleosomes compared to NucPosSimulator, but significantly higher numbers than NOrMAL and Template Filtering, indicating that our method is capable to resolve complex configurations of nucleosomes.

The execution time of PuFFIN is higher than NOrMAL and Template Filtering on both datasets, but shorter than NucPosSimulator on *P. falciparum*

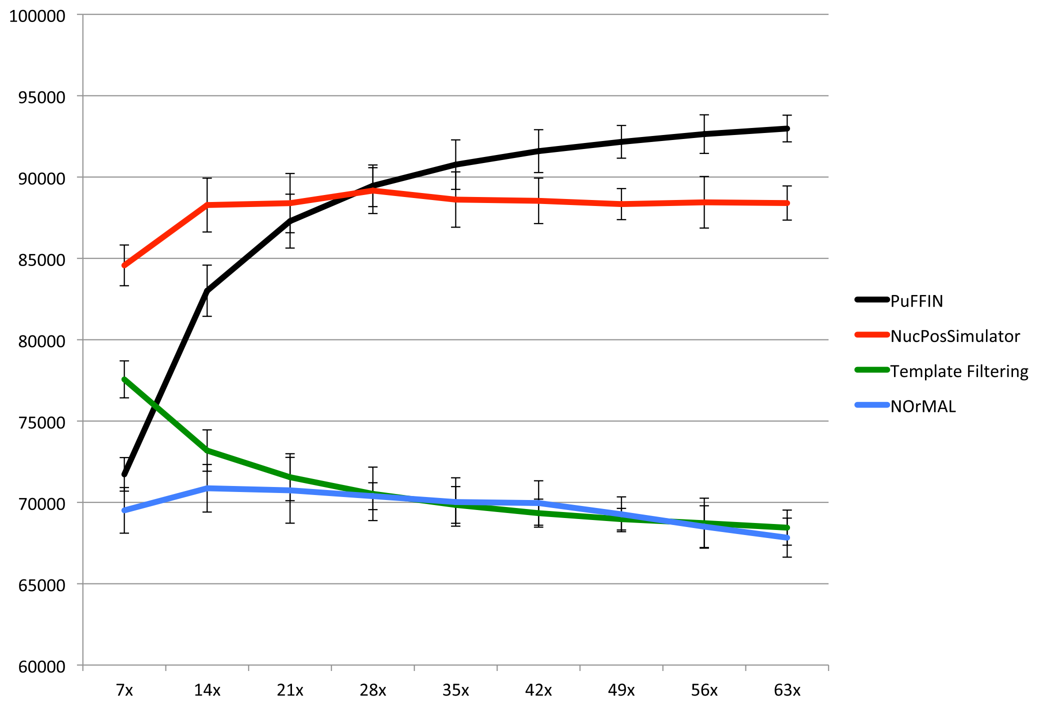

Figure 3.7: Dependency between the fold coverage (X axis) and number of detected nucleosomes (Y axis) for *P. falciparum*

	S. cerevisiae (W303 contig 7)		*P. falciparum* (3D7 chr. 2)	
	#nucleosomes	time (sec)	#nucleosomes	time (sec)
TEMPLATE FILTERING	630	**1**	2,725	**13**
NORMAL	592	4	3,247	40
NUCPOSSIMULATOR	**802**	75	3,722	920
PUFFIN	709	165	**3,760**	350

Table 3.1: Number of reported nucleosomes and execution times on yeast and the human malaria parasite.

and higher on *S. cerevisiae* datasets. Our implementation of PUFFIN is currently written in Python, while the other tools use either Java or C/C++. We believe that speed of our tool could be easily improved by one order of magnitude by implementing it in C/C++.

To investigate the sensitivity of the tools on the quantity of the input data (coverage), we performed an experiment in which an increasing fractions of the input reads were discarded. Specifically, we sampled the *P. falciparum* dataset by randomly selecting a given fraction of the input reads (20%, 30% . . . , 100%) and ran the four tools on the resulting datasets. Subsamples have 7x, 14x,. . . , 63x fold coverage. Figure 3.7 shows that the performance of PuFFIN degrades monotonically as the quantity of the data decreases, while NucPosSimulator remains more stable over a larger range of input data. We therefore recommend to use sequence data with a minimum of 30-fold for the analysis of nucleosome positions if PuFFIN is used.

3.3.2.1 Association between nucleosome occupancy and gene expression

PuFFIN was used to study the correlation between nucleososome positioning and gene expression in the human malaria parasite [6]. As a part of that study we carried out nucleosome positioning on two independent datasets for *P. falciparum* from [5, 3]. The former dataset was composed by single-end reads, so we extended each reads to 146 bp in order to use PuFFIN.

We performed an association analysis between gene expression levels (from RNA-Seq data) and nucleosome occupancy (produced by PuFFIN) to test the hypothesis that decreased nucleosome occupancy in promoter regions is associated with higher transcriptional activity [42].

First, we grouped genes into ten transcription clusters based on steady-state mRNA levels. Then, we computed nucleosome occupancy levels for the 500 bp region directly upstream of the translation start codon and for gene bodies.

The analysis indicated a strong correlation between nucleosome density in the promoter region (both in terms of number of nucleosomes and nucleosome levels) and transcriptional activity. Figure 3.8 A,B,E,F clearly shows that highly expressed gene clusters have a more open chromatin (i.e., less nucleosomes) than clusters of genes with low expression values. We observed the opposite correlation between nucleosome occupancy and transcriptional activity inside coding regions of highly expressed genes, as compared to the promoter regions. Highly transcribed genes were on average bound by more nucleosomes and at higher levels (Figure 3.8 C,D,G,H). We also tried a different number of clusters, obtaining similar results.

We carried out the same analysis on nucleosome maps obtained by NOrMAL. We observed similar, but somewhat weaker, association between nucleosome occupancy and transcript levels in this case, indicating that PuFFIN extract more biologically-meaningful features from the data over our previous tool NOrMAL.

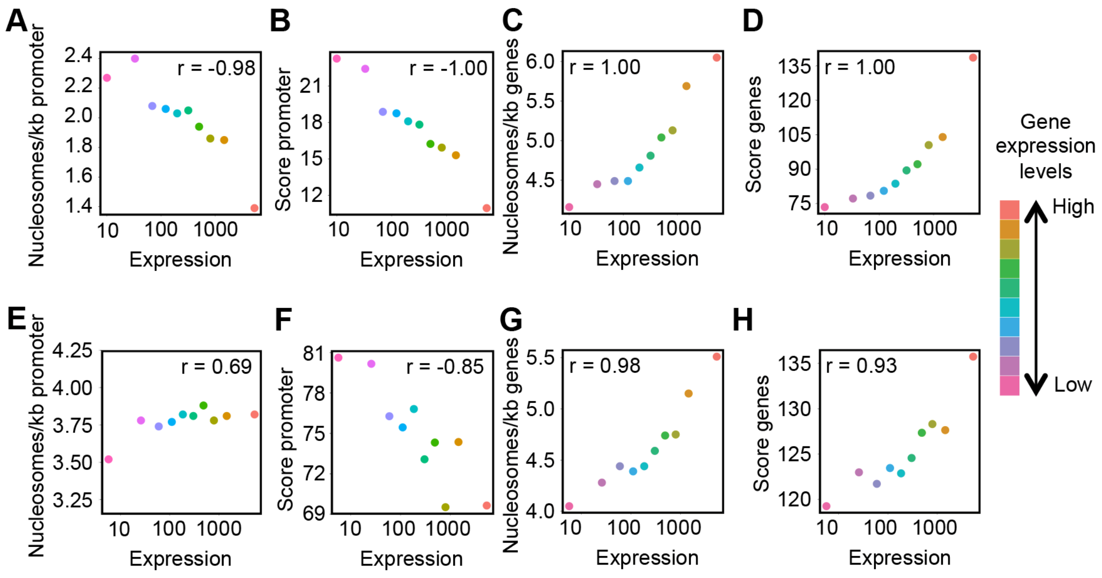

Figure 3.8: Association between nucleosome occupancy and gene transcription levels. TOP: analysis for [5] data, BOTTOM: for [3]. A,E: Average number of nucleosomes per kilobase upstream of the TSS versus average expression level for each transcription cluster; B,F: Average nucleosome score upstream of the TSS versus average expression level; C,G: Average number of nucleosomes per kilobase inside coding regions versus average expression level for each transcription cluster; D,H: Average nucleosome score inside coding regions versus average expression level. Value r represents Spearman's correlation coefficient

3.4 Conclusion

In this Chapter, we described a novel method to solve the nucleosome positioning problem when paired-end data is available. Our method employs a multi-resolution strategy that circumvents a smoothing step that usually requires user-defined parameters to set the strength of the smoothing and type of kernel to be used. Experimental results show that our method more accurately detects nucleosome positions as compared to existing software tools, in particular when complex nucleosome configurations are present in the data. On the human malaria parasite data produced by our collaborators PuFFIN detected stronger associations between nucleosome occupancy and gene expression levels compared to other tools, which indicates that our tool extracts more biologically-relevant features from the data.

