UNIVERSITY OF CALIFORNIA,
IRVINE

Toward a More Accurate Genome: Algorithms for the Analysis of High-Throughput Sequencing Data

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

William Jacob Benhardt Biesinger

Dissertation Committee:
Professor Xiaohui Xie, Chair
Professor Pierre Baldi
Professor Chen Li

2014

DEDICATION

To Emily, my helpmate forever.

TABLE OF CONTENTS

Page

LIST OF FIGURES **vi**

LIST OF TABLES **ix**

LIST OF ALGORITHMS **x**

ACKNOWLEDGMENTS **xi**

CURRICULUM VITAE **xiii**

ABSTRACT OF THE DISSERTATION **xvi**

1 Introduction **1**

2 AREM: Aligning Reads by Expectation-Maximization **6**

2.1 Introduction 6
2.1.1 Contributions 7
2.1.2 Chapter Outline 8
2.1.3 Related Work 9
2.2 Optimizing Peak Caller 9
2.2.1 Alignment 10
2.2.2 Peak Finding 10
2.2.3 Motif finding 11
2.3 A Probabilistic Framework for ChIP Read Realignment 12
2.3.1 Notations 13
2.3.2 Mixture model 13
2.3.3 Parameter estimation 15
2.3.4 Expectation-maximization algorithm 16
2.3.5 Implementation of E-M updates 17
2.4 Results 19
2.4.1 AREM identifies additional binding sites 21
2.4.2 AREM's sensitivity is increased with ambiguous reads 23
2.4.3 AREM is sensitive to repeat regions 23
2.5 Discussion 25

3 Discovering and Mapping Chromatin States Using a Tree Hidden Markov Model 27
3.1 Introduction . . . 27
3.1.1 Contributions . . . 29
3.1.2 Chapter Outline . . . 30
3.1.3 Related Work . . . 31
3.2 Tree Hidden Markov Model . . . 32
3.2.1 Model Description and Notation . . . 32
3.2.2 Parameters . . . 34
3.2.3 Model Description and Parametrization . . . 35
3.2.4 Incorporating Missing Markers and Hidden Cell Types . . . 37
3.2.5 Data Preprocessing . . . 38
3.3 Variational Inference . . . 39
3.3.1 Mean field (MF) variational inference . . . 41
3.3.2 Mean-field derivation and normalization . . . 42
3.3.3 Structured mean field(SMF) variational inference . . . 45
3.3.4 Structured mean-field derivation and normalization . . . 46
3.3.5 Loopy belief propagation (LBP) . . . 49
3.3.6 Parameter Learning . . . 50
3.4 Results . . . 51
3.4.1 Artificial Data . . . 51
3.4.2 Data Processing for ENCODE Dataset . . . 54
3.4.3 Comparing Approximate Inference Methods . . . 55
3.4.4 Model Complexity for Human ENCODE dataset . . . 56
3.4.5 TreeHMM on Complete Genomes using the SMF Approximation . . . 57
3.4.6 Comparison with ChromHMM . . . 59
3.5 Discussion . . . 61

4 Genomix: Scalable *de-novo* genome assembly 71
4.1 Introduction . . . 71
4.1.1 Contributions . . . 73
4.1.2 Chapter Outline . . . 73
4.1.3 Related Work . . . 73
4.2 Methods . . . 74
4.2.1 De Bruijn Graph Overview . . . 75
4.2.2 Notation . . . 77
4.2.3 Graph Building using Hyracks . . . 77
4.2.4 Remove Bad Coverage . . . 78
4.2.5 Graph Compression . . . 78
4.2.6 Tip Remove . . . 81
4.2.7 Bubble Merge . . . 81
4.2.8 Scaffolding . . . 83
4.3 Results . . . 88
4.4 Discussion . . . 92

5 Conclusion **94**

Bibliography **96**

LIST OF FIGURES

Page

2.1 **AREM workflow diagram.** Given a set of multiply-mapping and singly-mapping reads, AREM shifts stranded reads towards each other, identifies all *potential* peak regions, then iterates between updating read alignment probabilities and peak enrichment scores until convergence. Once converged, AREM optimizes the enrichment score further, seeking the peak width that gives the optimal enrichment score. Finally, the process is repeated for the control data in order to determine the overall false discovery rate. 20

2.2 **B-E** *de novo* discovery of motifs. From top to bottom: **B)** CTCF in MACS peaks from uniquely mapping reads, **C)** CTCF in AREM's peaks with multireads, **C)** Srebp-1 in MACS peaks from uniquely mapping reads and **D)** Srebp-1 in AREM's peaks with multireads. 22

2.3 Graphs displaying varying parameters and number of possible alignments per read. **A)** Total number of peaks discovered. **B)** Percentage of peaks with repetitive sequences. **C)** False Discovery Rate. **D)** Percentage of peaks with motif. 25

3.1 Example graphical model for a tree-structured HMM with three cell types. Hidden state variables representing chromatin states (white) are connected horizontally in a chain as well as vertically in a tree structure. Each chain in the graph represents a certain cell type. For example, the top chain represents the root cell type (e.g., ES cells). Observed nodes (grey) represent chromatin modifications and are connected only to the hidden variables. 33

3.2 **Lineage trees used for artificial data** 51

3.3 Free energy for approximate inference methods. Free energy for different inference methods are compared, with parameters learned using exact inference. The test dataset is restricted to a 2MB region of chromosome 22 with only three cell types and $K = 5$. The approximate methods use the parameters (learned by the exact method) and only perform inference steps. Note that for the exact algorithm (clique), the free energy equals the negative log-likelihood. 56

3.4 **Choosing model complexity (the optimal number of states K).** We ran the SMF variational EM algorithm on chromosome 22 with a range of K values . The final free energy for different K and the complexity-penalized likelihood score (BIC) is shown, revealing that a model with between 15 and 20 states are well-supported by the data. 57

3.5 Learned chromatin states with the associated chromatin modification and enrichment in distinct genomic regions - human data. Left panel: the probability of observing each histone mark in each of 18 hidden states is summarized. Right panel: fold enrichment of hidden states in various genomic regions reveals strong positional preferences of learned chromatin states. 65
3.6 Learned chromatin states with the associated chromatin modification and enrichment in distinct genomic regions - mouse data. Left panel: The probability of observing histone marks in each inferred state is summarized. Right panel: fold enrichment of hidden states in various genomic regions reveals strong positional preferences of chromatin states. The values are average enrichments across all 3 cell types. 66
3.7 Learned transition matrix α for the root cell type (H1 ES). 66
3.8 **Transition matrix θ shown as specific submatrices for each vertical parent state** Each sub-matrix is associated with vertical parent state 1 to 18 indicated at the top-left. A strong diagonal in the transition matrix indicates spatial persistence of states, while a strong vertical line indicates persistance from the corresponding vertical state across cell types. 67
3.9 **Comparison with ChromHMM.** Left panel: Emission probabilities learned by ChromHMM with $K = 18$. Right panel: Confusion matrix showing the total number of bases identified as belonging to a particular state by the two methods. In each cell, the top number is the log_{10}(number of bins) in the intersection, the value below corresponds to the percentage in the corresponding tree-HMM sites and is used to color the heatmap. (values in each column sum up to 1.) 68
3.10 **Learned transition matrix α for the root cell type - mouse dataset** 68
3.11 **Transition matrix θ shown as specific sub-matrices for each parent state - mouse dataset** Each sub-matrix is associated with vertical parent state 1 to 10 indicated at the top-left. 69
3.12 Hierarchical clustering of binarized histone marks in human. Branch lengths indicate the manhattan distance between species, considering all histone marks. The two most similar cell types are indicated in red, and the next two most similar are indicated in green. 69
3.13 **K-means clustering reveals different cell-type specificities for different chromatin states.** K-means clustering with 20 clusters was performed on the posterior probability of all the bins being in each state, posterior probability is shown as heatmap color, cell types are indicated at the top of each column. From left to right: Insulator/CTCF (state 14), Strong Enhancer (state 5), and Active Promoter (state 3). 70

4.1 The three types of overlap and the four corresponding edge types used to indicate the relationship between nodes. RF and FR edges are symmetric with themselves. 77