Chapter 4

ThIEF: a Novel Tool for Tracking Genomic Features

Recent advancements in high-throughput DNA sequencing technology has led to the rapid decrease in the cost associated with sequencing. This has enabled life scientists to carry out increasingly large-scale experiments. For instance, in the context of epigenetics, it is now relatively affordable to run multiple genome-wide experiments: for example, one can take "snapshots" of nucleosome levels at different time points during a particular cell cycle. As a result, this has opened the possibility of exploring nucleosome dynamics. From an analytic point of view, the associated question is how to compare multiple genome-wide nucleosome maps, either for evolutionally-related organisms, or for the same organism at different conditions/time points. Similar problems arise in genomics and epigenetics to analyze other genomic features that can change over time, e.g., transcription factor binding events, DNA methylation, among others.

In this Chapter we focus on the general problem of comparing multiple genome-wide "genomic-feature" maps. Arguably, the most natural way to compare nucleo-

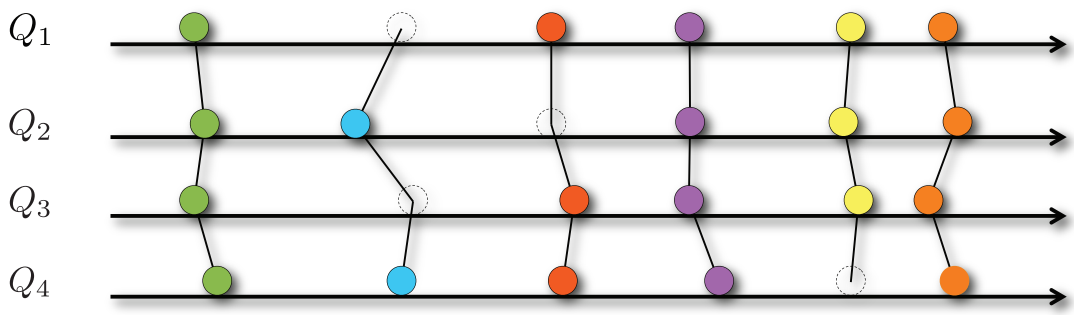

Figure 4.1: An illustration of the genomic-feature map aligning problem

some/feature maps is to align them in a similar way we align DNA sequences: we put multiple nucleosome/feature maps on top of each other, with the objective to "track" the trajectory of each individual nucleosomes/feature across time, in a way that some total cost (i.e., total traveled distance in the case of nucleosomes) is minimized. We call such trajectory an *alignment*: similarly to multiple sequence alignment we could have "insertion" or "deletions" of nucleosomes/features at specific time points.

4.0.1 Problem definition

We define a *genomic-feature map* as a set of genomic features $Q = \{f_1, \ldots, f_n\}$, where each feature $f \in Q$ is a vector $f = (\mu^{(f)}, a_1^{(f)}, \ldots, a_l^{(f)})$ with $l + 1$ components, where $\mu^{(f)}$ is the genomic coordinate of that feature in the genome (i.e., chromosome number and position in the chromosome) and each $a_j^{(i)}, j = 1..l$ is an *attribute* of that feature (e.g., confidence score of a nucleosome, level of DNA methylation, strenght of trascription factor binding, "fuzziness" of a nucleosome, etc.).

Given two genomic-feature maps Q_1 and Q_2, the goal is to align them. Specifically, each feature $i \in Q_1$ will be either matched to a feature $j \in Q_2$ so that we minimize the cost $\Delta(i, j)$ (e.g., the distance $|\mu^{(i)} - \mu^{(j)}|$) or we will report that feature $i \in Q_1$ has no match in Q_2 (insertion/deletion). In this latter case we will say that j is a *gap* and denote it as \$. In the general case where we need to align $k > 2$ maps, the cost function will take form $\Delta(q_1, q_2, \ldots, q_k)$. Figure 4.1 shows a simple example where we are supposed to align four maps: circles represent features to align; dashed circle mark the gap; matched features are connected with solid lines.

In order to define the problem more precisely, we need first to set some constraints.

Assumption 1 *The order of aligned features should be preserved across different maps. Formally: if we have order $\mu_1^{(\psi)} < \mu_2^{(\psi)} < \cdots < \mu_k^{(\psi)}$ of locations on map ψ, then we should have the same order for matching features $\mu_1^{(\phi)} < \mu_2^{(\phi)} < \cdots < \mu_k^{(\phi)}$ in another map ϕ.*

4.1 Previous work

Despite the fact that the map alignment problem should arise in several applications of genomics and epigenetics, we could not find any available/published tools aimed to solve it. We speculate that when faced with this problem, computational biologists use a variation of what we call the "naive approach", which uses a sliding window and a greedy strategy. We will discuss the naive method in the following subsection. We should also mention that there exists a set of algorithms that aim to align microscopic images of cell samples with the goal to track how those cells move and divide (see, e.g., [59, 25, 58]). In our application, however, we are interested in only one-to-one matchings (i.e., no cell-division). Moreover these methods usually do not handle multiple maps, but focus on two successive snapshots.

The problem of aligning genomic features from multiple maps is similar to the well-known problem of *Multiple-Sequence Alignment* (MSA) (see, e.g., [17]). MSA is a central problem in bioinformatics: it has a wide range of application, from phylogenetic tree reconstruction, similarity between transcription factor binding sites, discovery of protein domains, etc. (see, e.g., [63, 57, 39, 47]). A very large corpus of literature has been published on MSA and its applications. The majority of efficient methods employ sophisticated heuristics, since the global optimization problem of aligning long sequences is computationally costly (the problem is *NP*-complete). Heuristic methods include progressive alignment construction [27, 73, 72, 41], iterative methods, hidden Markov models, genetic algorithms [52, 51], simulated annealing [38, 30], etc.

A direct solution for MSA uses the dynamic programming to identify the globally optimal alignment. The input to MSA is two or more DNA sequenced to be aligned, a gap penalty score δ, and a substitution matrix which assigns the cost of aligning each possible pair of symbols in the alphabet. For k individual sequences, the dynamic programming algorithm requires one to construct a k-dimensional dynamic programming matrix that stores the cost of optimally aligning any prefix (or suffix) of the k sequences. The running time for aligning k sequences of size n is $\Theta(n^k)$ and the required space is $\Theta(n^k)$ (space can be reduced using divide-and-conquer). As said, if one wants to find the global optimum, the problem has been shown to be *NP*-complete [18, 76, 34]. To speed up the algorithm different bounding techniques could be used (see e.g., [16, 43, 7]). There is an obvious equivalence between the genomic-feature map alignment problem and MSA: the major difference is that instead of a substitution matrix, we use a cost function $\Delta(\cdot)$. However, due to possible long gaps in the alignment, several of the proposed heuristics to speed up the execution of MSA cannot be used for our problem.

The problem of aligning genomic feature can be also represented as *multi-target tracking* problem or *data association* problem. These problems have been known for decades and are extensively studied. These problems arise when there is a need to track features on video sequences, radar scans, etc. The main computational challenge is to overcome scalability when dealing with a large number of snapshots and a relatively small number of objects to track. In our case, we are interested in dealing with a relatively small number of maps but a large number of targets (features). As a result, we initially designed an algorithmic solution from the ground up by representing *multi-target tracking* problem as a *k-partite matching* problem.

In this Chapter we propose an efficient tool to solve the problem of aligning multiple genomic-feature maps. We will focus on solving the *k-partite matching* problem, which is known to be NP-complete for $k > 2$. We provide two algorithms: the first builds iterative approximations of the optimal alignment, the second computes the solution via integer linear programming [8]. To make the second approach feasible to real-world-sized data we applied branch-and-bound technique to reduce the size of the problem. Then we relax the problem to a linear program, which we solve using the off-the-shelf solver GLPK [44].

4.2 Methods

4.2.1 Naive Greedy approach

We first describe the naive (greedy) approach, and its variations thereof. As said, we believe that this is the most commonly used approach, and we will use it as the baseline for our performance evaluations.