4.2 ***Compressing the graph***. **Top:** Simplified view of seven overlapping kmers. Nodes C, D, and E can be compressed into a single supernode since the pairs {C, D} and {D, E} have *degree* = 1 towards each other. **Middle-top:** Equivalent bidirected graph detailing edge types induced by kmer orientations. In the merge phase, D chooses "heads" and will subsume adjacent "tail" nodes C and E. **Center:** C and E update their neighbors to point to the subsuming node D. **Middle-bottom:** C and E merge towards D, and D uses their neighbor's edge type information to resolve its new edge types back towards the neighbors of C and E. The new kmer in CDE is now of length $len(\mathrm{C}) + len(\mathrm{D}) + len(\mathrm{E}) - (K - 1) \times 2$. To simplify graph updates, merged nodes maintain the key and orientation of the original "heads" nodes, in this case D. **Bottom:** Simplified view of the merged nodes. 80
4.3 **Node designations for scaffolding.** The walk starts at a seed node, indicated by s, and has already proceeded through walk nodes w_1 through w_3. The frontier node f makes the decision about which node to include next in the walk from the candidates c_1 and c_2. Ray's algorithm makes the inclusion decision based only on this information. Our improvements include searching beyond the candidates into complete paths which may include additional branches. Over the search distance d, c_1 has split into two paths p^1_{1*} and p^1_{2*}, whereas candidate c_2 still has only one search path, p^2_{1*}. The decision made in f about whether to include c_1 or c_2 can be augmented by the combined kmer from each path as well as read information from the paths. We omit edge types and edges that don't participate in the algorithm (e.g., there could be other edges incoming to one of the path nodes). 84
4.4 Runtime comparison as the size of the genome is increased while maintaining 100x coverage (synthetic data). These first three steps are take a substantial portion of the total runtime since all later steps operate on a greatly simplified graph. The scaffolding step may take much longer than these steps, depending on the search options used. 90

LIST OF TABLES

Page

2.1 **Comparison of peak-calling methods for cohesin and Srebp-1.** Three peak callers (MACS, SICER, and AREM) were run on both datasets. For AREM, the maximum number of retained alignments per read is varied (from 1 to 80). The total number of peaks and bases covered by peaks is reported as well as the FDR by swapping treatment and control. For both datasets, AREM's minimum enrichment score was fixed at 1.5 with 20 maximum alignments per read. For comparison, the motif background rate of occurence was 4.5% (CTCF) and 27% (Srebp-1) in 100,000 genomic samples, sized similarly to Rad21 MACS peaks and Srebp-1 MACS peaks, respectively. 26

3.1 Accuracy of different algorithms in recovering parameter values using artificial data. 53
3.2 H1-ES ChIP-seq enrichment in predicted promoter and enhancer regions. . . 60
3.3 **Average coverage and cell type-specific enrichment of learned chromatin states.** The "mean" column gives the average percent coverage, or the fraction of the segmented genome which is covered by the state. The following columns show the enrichment of each state in each cell type relative to the across-species mean (column 2). 62
3.4 Measured histone markers of Bone Marrow, G1E and G1E-ER4 - mouse dataset. 62

4.1 Assembly results for three bacterial genomes using Velvet, Ray, and Genomix. In all cases, the algorithms were run with $K = 31$ in single-end mode and all metrics are as reported by GAGE. Abbreviations are Inv. = Inversions, Rel. = Relocations, Trans. = Translocations. 91
4.2 Assembly results for three bacterial genomes using Velvet, Ray, and Genomix. In all cases, the algorithms were run with $K = 31$ in single-end mode and all metrics are as reported by GAGE. Abbreviations are Inv. = Inversions, Rel. = Relocations, Trans. = Translocations. 92

LIST OF ALGORITHMS

Page
1 Variational expectation-maximization algorithm 41
2 Strucutred Mean Field: Normalization Algorithm 48
3 Strucutred Mean Field: Forward Algorithm 48
4 Strucutred Mean Field: Backward Algorithm 49
5 Graph Building 78

ACKNOWLEDGMENTS

There are so many to thank for their support and guidance along this journey. First, I would like to thank my advisor, Xiaohui Xie. Without his optimism, excitement, encouragement, and an uncanny ability to turn a tough mathematical problem into mincemeat in seconds, I would not have gotten this far. Next, THANK YOU to my sweet and incredibly supportive wife, Emily. We were married in my first semester at UCI, and two kids and (hopefully) a PhD later, we are closer and happier than ever. My two boys have filled my evening with delight and lifted my heart many times after a hard day. Next comes *all* my parents, for filling my commutes with words of wisdom and for being supportive and helpful especially in the hard times.