In the naive algorithm, for each feature at location i in the first map we try to find the closest match on the other maps such that all matching candidates are located within a window of predefined length w, centered at i. If for some map there

```
Input: $\{Q_1, Q_2, \ldots, Q_k\}$ - a set of $l$ genomic feature maps to align, $w$ -
       size of the window
Output: $R$ - a set of alignments, where each alignment is an $l$-tuple
        such that each $i$-th element ($i = 1..k$) of a tuple is a feature
        either belonging to $i$-th genomic map or is a gap (marked as
        \$). Each genomic feature from $j$-th map is present exactly
        once and only at $j$-th position of the alignment tuple
$R \leftarrow \emptyset$;
for $i = \{1..k\}$ do
    for $\forall f \in Q_i$ do
        $r = (\$_1, \ldots, \$_{i-1}, f, \cdot, \ldots, \cdot)$ - is resulting tuple of size $k$
        for $j = (i+1)..k$ do
            $\Psi \leftarrow$ set of features $\psi$ $s.t.$ $\psi \in Q_j \wedge |\mu_\psi - \mu_f| \leq w$
            if $\Psi = \emptyset$ then
                $r_j \leftarrow \$$
            else
                $\psi_{min} \leftarrow argmin_{z \in \psi} \Delta(f, z)$
                $r_j \leftarrow \psi_{min}$
                $Q_j \leftarrow Q_j \setminus \{\psi_{min}\}$
            end
        end
        $R \leftarrow R \cup r$
    end
end
```

Figure 4.2: The naive (greedy) approach for aligning genome feature maps

are no candidate features in the window, we introduce a gap. If a feature is matched in constructing an alignment, the algorithm marks it as "used" (so that it is not assigned to other alignments). After we process the first map, we move to the next map: we check whether any feature were not marked "used"; if any is still unmatched the same sliding window approach is employed. We repeat the method until all the features are marked as used. A detailed pseudo-code is shown in Figure 4.2.1.

The naive algorithm is quite fast (running time is $\Theta(n^2)$), but it does not guarantee to produce an optimal solution. First, it relies on the assumption that the alignment score of three or more features is decomposable into the combination of pair-wise alignments. Secondly, the algorithm is very sensitive to the choice of window size

w. On Figure 4.3 we provide an example of what could happen if the “wrong” window size is used. In the example, we align four feature maps: circles represent a feature to align (solid circle - feature is present, dashed circle - feature is missing and not part of the input), black lines represent true matchings that we are expected to recover. Let consider one step of the naive algorithm: when it considers the “red” feature, it will try to match it with features marked with thick-line circles. The resulting alignment is shown in green: observe that this alignment is sub-optimal.

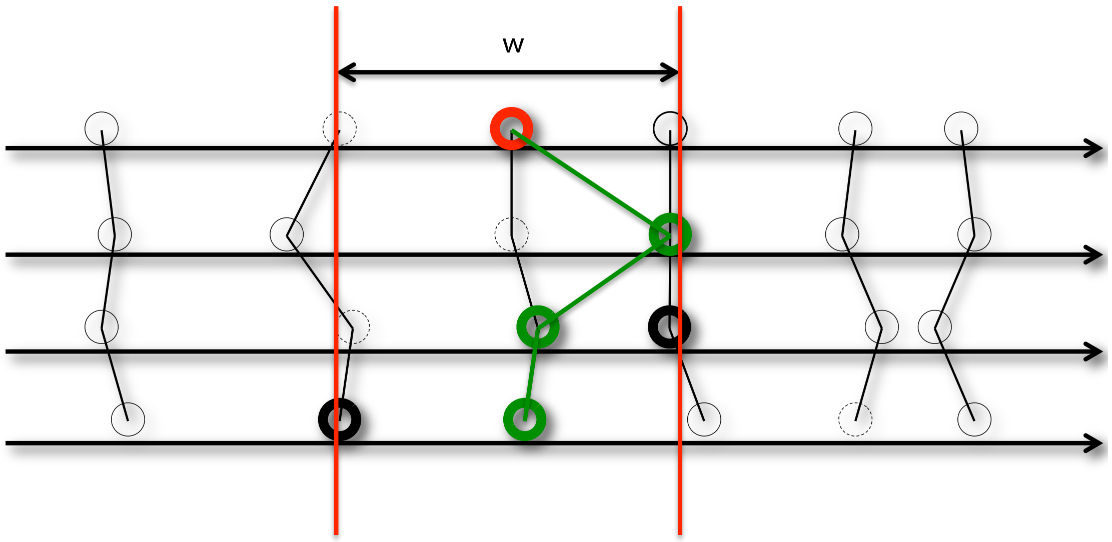

Figure 4.3: An illustration of the naive (greedy) method creating a sub-optimal alignment

4.2.2 ThIEF:Iterative algorithm

In this subsection we present our first efficient algorithm, called ThIEF:Iterative. As the name suggests, it is an iterative algorithm, which constructs alignments by processing input maps pairwise. The algorithm starts by aligning the first two maps by solving the *weighted bipartite matching* problem. Then, the algorithm aligns the resulting alignment between the first two maps with the third map. At each iteration the algorithm solves a bipartite matching problem between the current alignment and the next feature map.

Each node is assigned a genomic location. When a genomic feature f is mapped to a node $v_f \in V$, then location of v_f is μ_f. Since the algorithm needs to align alignment to maps, we need to extend the notion of "location" to alignments: in this case, we simply average the coordinates of the features that belong to an alignment. In this way, nodes corresponding to partial alignments will have location attribute and a cost function can be evaluated. The weight for an edge (w, v) is computed as the distance between nodes $c_{(w,v)} = \Delta(\mu_w, \mu_v)$.

The bipartite graph has to have an equal number of vertices in each partition, so for every vertex in one partition we introduce a "dummy" vertex in the other one. The resulting bipartite graph allow us to reduce the matching problem to $n-to-n$ assignment problem. The presence of dummy nodes also naturally incorporates gap resolution: when a feature is matched against a dummy node we are implicitly introducing a gap. The cost of edges connecting "dummy" node to features is the gap penalty δ, and edges "dummy"-"dummy" are not allowed. To solve the $n - to - n$ assignment problem we use the *Hungarian algorithm* [32] , which has $\Theta(n^3)$ time complexity. Details on ThIEF:Iterative algorithm are shown on Figure 4.4.

The total running time of this approach is $\Theta(km^3)$, where k is the number of maps and m is an upper bound on the number of alignments. In the worst case, if we aligning maps such that features on each map are aligned with corresponding gaps then we could have $m \in O(2^{k-1}n)$. In practice, however, we have $m \in O(n)$. The other disadvantage of this approach is that it does not guarantee optimality, unless cost function Δ is decomposable into sum of pairwise costs. The order in which maps are processed can give rise to different (sub-optimal) solutions.

```
Input: $\{Q_1, Q_2, \ldots, Q_k\}$ - a set of $k$ genomic feature maps to align, $\delta$ -
        penalty for missing feature
Output: $R$ - a set of alignments, where each alignment is an $l$-tuple
         such that each $i$-th element ($i = 1..k$) of a tuple is a feature
         either belonging to $i$-th genomic map or is a gap (marked as
         \$). Each genomic feature from $j$-th map is present exactly
         once and only at $j$-th position of the alignment tuple
$R \leftarrow \emptyset$
for $\forall f \in Q_1$ do
    $r \leftarrow (f, \cdot, \ldots, \cdot)$ - tuple of size $k$
    $R \leftarrow R \cup r$
end
for $\forall j \in \{2, \ldots, k\}$ do
    Solving m-to-n assignment problem
    $X \leftarrow \emptyset \wedge Y \leftarrow \emptyset$
    for $\forall r \in R$ do
        $X \leftarrow X \cup \{\text{average location of } r\}$
        $Y \leftarrow Y \cup \{\$\}$
    end
    for $\forall f \in Q_j$ do
        $Y \leftarrow Y \cup \{\mu_f\}$
        $X \leftarrow X \cup \{\$\}$
    end
    $M \leftarrow$Hungarian Algorithm$(X, Y)$, with matching costs
    $\Delta(\$, \cdot) = \Delta(\cdot, \$) = \delta$, $\Delta(\$, \$) = \infty$, otherwise $\Delta(x, y) = |x - y|$;
    for $m = (a, b) \in M$ do
        if $a \neq \$$ then
            corresponding to $a$ track $r \leftarrow (r_1, \ldots, r_j, b, \cdot, \ldots, \cdot)$
        else
            $R \leftarrow R \cup ([\$]^{(j)}, b, \cdot, \ldots, \cdot)$
        end
    end
end
return $R$
```

Figure 4.4: Sketch of ThIEF:Iterative

4.2.3 ThIEF:LP Linear Programing Solution

Next, we implemented an optimal algorithm called ThIEF:LP, which casts the alignment problem as *k-partite matching* problem. Our problem is slightly more general than the k-partite matching problem because we need to deal with gaps.

First we build a hyper-graph $H = (V, E)$, where each vertex $v \in V$ represent a genomic feature, and a hyper-edge $e \in E$ connects a subset of vertices (i.e., $e \subseteq V$)

representing a possible alignment. By construction, the graph is k-partite: each hyper-edge contains at most one vertex from a partition (map). Allowing hyper-edges to “skip” partitions allows us to model gaps in the alignment.