I can't get away without thanking all the great members of the Xie lab, who made lab meetings fun, social, and delicious, and will continue to be my friends forever. If you ever need a place to stay, my couch is there for you! Come watch the little ones grow up! Special mention goes out to: Yi Li, the champion of mixture models, Elmira Forouzmand and Lingjie Weng, both of whom inherited large projects from me and spent weeks trying to wrap their heads around my code and who subsequently carried the projects off masterfully, despite working by themselves for large spans. Daniel Newkirk, who kept us all laughing and whose hobby of hoarding excellent hardware kept us all running despite too many Murphy's-law-failures. Daniel Quang gets the award for "Most Irate Rants About My Algorithm Not Working" while also quite possibly being the coolest person in the lab. Yifei Chen amazingly picked up python independent of my ranting about its superiority, and has put its awesome deep learning packages to good use. I miss Alvin Chon for the awesome ideas, vision, and general awesomeness, and Neil Infante for his good nature, hatred of my beloved python, and wife's cooking, and Mengfan Tang and Yuanfeng Wang for being the kindest, gentlest people I've met. Anbang Xu, Jianfeng Jia, and Nan Zhang all deserve my thanks for forming the framework and algorithms for the Genomix project, and making a stint to the second floor so much fun. Countless other friends have enriched the journey; I wish I could include you all here.

UCI has been a unique place, full of just the right mixture of expertise and easy-going attitudes, fostering excellent collaborations from people who both know what they're doing and are so happy to help. I'm sad that the IGB training grant evaporated since many collaborations started there and future students are surely going to miss out.

My beginnings at UCI were in Systems Biology and I owe many thanks to the scientific insight of the faculty there, with particular mention of the bureaucratic ninja skills of Karen Martin and the effective Socratic teaching method of Arthur Lander. So many professors here have shaped my understanding and perspective and have shared their contagious excitement for this field. Thank you.

For the AREM paper, I thank my co-authors, Daniel Newkirk and Alvin Chon, for releasing the paper under the excellent Creative Commons License, BMC Bioinformatics, and, finally, Tao Liu and the Liu lab for releasing MACS as open-source software. For the TreeHMM paper, my coauthor Yuanfeng Wang was a huge help, as well as Yifei Chen for some preliminary discussions. Elmira, Anbang, Jianfeng, Nan, Hongzhi Wang, Yingyi Bu, and Chen Li all deserve thanks for their help and support of the Genomix project.

My funding over this journey has been a lifesaver, and I owe it all in various parts to NSF grant DBI-0846218, NIH grant HD062951 National Institutes of Health/NLM bioinformatics training grant (T15LM07443).

CURRICULUM VITAE

William Jacob Benhardt Biesinger

EDUCATION

Doctor of Philosophy in Computer Science **2014**
University of California, Irvine *Irvine, California*

Bachelor of Science in Bioinformatics **2008**
Brigham Young University *Provo, Utah*

RESEARCH EXPERIENCE

Graduate Research Assistant **2008–2014**
University of California, Irvine *Irvine, California*

Undergraduate Research Assistant **2007–2008**
Brigham Young University *Provo, Utah*

Intern Research Assistant **2007, 2008**
City of Hope National Medical Center *Duarte, California*

TEACHING EXPERIENCE

Teaching Assistant **2009–2013**
University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

Integrative ChIP-seq/Microarray Analysis Identifies a CTNNB1 Target Signature Enriched in Intestinal Stem Cells and Colon Cancer PLOS One	**2014**
Biallelic genome modification in F0 Xenopus tropicalis embryos using the CRISPR/Cas system genesis	**2013**
Discovering and mapping chromatin states using a tree hidden Markov model BMC Bioinformatics	**2013**
Transcriptome-wide analyses of CstF64–RNA interactions in global regulation of mRNA alternative polyadenylation Proceedings of the National Academy of Sciences	**2012**
Genome-wide analysis of hepatic LRH-1 reveals a promoter binding preference and suggests a role in regulating genes of lipid metabolism in concert with FXR BMC Genomics	**2012**
AREM: aligning short reads from ChIP-sequencing by expectation maximization Journal of Computation Biology	**2011**
Genome-wide localization of SREBP-2 in hepatic chromatin predicts a role in autophagy Cell Metabolism	**2011**
Combined biological and computational approaches to understand the role of Get1/Grhl3 in epidermal differentiation Journal of Investigative Dermatology	**2011**
A risk variant in an miR-125b binding site in BMPR1B is associated with breast cancer pathogenesis Cancer Research	**2009**

REFEREED CONFERENCE PUBLICATIONS

Discovering and mapping chromatin states using a tree hidden Markov model **2013**
Research in Computational and Molecular Biology

AREM: aligning short reads from ChIP-sequencing by expectation maximization **2011**
Research in Computational and Molecular Biology

SOFTWARE

Genomix `https://github.com/uci-cbcl/genomix`
Scalable De Bruijn graph-based genome assembly using Hyracks and Pregelix.

TreeHMM `https://github.com/uci-cbcl/tree-hmm`
Graphical model for identifying chromatin states simultaneously in multiple cell types.