We build the graph H iteratively: we start from the vertices in the first map/partition, then we add the nodes new partition and “refine” the set of possible hyper-edges. To do so, for every hyper-edge $e \in E$ we consider its extension using vertices in new partitions: if the extension using vertex v is feasible (see next paragraph) then we add the extended hyper-edge $e_* = e \cup \{v\}$ to E. In addition we add new hyper-edge $e_{new} = \{v\}$. Observe that when adding a feature map the number of vertexes grows linearly, but the number of possible hyper-edges grows exponentially.

To limit the size of the graph H we use the following heuristics. First, we do not generate any “crossing” hyper-edges (i.e., that hyper-edges that violate assumption (1) above). Second, we filter out hyper-edges that connect features that are located further than 2δ. Otherwise, such hyper-edge could be split into two hyper-edges with smaller total cost.

The outline of ThIEF:LP is shown on Figure 4.5.

4.3 Experimental Results

To generate synthetic feature maps we proceeded as follow. Each dataset depends on five parameters, namely (1) the number k of feature maps, (2) the number n of features, (3) the minimum distance d between features, (4) the probability p of a gap, and (5) the movement σ. The algorithm generates an initial map Q where the features are placed at random locations so that the average distance between adjacent features is uniformly distributed in $[d, 2d]$.

```
Data: $\{Q_1, Q_2, \ldots, Q_k\}$ - a set of $k$ genomic feature maps to align, $\delta$ -
      penalty for missing feature
Result: $R$ - a set of hyperedges, where each edges is an $l$-tuple such that
        each $i$-th element ($i = 1, \ldots, k$) of a tuple is a feature either
        belonging to $i$-th genomic map or is a gap (marked as \$). Each
        genomic feature from $j$-th map is present exactly once and only
        at $j$-th position of the alignment tuple.
$E_1 \leftarrow \emptyset$
$V \leftarrow \cup_{f \in \{Q_1, \ldots, Q_l\}} f$
for $\forall f \in Q_1$ do
    $E_1 \leftarrow E_1 \cup (f, \cdot, \ldots, \cdot)$
end
for $\forall j = 2, \ldots, l$ do
    $E_j \leftarrow \emptyset$
    for $\forall f \in Q_j$ do
        $E_j \leftarrow E_j \cup ([\$]^{(j-1)}, f, \cdot, \ldots, \cdot)$
        for $\forall e \in E_{j-1}$ s.t. $distance(e, f) \leq 2\delta$ do
            $e_j \leftarrow f$
            $E_j \leftarrow E_j \cup \{e\}$
        end
    end
end
$E \leftarrow E_l$
for $e \in E$ do
    $e_{cost} \leftarrow \Delta((e_1, \ldots, e_l))$
end
Create LP for the problem of finding minimum weight hyperedge cover
for $H = (V, E)$
Solve LP using GLPK
$R \leftarrow$ resulting set of hyperedges
```

Figure 4.5: Sketch of THIEF:LP algorithm

From the initial map Q we generated k maps as follows. For each map we take a feature $f \in Q$ and we shift its location a random quantity drawn from Gaussian distribution with parameters $(0, \sigma)$. By keeping track of which feature moved where during these step we can establish the "ground-truth" alignment set. The final step is to scan all the features and, with probability p, replace them with a gap.

Observe that this procedure can produce alignments that do not satisfy the assumptions from Section 4.1.1, in particular, trajectories of alignments could cross.

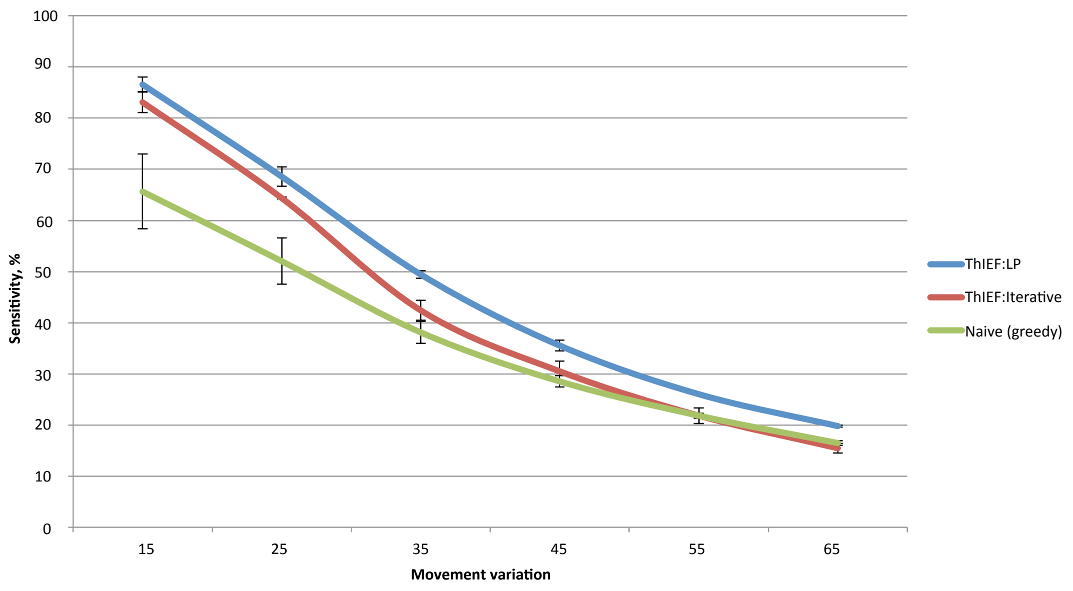

Figure 4.6: Percentage of true alignments recovered for several choices of movement parameter σ

4.3.1 Performance Analysis

We analyzed the performance of ThIEF:LP and ThIEF:Iterative against the naive (greedy) approach on synthetic datasets. We generated a large number of datasets consisting of $k = 3$ maps and $n = 1000$ features, using different values for minimum distance d between features, gap probability p and movement variation σ.

We compared the alignments produced by these tools against the "ground-truth" and measured the percentage of recovered alignments (sensitivity) and the proportion of recovered alignments in the output (specificity). Figure 4.6 shows the average sensitivity as a function of the movement parameter σ. Observe that ThIEF:LP outperforms the other approaches, as we would expect given that ThIEF:LP is guaranteed to generate optimal solutions. ThIEF:Iterative has slightly worse performance but still better then naive approach. Also observe that as σ increases, the performance decreases: this could be explained by the fact that with more variation in the location

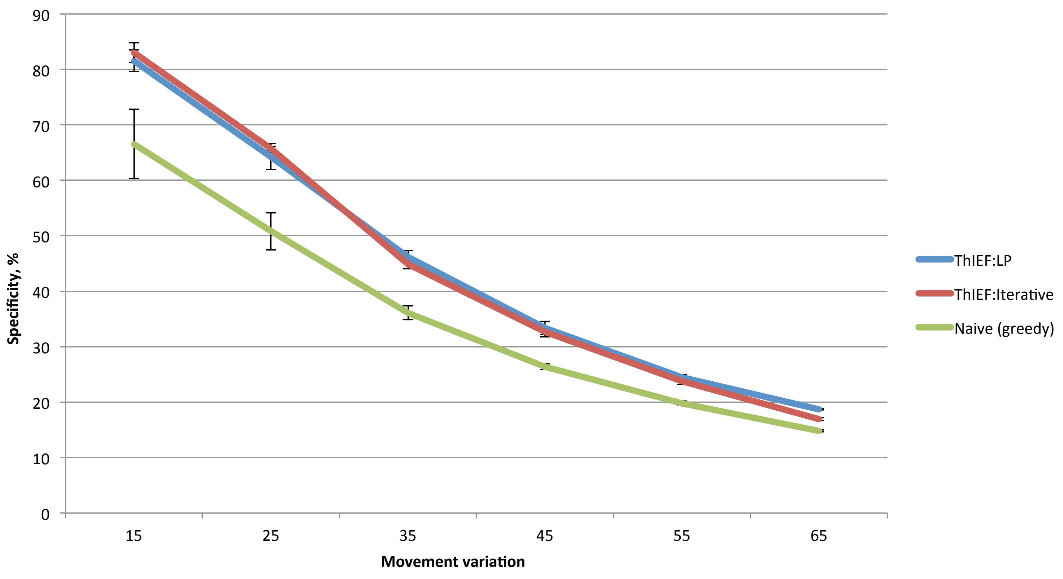

Figure 4.7: Percentage of true positive alignments in the output recovered for several choices of movement parameter σ

of the features it becomes more likely to have crossing of alignment trajectories, which none of the tools was designed to capture. Specificity analysis shows a similar behavior (see Figure 4.7). Here ThIEF:LP and ThIEF:Iterative have almost identical performance, which is significantly better than the naive.