AREM `https://github.com/uci-cbcl/AREM`
ChIP-seq peak caller that re-aligns ambiguously aligning reads.

HTS-waterworks `https://github.com/uci-cbcl/HTS-waterworks`
Pipeline for high-throughput sequencing, including comparison of alignment, ChIP-seq and RNA-seq methods. Built on Ruffus.

Ruffus `https://github.com/bunbun/ruffus`
Python pipeline framework co-developed with Leo Goodstadt of the Wellcome Trust Centre for Human Genetics.

ABSTRACT OF THE DISSERTATION

Toward a More Accurate Genome: Algorithms for the Analysis of High-Throughput Sequencing Data

By

William Jacob Benhardt Biesinger

Doctor of Philosophy in Computer Science

University of California, Irvine, 2014

Professor Xiaohui Xie, Chair

High-throughput sequencing enables basic and translational biology to query the mechanics of both life and disease at single-nucleotide resolution and with breadth that spans the genome. This revolutionary technology is a major tool in biomedical research, impacting our understanding of life's most basic mechanics and affecting human health and medicine. Unfortunately, this important technology produces very large, error-prone datasets that require substantial computational processing before experimental conclusions can be made. Since errors and hidden biases in the data may influence empirically-derived conclusions, accurate algorithms and models of the data are critical. This thesis focuses on the development of statistical models for high-throughput sequencing data which are capable of handling errors and which are built to reflect biological realities.

First, we focus on increasing the fraction of the genome that can be reliably queried in biological experiments using high-throughput sequencing methods by expanding analysis into repeat regions of the genome. The method allows partial observation of the gene regulatory network topology through identification of transcription factor binding sites using Chromatin Immunoprecipitation followed by high-throughput sequencing (ChIP-seq). Binding site clustering, or "peak-calling", can be frustrated by the complex, repetitive nature of genomes. Traditionally, these regions are censored from any interpretation, but we re-enable their interpretation using a probabilistic method for realigning problematic DNA reads.

Second, we leverage high-throughput sequencing data for the empirical discovery of underlying epigenetic cell state, enabled through analysis of combinations of histone marks. We use a novel probabilistic model to perform spatial and temporal clustering of histone marks and capture mark combinations that correlate well with cell activity. A first in epigenetic modeling with high-throughput sequencing data, we not only pool information across cell types, but directly model the relationship between them, improving predictive power across several datasets.

Third, we develop a scalable approach to genome assembly using high-throughput sequencing reads. While several assembly solutions exist, most don't scale well to large datasets, requiring computers with copious memory to assemble large genomes. Throughput continues to increase and the large datasets available today and in the near future will require truly scalable methods. We present a promising distributed method for genome assembly which distributes the de Bruijn graph across many computers and seamlessly spills to disk when main memory is insufficient. We also show novel graph cleaning algorithms which should handle increased errors from large datasets better than traditional graph structure-based cleaning.

High-throughput sequencing plays an important role in biomedical research, and has already affected human health and medicine. Future experimental procedures will continue to rely on statistical methods to provide crucial error and bias correction, in addition to modeling expected outcomes. Thus, further development of robust statistical models is critical to the future high-throughput sequencing, ensuring a strong foundation for correct biological conclusions.

Chapter 1

Introduction

The advent of high-throughput sequencing has brought about a revolution in biology, forever changing the way we ask questions and seek answers in both basic and translational biology. It has enabled biologists to query the mechanics of both life and disease at unprecedented scale [55] and has transformed the scope of entire fields of biological study, taking them from small-scale experiments to "-omics" scale studies, that is, comprehensive functional surveys of all genes ("genomics" [75]), all proteins ("proteomics" [127]), and small molecules ("chemical genomics" [51]), and these fields' interactions in specific subfields such as toxicology ("toxicogenomics" [1]), metabolism ("metabolomics" [35]), sleep patterns ("circadiomics" [89]), etc, touching many subdisciplines of biological study. This incredible revolution has been possible through the ever-decreasing costs of sequencing and simultaneous increase in throughput and quality [104]. For comparison, Moore's law in computer hardware states that every two years, there is a doubling in the number of transistors on an integrated circuit [74]. High-throughput sequencing's number of sequenced base pairs per dollar cost has risen at at least that rate since 2001 and has been rising much faster than it since the end of 2007 [49]. In 2011, the rate of increase nearly shut down one of the major repositories tasked with hosting raw sequencing [32, 33], and in 2013, led to the first cancellation of an

XPRIZE (being “outpaced by innovation” [87]).

The effect of high-throughput sequencing on basic science and human health is already being felt today. Boyd et al. [17] review recent results stemming from high-throughput sequencing, including inroads in “Mendelian genetic disorders, hematologic cancer biology, infectious diseases, the immune system, transplant biology, and prenatal diagnostics.” Loman et al. [64] point out that high-throughput sequencing is now making the transition into the clinical laboratory, especially since smaller, more affordable benchtop sequencers are becoming available. In oncology and Mendelian diseases, high-throughput sequencing has been put to use in expression profiling, complete genome sequencing, and exome sequencing (including only the 1% of the genome that codes for protein), where the combination of this information leads to diagnosis of particular cancer subtypes [97, 118], reveals previously unsuspected cancer genes ([34] for review) and causes of Mendelian diseases [5] and has even led to personalized therapies [71, 81, 113]. In basic biology, high-throughput sequencing has given rise to a plethora of new experimental techniques that are able to reliably quantify single-nucleotide polymorphisms and structural variations [122], gene expression levels and their differences across samples [96, 114, 123], protein-DNA interactions [88, 137], and even DNA-DNA interactions [130]. Coupled with high-throughput sequencing, these new experimental methods fuel new discoveries through comprehensive inquiry.

As powerful as this technology is, it is not without its shortcomings. High-throughput sequencing reads must be put through rigorous quality control and correction since they tend to be shorter than the previous generation of Sanger reads, are more error-prone, and have several biases that can lead to incorrect conclusions if not properly accounted for [138]. Indeed, it is perhaps the rigorous statistical modeling of the data and experiments that has enabled high-throughput sequencing to become such a success, despite these major limitations. As new experimental procedures have been developed, new computational algorithms and statistical models were custom-built to account for particular experimental biases and describe expected outcomes. Largely thanks to high-throughput sequencing, in recent years the computational aspect of biology has become “the pillar of new biology” and is hardly recognizable as a separate science [86]. Richard Durbin, an early and major contributor to computational biology [27, 57, 59, 60], stated “ I would say that computation is now as important to biology as chemistry is” [2].

Within computational biology, probabilistic statistical models have proven particularly popular, given their robustness against noise in the data, communicability of the models, interpretability of the results, scalability with respect to data size, and their modular and expandable nature. Many major bioinformatic tools incorporate probabilistic models to: correct for sequencing errors in the data (see for example, [60, 68, 107, 121]); directly model biological phenomena such as gene regulatory networks ([67] for review), physical interactions within and between chromosomes [130]; discover sequence motifs in co-regulated genes [4, 112]; to discovery transcription factor binding sites ([76, 109, 133, 136, 137] among others); determine differentially expressed genes ([36, 114]); and for many other specific applications. Probabilistic modeling has even been proposed as a complete replacement for biology's scientific discovery process itself [38].

Advances in high-throughput sequencing's chemistry, throughput, and cost aren't enough; they must be coupled with progression in our understanding, hypotheses, and algorithms if its full potential impact is to be realized. In this thesis, we present three major contributions furthering our understanding of the genome through computational and algorithmic developments

Dissertation Outline and Contributions

The thesis is outlined as follows:

In chapter 2, we present our contributions to the analysis of ChIP-seq data and the resulting improvement in genomic annotation in two ChIP datasets. The contributions include an improved peak caller, a probabilistic model of the ChIP enrichment process which allows ambiguously aligning reads to be iteratively realigned to the genome and enabling peak calling in previously-censored repeat regions of the genome. These contributions are released as an open-source software package called AREM, or “Aligning Reads by Expectation-Maximization”, which is available at `https://github.com/uci-cbcl/AREM`. Portions of this chapter were published as part of [83].

In chapter 3, we present contributions centered on automatically determining epigenomic anotations and contrasting them across multiple species. We expand the HMM methodology of Ernst et al. [29] (called ChromHMM) for automatic epigenetic segmentation and annotation, but instead of using independent models for each cell type, we generalize the model by connecting the hidden nodes in a lineage. Our model comes closer to the biological reality that different cell types share common lineage from early stem-like progenitors and could be adapted for time series that include both lineage and spacial information. Since exact inference in the proposed model is computationally expensive, we explore several approximate variational inference methods including mean field, structured mean field and loopy belief propagation, and explore the accuracy of each method using synthetic and real data. We show that our model exhibits several desirable features including improved accuracy of inferring chromatin states, improved handling of missing data, and linear scaling with dataset size. Finally, we cluster the epigenetic marks in multiple species and segment the genome into 18 distinct chromatin state types, showing improved accuracy of the inferred state over the previous ChromHMM. These contributions are released as an open-source software package called TreeHMM, available at `https://github.com/uci-cbcl/tree-hmm`. Portions of this chapter were published as part of [10].