4.3.2 Execution Time

To study the speed of the three algorithms we measured the cumulative execution time on a variety of input datasets.

First, let's consider ThIEF:Iterative. The theoretical running time is $\Theta(km^3)$, where k is the number of maps and m is an upper bound on the total number of alignments. The functional dependency between m and n is data-dependent, specifically how many new alignments are introduced at every iteration (instead of extending existing ones). The worst case is $m \in O(2^{k-1}n)$. As a result, the total worst-case time-complexity could be as bad as $O(2^k n^3)$. Figure 4.8 shows the experimental dependency between the execution time of ThIEF:Iterative and the number of maps k. Observe that since the Y axis is log-scale, these experiments confirm that the actual running time is exponential. Figure 4.9 shows the dependency between the execution time of ThIEF:Iterative and the number of features n, for $k = 5$ and $k = 6$ maps. The solid lines are cubic functions of n fitted to the data. These experiments confirm the cubic dependency on number of features of the maps in the input.

Next, we consider ThIEF:LP. The running time is dominated by the cost of solving a linear program, which in the case of the simplex algorithm, has exponential worst-case running time (although in practice, for the large majority of the instances the time-complexity of simplex is polynomial). The size of the linear program depends on the size of the hyper-graph: in our implementation, the size of the graph can be exponential in k, that is $O(na^k)$, where $a > 1$ is a constant. Figure 4.10 shows the dependency between execution time and the number of maps k. Each curve shows a linear trend (note that Y axis is in log-scale). The different shape of the blue curve (twenty features per map) could be explained by the fact that with small inputs the I/O overhead of transferring the data to the GLPK solver dominates the execution time. When we consider the blue curve for at least five maps the size of the hyper-graph becomes big enough so that solving the linear program dominate the execution time. These experiments support the claim of exponential complexity on the number of maps. Figure 4.11 shows a linear dependency between the execution time and the number of features to track, as expected from the theoretical analysis.

Since ThIEF:LP was implemented in Python and ThIEF:Iterative in C++, comparing their execution time is not very meaningful.

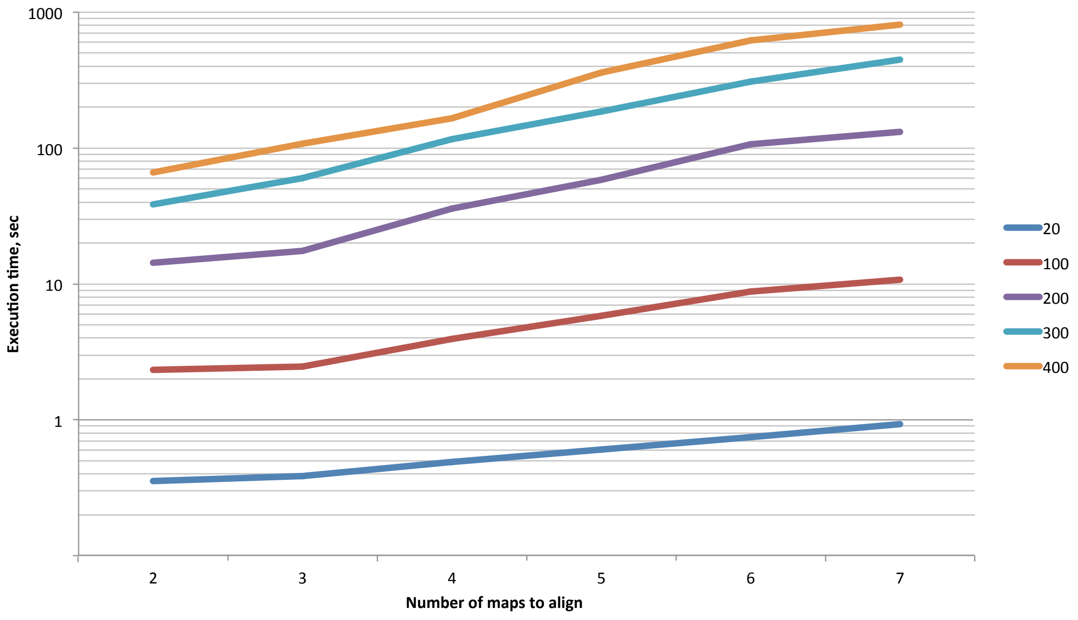

Figure 4.8: Dependency between execution time of THIEF:ITERATIVE and the number of maps to align, for different choices of the number of features.

4.4 Conclusion

In this Chapter we described the general problem of comparing multiple genome-wide “genomic-feature” maps in a framework similar to multiple sequence alignment. We implemented two novel tools to perform this task, called THIEF:LP and THIEF:ITERATIVE. THIEF:LP finds a global optimal solution by constructing a hyper-graph representing the problem and solves it via linear programming. THIEF:ITERATIVE reconstructs the final alignments by computing pair-wise alignments using the Hungarian algorithm: as a consequence solution the solution is not guaranteed to be optimal. We determined that THIEF:LP has slightly better sensitivity than THIEF:ITERATIVE, however the latter approach is more suitable for aligning a large number of maps. Both tools perform significantly better than naive (greedy) approach.

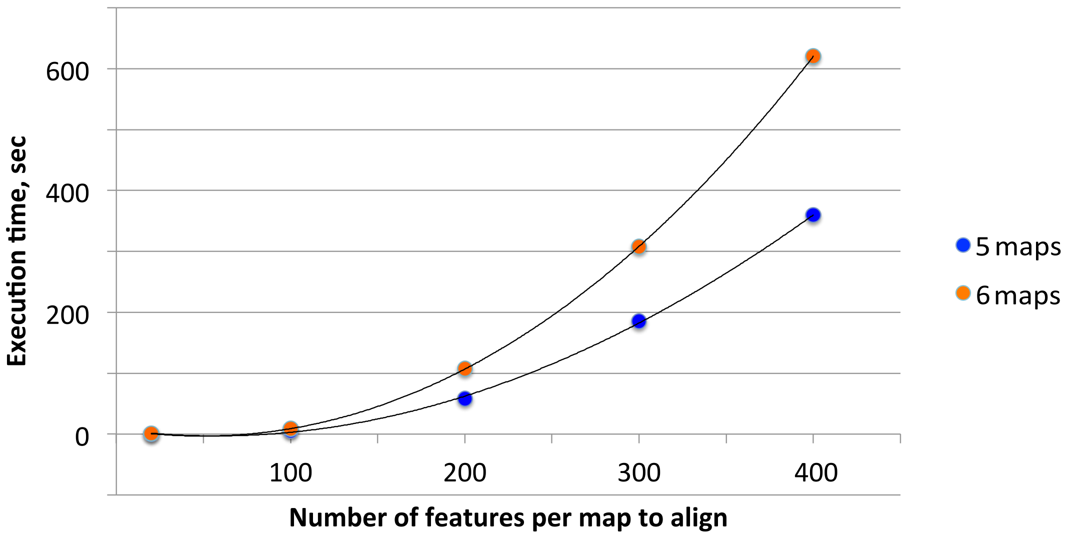

Figure 4.9: Dependency between execution time of THIEF:ITERATIVE and the number of features, for 5 and 6 maps

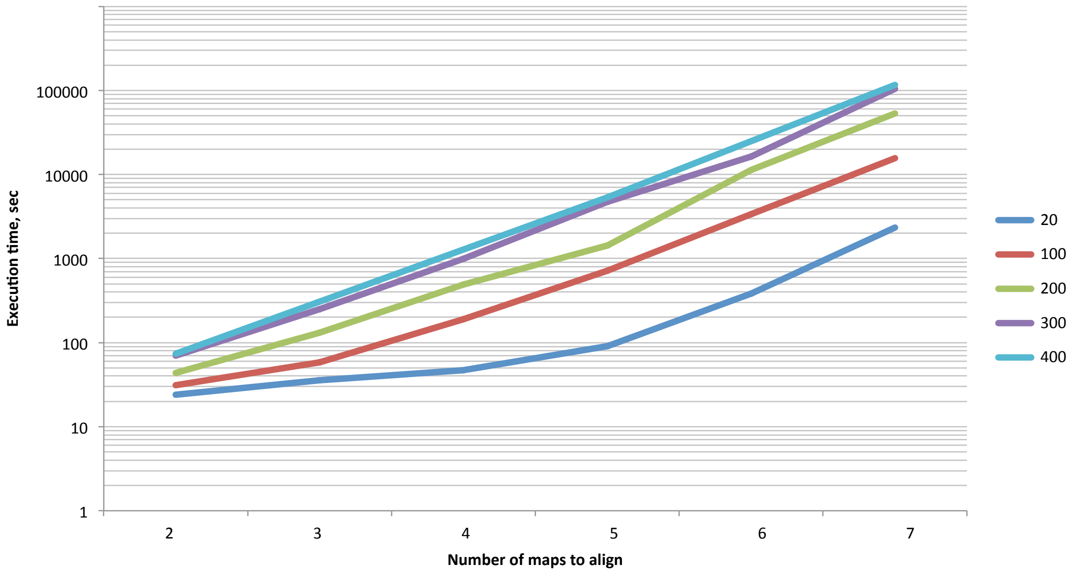

Figure 4.10: Dependency between execution time of THIEF:LP and the number of maps to align, for different choices of number of features.