In chapter 4, we introduce a de bruijn graph assembly method which combines several algorithms from popular assembly programs and includes a new approach to expanding contigs. The implementation is built on Hyracks and Pregelix, scales well with respect to the genome size, and shows an N_{50} value similar to previously reported findings. Further, the framework is inherently adaptable, able to be run on large clusters of commodity machines (such as Amazon's EC2 cloud), in more traditional grid environments, or even on a single machine with limited memory and computational resources. These contributions are released as an open-source software package called Genomix, available at `https://github.com/uci-cbcl/genomix`.

Chapter 2

AREM: Aligning Reads by Expectation-Maximization

2.1 Introduction

In recent years, high-throughput sequencing coupled to chromatin immunoprecipitation (ChIP-seq) has become one of the premier methods of analyzing protein-DNA interactions [88]. The ability to capture a vast array of protein binding locations genome-wide in a single experiment has led to important insights in a number of biological processes, including transcriptional regulation, epigenetic modification and signal transduction [13, 72, 85, 103].

Chromatin immunoprecipitation followed by high-throughput sequencing yields a snapshot of a single protein's DNA interactions. The procedure consists of a chemical cross-linking that captures all protein-DNA interactions within the cell followed by a random DNA shearing, a targeted pull-down of the protein of interest, and finally, a reversal of the cross-link. DNA adjacent to the interaction site is pulled down and eventually amplified and sequenced. When aligned back to the reference genome, DNA fragments will be enriched in the area directly surrounding protein-DNA interaction sites of the protein of interest, allowing a researcher to identify binding sites based on the enrichment of DNA fragments. The enriched regions are generally termed “peaks” and a whole industry of “peak-callers” has been developed by the computational biology community; this is an area that continues to receive major efforts.

Some approaches to calling peaks are outlined in section 2.1.3. Typically, signal from the enriched DNA fragments is compared against a control experiment to account for fluctuations due to the pull-down process, chromatin accessibility, GC content, or copy number variations in the sample and the most-enriched regions are assumed to contain the original protein-DNA interaction site. Many transcription factors have well-characterized preferences for particular DNA letters (“motif” sequences). Validation of peak-calling quality often consists of searching the identified peak regions for known motifs or submitting their sequences to a *de novo* motif search using MEME [3] or some other program.

Numerous methods have been developed to analyze ChIP-Seq data and they typically work well for identifying protein-DNA interactions located within non-repeat sequences. However, most genomes are riddled by repetitive DNA sequences. These sequences may show up dozens or even hundreds of times within a single genome and can be problematic in bioinformatic analysis. Normally, DNA fragments that align to the genome multiple times are either removed from the analysis completely (leaving holes in the analysis and our understanding of the genome) or one of several possible alignments is chosen at random. We present novel methodology for identifying protein-DNA interactions in these repeat sequences.

2.1.1 Contributions

In this chapter, we present our contributions to the analysis of ChIP-seq data and the resulting improvement in genomic annotation in two ChIP datasets.

- We implement an improved peak caller based on the method of Model-based Analysis of ChIP-Seq (MACS) [137]. In addition to the methods from MACS, we refine potential peaks by direct optimization of the local poisson enrichment p-value, identifying additional peaks at a higher resolution than previous methods.

- Second, we develop a probabilistic framework for ChIP-derived DNA read alignment that accounts for multiple possible alignments for each read. We show how to find a locally optimal maximum-likelihood (ML) solution in this framework using expectation-maximization.

- Third, we implement an efficient peak caller that solves both of the above problems jointly and demonstrate its use on two ChIP datasets.

These contributions are released as an open-source software package called AREM, or "Aligning Reads by Expectation-Maximization", which is available at `https://github.com/uci-cbcl/AREM`. Portions of this chapter were published as part of [83].

2.1.2 Chapter Outline

The remainder of this chapter is structured as follows: Section 2.1.3 surveys related work, including MACS, the basis for our improved peak caller. The peak caller is presented in 2.2. Section 2.3 presents the probabilistic framework for read realignment as well as our solution to the local maximum likelihood problem using expectation-maximization. Section 2.4 presents the results of combining these two methods in two ChIP datasets, and finally, section 2.5 includes a brief discussion of the methods and possible future work.