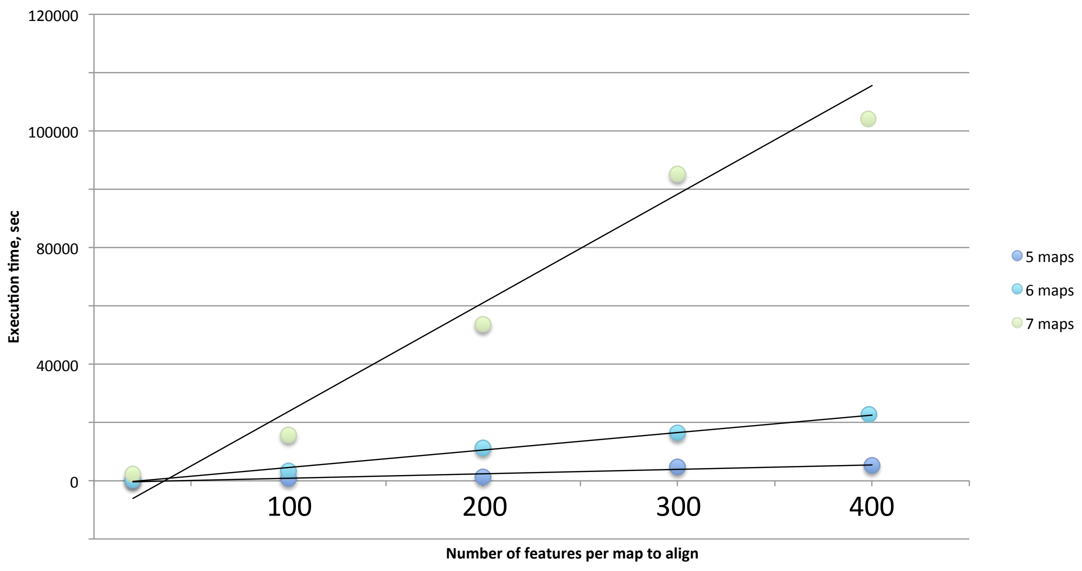

Figure 4.11: Dependency between execution time of THIEF:LP and the number of features, for 5 and 6 maps

Chapter 5

Conclusions

In this dissertation, we addressed some of the computational issues associated with the analysis of sequencing data enriched for nucleosomes. We proposed two novel algorithms to detect nucleosomes, for single- (NOrMAL) and paired-end (PuFFIN) sequencing data respectively. Then we examined the problem of aligning genomic feature maps. As a result we devised a novel framework THiEF that encompasses two algorithms THiEF:Iteration and THiEF:LP.

5.1 Publications

This dissertation includes two publications and one unpublished manuscript. The findings on NOrMAL was presented at ISMB 2012, Long Beach, CA and published in *Bioinformatics*. The work on PuFFIN was presented at RECOMB-SEQ 2014, Pittsburg, PA and will be published in *BMC Bioinformatics*.

Full list of publications by A. Polishko:

- "PuFFIN - A Parameter-free Method to Build Nucleosome Maps from Paired-end Reads" by **A. Polishko**, E. M. Bunnik, K. Le Roch and S. Lonardi, *BMC*

Bioinormatics (in press), 2014.

- "DNA-encoded nucleosome occupancy regulates transcriptional levels in the human malaria parasite Plasmodium falciparum", by E. Bunnik, **A. Polishko**, J. Prudhomme, N. Ponts, S. S. Gill, S. Lonardi and K. G. Le Roch, *BMC Genomics*, 15(1):347, 2014.

- "Mechanisms of small RNA generation from cis-NATs in response to environmental and developmental cues", by X. Zhang, Y. Lii, Z. Wu, **A. Polishko**, H. Zhang, V. Chinnusamy, S. Lonardi, J.-K. Zhu, R. Liu and H. Jin, *Molecular Plant*, 6(3): 704-715, 2013.

- "NOrMAL: accurate nucleosome positioning using a modified Gaussian mixture model." **A. Polishko**, N. Ponts, K. G. Le Roch, S. Lonardi. ISMB 2012 - Proceedings of Annual International Conference on Intelligent Systems for Molecular Biology, i242-i249, Long Beach, CA, 2012. Special Issue of *Bioinformatics*, 28(12): i242-i249, 2012. Presentation at ISMB 2012. Poster at WABI 2012.

- "ThIEF: a novel tool for TrackIng gEnomic Features", by **A. Polishko**, E. Bunnik, K. G. Le Roch, and S. Lonardi. (manuscript in preparation)

Bibliography

[1] Istvan Albert, Travis N Mavrich, Lynn P Tomsho, Ji Qi, Sara J Zanton, Stephan C Schuster, and B. Franklin Pugh. Translational and rotational settings of H2A.Z nucleosomes across the *Saccharomyces cerevisiae* genome. *Nature*, 446(7135):572–576, Mar 2007.

[2] James Allan, Ross M. Fraser, Tom Owen-Hughes, and David Keszenman-Pereyra. Micrococcal nuclease does not substantially bias nucleosome mapping. *Journal of Molecular Biology*, 417(3):152–64, January 2012.

[3] Richárd Bártfai, Wieteke A. Hoeijmakers, Adriana M. Salcedo-Amaya, Arne H. Smits, Eva Janssen-Megens, Anita Kaan, Moritz Treeck, Tim-Wolf W. Gilberger, Kees-Jan J. Françoijs, and Hendrik G. Stunnenberg. H2A.Z demarcates intergenic regions of the plasmodium falciparum epigenome that are dynamically marked by H3K9ac and H3K4me3. *PLoS pathogens*, 6(12), 2010.

[4] Jeremie Becker, Christopher Yau, John M. Hancock, and Christopher C. Holmes. NucleoFinder: a statistical approach for the detection of nucleosome positions. *Bioinformatics*, 29(6):711–716, March 2013.

[5] Evelien M Bunnik, Duk-Won Doug Chung, Michael Hamilton, Nadia Ponts, Anita Saraf, Jacques Prudhomme, Laurence Florens, and Karine G Le Roch. Polysome profiling reveals translational control of gene expression in the human malaria parasite plasmodium falciparum. *Genome biology*, 14(11):R128, 2013.

[6] Evelien M Bunnik, Anton Polishko, Jacques Prudhomme, Nadia Ponts, Sarjeet S Gill, Stefano Lonardi, and Karine G Le Roch. Dna-encoded nucleosome occupancy is associated with transcription levels in the human malaria parasite plasmodium falciparum. *BMC Genomics*, 15(1):347, 2014.

[7] H Carrillo and D Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, 1988.

[8] Yuk Hei Chan and Lap Chi Lau. On linear and semidefinite programming relaxations for hypergraph matching. *Mathematical programming*, 135(1-2):123–148, 2012.

[9] Kaifu Chen, Yuanxin Xi, Xuewen Pan, Zhaoyu Li, Klaus Kaestner, Jessica Tyler, Sharon Dent, Xiangwei He, and Wei Li. DANPOS: dynamic analysis of nucleosome position and occupancy by sequencing. *Genome research*, 23(2):341–51, February 2013.

[10] Ramu Chenna, Hideaki Sugawara, Tadashi Koike, Rodrigo Lopez, Toby Gibson, Desmond Higgins, and Julie Thompson. Multiple sequence alignment with the clustal series of programs. *Nucleic acids research*, 31(13):3497–3500, 2003.

[11] Hyungwon Choi, Alexey Nesvizhskii, Debashis Ghosh, and Zhaohui Qin. Hierarchical hidden markov model with application to joint analysis of chip-chip and chip-seq data. *Bioinformatics*, 25(14):1715–1721, 2009.

[12] H Chui. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2-3):114–141, 2003.

[13] Hope A Cole, Bruce H Howard, and David J Clark. The centromeric nucleosome of budding yeast is perfectly positioned and covers the entire centromere. *Proceedings of the National Academy of Sciences*, 108(31):12687–12692, 2011.

[14] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 2, pages 1322–1328. Ieee, 1999.

[15] Hugh Durrant-whyte and Tim Bailey. Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *History*, pages 1–9, 2006.

[16] Robert Edgar. Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC bioinformatics*, 5(1):113, 2004.

[17] Robert C Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–7, January 2004.

[18] Isaac Elias. Settling the intractability of multiple alignment. *Journal of Computational Biology*, 13(7):1323–1339, 2006.

[19] Yair Field, Yvonne Fondufe-Mittendorf, Irene Moore, Piotr Mieczkowski, Noam Kaplan, Yaniv Lubling, Jason Lieb, Jonathan Widom, and Eran Segal. Gene expression divergence in yeast is coupled to evolution of dna-encoded nucleosome organization. *Nat Genet*, 41(4):438–445, 2009.

[20] Yair Field, Noam Kaplan, Yvonne Fondufe-Mittendorf, Irene K Moore, Eilon Sharon, Yaniv Lubling, Jonathan Widom, Eran Segal, and Uwe Ohler. Distinct modes of regulation by chromatin encoded through nucleosome positioning signals. *PLoS Computational Biology*, 4(11):e1000216, Nov 2008.

[21] Oscar Flores and Modesto Orozco. nucleR: a package for non-parametric nucleosome positioning. *Bioinformatics (Oxford, England)*, 27(15):2149–50, August 2011.

[22] Kai Fu, Qianzi Tang, Jianxing Feng, X Shirley Liu, and Yong Zhang. DiNuP: A Systematic Approach to Identify Regions of Differential Nucleosome Positioning. *Bioinformatics (Oxford, England)*, 28(15):1965–1971, June 2012.

[23] Paul Giresi, Jonghwan Kim, Ryan McDaniell, Vishwanath Iyer, and Jason Lieb. Faire (formaldehyde-assisted isolation of regulatory elements) isolates active regulatory elements from human chromatin. *Genome research*, 17(6):877–885, 2007.

[24] Paul Giresi and Jason Lieb. Isolation of active regulatory elements from eukaryotic chromatin using faire (formaldehyde assisted isolation of regulatory elements). *Methods (San Diego, Calif.)*, 48(3):233–239, 2009.

[25] Steven Gold, Anand Rangarajan, et al. Softmax to softassign: Neural network algorithms for combinatorial optimization. *Journal of Artificial Neural Networks*, 2(4):381–399, 1996.

[26] Steven Gold, Anand Rangarajan, Chien-Ping Lu, Suguna Pappu, and Eric Mjolsness. New algorithms for 2d and 3d point matchingpose estimation and correspondence. *Pattern Recognition*, 31(8):1019–1031, 1998.

[27] Desmond Higgins and Paul Sharp. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1):237–244, 1988.

[28] DG Higgins, JD Thompson, and TJ Gibson. Using clustal for multiple sequence alignments. *Methods in enzymology*, 266:383–402, 1996.

[29] Peter Humburg, Chris Helliwell, David Bulger, and Glenn Stone. Chipseqr: analysis of chip-seq experiments. *BMC Bioinformatics*, 12(1):39, 2011.

[30] Masato Ishikawa, Tomoyuki Toya, Masaki Hoshida, Katsumi Nitta, Atushi Ogiwara, and Minoru Kanehisa. Multiple sequence alignment by parallel simulated annealing. *Comput. Appl. Biosci.*, 9(3):267–273, 1993.

[31] Hongkai Ji, Hui Jiang, Wenxiu Ma, David Johnson, Richard Myers, and Wing Wong. An integrated software system for analyzing chip-chip and chip-seq data. *Nature Biotechnology*, 26(11):1293–1300, 2008.

[32] Roy Jonker and Ton Volgenant. Improving the hungarian assignment algorithm. *Operations Research Letters*, 5(4):171–175, 1986.

[33] Raja Jothi, Suresh Cuddapah, Artem Barski, Kairong Cui, and Keji Zhao. Genome-wide identification of in vivo protein–dna binding sites from chip-seq data. *Nucleic Acids Research*, 36(16):5221–5231, 2008.

[34] W Just. Computational complexity of multiple sequence alignment with sp-score. *Journal of computational biology : a journal of computational molecular cell biology*, 8(6):615–623, 2001.

[35] Noam Kaplan, Irene Moore, Yvonne Fondufe-Mittendorf, Andrea Gossett, Desiree Tillo, Yair Field, Emily LeProust, Timothy Hughes, Jason Lieb, Jonathan Widom, and Eran Segal. The dna-encoded nucleosome organization of a eukaryotic genome. *Nature*, 458(7236):362–366, 2009.

[36] Rosa Karlić, Ho-Ryun Chung, Julia Lasserre, Kristian Vlahoviček, and Martin Vingron. Histone modification levels are predictive for gene expression. *Proceedings of the National Academy of Sciences*, 107(7):2926–2931, 2010.

[37] Peter Kharchenko, Michael Tolstorukov, and Peter Park. Design and analysis of chip-seq experiments for dna-binding proteins. *Nat Biotech*, 26(12):1351–1359, 2008.

[38] Jin Kim, Sakti Pramanik, and Moon Chung. Multiple sequence alignment using simulated annealing. *Comput. Appl. Biosci.*, 10(4):419–426, 1994.

[39] Ekaterina Kotelnikova, Vsevolod Makeev, and Mikhail Gelfand. Evolution of transcription factor dna binding sites. *Gene*, 347(2):255–263, 2005.

[40] Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat Meth*, 9(4):357–359, April 2012.

[41] MA Larkin, G Blackshields, NP Brown, R Chenna, PA McGettigan, H McWilliam, F Valentin, IM Wallace, A Wilm, R Lopez, JD Thompson, TJ Gibson, and DG Higgins. Clustal w and clustal x version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[42] Cheol-Koo Lee, Yoichiro Shibata, Bhargavi Rao, Brian D Strahl, and Jason D Lieb. Evidence for nucleosome depletion at active regulatory regions genome-wide. *Nature genetics*, 36(8):900–905, 2004.

[43] DJ Lipman, SF Altschul, and JD Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences*, 86(12):4412–4415, 1989.

[44] Andrew Makhorin. Glpk (gnu linear programming kit), 2008.

[45] T. N Mavrich, I. P Ioshikhes, B. J Venters, C Jiang, L. P Tomsho, J Qi, S. C Schuster, I Albert, and B. F Pugh. A barrier nucleosome model for statistical positioning of nucleosomes throughout the yeast genome. *Genome Research*, 18(7):1073–1083, Jul 2008.

[46] Travis N Mavrich, Cizhong Jiang, Ilya P Ioshikhes, Xiaoyong Li, Bryan J Venters, Sara J Zanton, Lynn P Tomsho, Ji Qi, Robert L Glaser, Stephan C Schuster, David S Gilmour, Istvan Albert, and B. Franklin Pugh. Nucleosome organization in the drosophila genome. *Nature*, 453(7193):358–362, May 2008.

[47] Alan Moses, Derek Chiang, Daniel Pollard, Venky Iyer, and Michael Eisen. Monkey: identifying conserved transcription-factor binding sites in multiple alignments using a binding site-specific evolutionary model. *Genome biology*, 5(12):R98, 2004.

[48] Felix Mueller-Planitz, Henrike Klinker, and Peter B Becker. Nucleosome sliding mechanisms: new twists in a looped history. *Nature structural & molecular biology*, 20(9):1026–32, September 2013.

[49] Nishanth Ulhas Nair, Avinash Das Sahu, Philipp Bucher, and Bernard M. E. Moret. Chipnorm: A statistical method for normalizing and identifying differential regions in histone modification chip-seq libraries. *PLoS ONE*, 7(8):e39573, 08 2012.

[50] Abhinav Nellore, Konstantin Bobkov, Elizabeth Howe, Aleksandr Pankov, Aaron Diaz, and Jun S Song. NSeq: a multithreaded Java application for finding positioned nucleosomes from sequencing data. *Frontiers in genetics*, 3(January):320, January 2012.

[51] C Notredame and DG Higgins. Saga: Sequence alignment by genetic algorithm. *Nucleic Acids Res.*, 24(8):1515–1524, 1996.

[52] C Notredame, EA O'Brien, and DG Higgins. Raga: Rna sequence alignment by genetic algorithm. *Nucleic acids research*, 25(22):4570–4580, 1997.

[53] E. Parzen. On estimation of a probability density function and mode. *Annals of mathematical statistics*, 33:1065–1076, 1962.

[54] Anton Polishko, Nadia Ponts, Karine G Le Roch, and Stefano Lonardi. NORMAL: accurate nucleosome positioning using a modified Gaussian mixture model. *Bioinformatics (Oxford, England)*, 28(12):i242–9, June 2012.

[55] Nadia Ponts, Elena Y. Harris, Stefano Lonardi, and Karine G. Le Roch. Nucleosome occupancy at transcription start sites in the human malaria parasite: A hard-wired evolution of virulence? *Infection, Genetics and Evolution*, 11(4):716–724, 2010.

[56] Nadia Ponts, Elena Y Harris, Jacques Prudhomme, Ivan Wick, Colleen Eckhardt-Ludka, Glenn R Hicks, Gary Hardiman, Stefano Lonardi, and Karine G Le Roch. Nucleosome landscape and control of transcription in the human malaria parasite. *Genome research*, 20(2):228–38, February 2010.

[57] Rainer Pudimat, Ernst-GÃnter Schukat-Talamazzini, and Rolf Backofen. A multiple-feature framework for modelling and predicting transcription factor binding sites. *Bioinformatics*, 21(14):3082–3088, 2005.

[58] Anand Rangarajan, Haili Chui, and Fred L Bookstein. The softassign procrustes matching algorithm. In *Information Processing in Medical Imaging*, pages 29–42. Springer, 1997.

[59] Anand Rangarajan, Alan Yuille, and Eric Mjolsness. Convergence properties of the softassign quadratic assignment algorithm. *Neural Computation*, 11(6):1455–1474, 1999.

[60] T Rausch, AK Emde, D Weese, A Döring, C Notredame, and K Reinert. Segment-based multiple sequence alignment. *Bioinformatics (Oxford, England)*, 24(16), 2008.

[61] Karine G Le Roch, Yingyao Zhou, Peter L Blair, Muni Grainger, J Kathleen Moch, J David Haynes, Patricia De La Vega, Anthony A Holder, Serge Batalov, Daniel J Carucci, and Elizabeth A Winzeler. Discovery of gene function by expression profiling of the malaria parasite life cycle. *Science*, 301(5639):1503–8, Sep 2003.

[62] Morten Beck Rye, Pål Sætrom, and Finn Drabløs. A manually curated chip-seq benchmark demonstrates room for improvement in current peak-finder programs. *Nucleic acids research*, 39(4):e25–e25, 2011.

[63] Rafik A Salama and Dov J Stekel. A non-independent energy-based multiple sequence alignment improves prediction of transcription factor binding sites. *Bioinformatics*, 29(21):2699–2704, 2013.

[64] Shin Sasaki, Cecilia C. Mello, Atsuko Shimada, Yoichiro Nakatani, Shin-Ichi Hashimoto, Masako Ogawa, Kouji Matsushima, Sam Guoping G. Gu, Masahiro Kasahara, Budrul Ahsan, Atsushi Sasaki, Taro Saito, Yutaka Suzuki, Sumio Sugano, Yuji Kohara, Hiroyuki Takeda, Andrew Fire, and Shinichi Morishita. Chromatin-associated periodicity in genetic variation downstream of transcriptional start sites. *Science (New York, N.Y.)*, 323(5912):401–404, January 2009.

[65] Dustin E Schones, Kairong Cui, Suresh Cuddapah, Tae-Young Roh, Artem Barski, Zhibin Wang, Gang Wei, and Keji Zhao. Dynamic regulation of nucleosome positioning in the human genome. *Cell*, 132(5):887–898, 2008.

[66] Robert Schopflin, Vladimir B. Teif, Oliver Muller, Christin Weinberg, Karsten Rippe, and Gero Wedemann. Modeling nucleosome position distributions from experimental nucleosome positioning maps. *Bioinformatics*, 29(19):2380–2386, 2013.

[67] Eran Segal, Yvonne Fondufe-Mittendorf, Lingyi Chen, AnnChristine Thåström, Yair Field, Irene K Moore, Ji-Ping Z Wang, and Jonathan Widom. A genomic code for nucleosome positioning. *Nature*, 442(7104):772–778, 2006.

[68] Li Shen, Ning-Yi Shao, Xiaochuan Liu, Ian Maze, Jian Feng, and Eric Nestler. diffreps: detecting differential chromatin modification sites from chip-seq data with biological replicates. *PloS one*, 8(6):e65598, 2013.

[69] Sushma Shivaswamy, Akshay Bhinge, Yongjun Zhao, Steven Jones, Martin Hirst, and Vishwanath R Iyer. Dynamic remodeling of individual nucleosomes across a eukaryotic genome in response to transcriptional perturbation. *Plos Biol*, 6(3):e65, Jan 2008.

[70] CA) Shozo Mori (Raytheon Systems Company Advanced CI Systems, San Jose. Multi-Target Tracking Theory in Random Set Formalism. 1998.

[71] Fabian Sievers, Andreas Wilm, David Dineen, Toby Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Soding, Julie Thompson, and Desmond Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular Systems Biology*, 7(1), 2011.

[72] Julie Thompson, Toby Gibson, and Des Higgins. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics / editoral board, Andreas D. Baxevanis ... [et al.]*, Chapter 2, 2002.

[73] Julie Thompson, Desmond Higgins, and Toby Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[74] Anton Valouev, Jeffrey Ichikawa, Thaisan Tonthat, Jeremy Stuart, Swati Ranade, Heather Peckham, Kathy Zeng, Joel a Malek, Gina Costa, Kevin McKernan, Arend Sidow, Andrew Fire, and Steven M Johnson. A high-resolution, nucleosome position map of C. elegans reveals a lack of universal sequence-dictated positioning. *Genome research*, 18(7):1051–63, July 2008.

[75] Anton Valouev, David S Johnson, Andreas Sundquist, Catherine Medina, Elizabeth Anton, Serafim Batzoglou, Richard M Myers, and Arend Sidow. Genome-wide analysis of transcription factor binding sites based on chip-seq data. *Nature methods*, 5(9):829–834, 2008.

[76] L Wang and T Jiang. On the complexity of multiple sequence alignment. *Journal of computational biology : a journal of computational molecular cell biology*, 1(4):337–348, 1994.

[77] Assaf Weiner, Amanda Hughes, Moran Yassour, Oliver J Rando, and Nir Friedman. High-resolution nucleosome mapping reveals transcription-dependent promoter packaging. *Genome research*, 20(1):90–100, January 2010.

[78] Elizabeth G Wilbanks and Marc T Facciotti. Evaluation of algorithm performance in chip-seq peak detection. *PloS one*, 5(7):e11471, 2010.

[79] Han Xu, Chia-Lin Wei, Feng Lin, and Wing-Kin Sung. An hmm approach to genome-wide identification of differential histone modification sites from chip-seq data. *Bioinformatics*, 24(20):2344–2349, 2008.

[80] Guo-Cheng Yuan and Jun S Liu. Genomic sequence is highly predictive of local nucleosome depletion. *PLoS computational biology*, 4(1):e13, 2008.

[81] Guo-Cheng Yuan, Yuen-Jong Liu, Michael F Dion, Michael D Slack, Lani F Wu, Steven J Altschuler, and Oliver J Rando. Genome-scale identification of nucleosome positions in s. cerevisiae. *Science*, 309(5734):626–630, 2005.

[82] Chongzhi Zang, Dustin Schones, Chen Zeng, Kairong Cui, Keji Zhao, and Weiqun Peng. A clustering approach for identification of enriched domains from histone modification chip-seq data. *Bioinformatics (Oxford, England)*, 25(15):1952–1958, 2009.

[83] Ken Zaret. Micrococcal nuclease analysis of chromatin structure. *Current protocols in molecular biology*, Chapter 21, February 2005.

[84] Xiaoming Zhang, Yifan Lii, Zhigang Wu, Anton Polishko, Huiming Zhang, Viswanathan Chinnusamy, Stefano Lonardi, Jian-Kang Zhu, Renyi Liu, and Hailing Jin. Mechanisms of small rna generation from cis-nats in response to environmental and developmental cues. *Molecular plant*, 6(3):704–715, 2013.

[85] Xuekui Zhang, Gordon Robertson, Sangsoon Woo, Brad G Hoffman, and Raphael Gottardo. Probabilistic inference for nucleosome positioning with MNase-based or sonicated short-read data. *PloS one*, 7(2):e32095, January 2012.

[86] Yong Zhang, Tao Liu, Clifford A Meyer, Jérôme Eeckhoute, David S Johnson, Bradley E Bernstein, Chad Nusbaum, Richard M Myers, Myles Brown, Wei Li, et al. Model-based analysis of chip-seq (macs). *Genome Biol*, 9(9):R137, 2008.

[87] Yong Zhang, Hyunjin Shin, Jun S Song, Ying Lei, and X Shirley Liu. Identifying positioned nucleosomes with epigenetic marks in human from ChIP-Seq. *BMC genomics*, 9:537, January 2008.

[88] Zhenhai Zhang and B. Franklin Pugh. High-Resolution Genome-wide Mapping of the Primary Structure of Chromatin. *Cell*, 144(2):175–186, January 2011.