2.1.3 Related Work

For the alignment phase, Eland, MAQ, Bowtie, and SOAP are among the most popular for mapping short reads to a reference genome [22, 58, 60, 61] and provide many or all of the potential alignments for a given sequence read. Once potential mappings have been identified, significantly enriched genomic regions are identified using one of several available tools [12, 30, 47, 76, 93, 99, 109, 133, 137]. Some peak finders are better suited for histone modification studies, others for transcription factor binding site identification. These peak finders have been surveyed on several occasions [52, 90, 126].

2.2 Optimizing Peak Caller

MACS searches the pileup of reads and leverages the fact that reads overlapping the ChIP binding site display a distinct bimodal binding profile, with forward strand reads appearing only upstream of the binding site, while negative strand reads only appear downstream of the binding site. After determining an empirically-optimal shift pushing the two strands together into a taller, unimodal distribution, MACS compares the number of reads within each peak region against a locally-defined poisson background. The background rate is the highest of the region directly surrounding the peak, the several-kilobase region around that, and the background region for the entire genome. Statistically speaking, this method will lead to more conservative predictions than using only one of the background rates. The peak boundaries themselves are defined heuristically as the region where the read count goes above some predefined threshold; all sites going above that threshold are considered candidate peaks, though only those surpassing a predefined poisson p-value are reported as peaks in the end.

In our methodology, after performing the bimodal to unimodal read shift and identifying candidate peaks, we apply a further filter to refine potential peaks. Specifically, we attempt to prune the edges of the peak regions where doing so decreases the poisson p-value by excluding fringe reads from the peak region.

2.2.1 Alignment

We aligned the data using Bowtie [58] with the Burrows-Wheeler index provided by the Bowtie website. The index is based on the unmasked MM9 reference genome from the UCSC Genome Browser [95]. We clipped the first base of all raw reads to remove sequencing artifacts and allowed a maximum of two mismatches in the first 28 bases of the remaining sequence. We generated several alignment collections for both Srebp-1 and Rad21 by varying k, the maximum number of reported alignments. We restricted our study to search the 1, 10, 20, 40, and 80 best alignments. Table 1 shows that the total number of alignments was only starting to plateau at k=80, indicating that many sequences have more than 80 possible alignments, for practicality we restricted our search as above. We calculated map confidence scores from Bowtie output as in [60]. We also provide an option for using the aligner's confidence scores directly rather than recalculating them from mismatches and sequence qualities. During preparation of the sequencing library, unequal amplification can result in biased counts for reads. To eliminate this bias, we limit the number of alignments to one for each start position on each strand. In particular, we choose the best alignment (based on quality score) for each position; in the event that all alignments have the same quality score, we choose a random read to represent that particular position.

2.2.2 Peak Finding

Our peak finding method is an adapted version of the MACS [137] peak finder. Like MACS, we empirically model the spatial separation between +/- strand tags and shift both treatment and control tags. We also continue MACS' conservative approach to background modeling, using the highest of three rates as the background (in this study, genome-wide or within 1,000 or 10,000 bases). As a divergence from MACS, we use a sliding window approach to identify large potentially enriched regions then use a smoothened greedy approach to refine called peaks. We call peaks within this large region by greedily adding reads to improve enrichment, but avoid local optima by always looking up to the full sliding window width away. The initial large regions correspond to the K regions used for the E-M steps of Section 2.3.5. During the E-M steps, local background rates are used as during final peak-calling. Peaks reported in this study are above a p-value of 10^{-5}. All enrichment scores and p-values are calculated using the poisson linear interpolation described in equation 2.12. Once E-M is complete on the treatment data and peaks are called, we reset the treatment alignment probabilities, swap treatment and control and rerun the algorithm, including E-M steps, to determine the False Discovery Rate (FDR). For all algorithms tested in this study, we define the FDR as the ratio of peaks called using control data to peaks called using treatment data. This method of FDR calculation is common in ChIP-Seq studies (e.g., [133, 137]).

2.2.3 Motif finding

Motif presence helps determine peak quality, as shown in [14]. To determine if our new peaks were of the same quality as the other peaks, we performed *de novo* motif discovery using MEME [3] version 4.4. Input sequence was limited to 150 bp (Rad21) and 200 bp (Srebp-1) around the summit of the peaks called by MACS from uniquely mapping reads. All sequences were used for Srebp-1, while 1,000 sequences were randomly sampled a total of 5 times for Rad21. The motif signal was strong in both datasets and we extracted the discovered motif position weight matrix (PWM) for further use. We also performed the motif search using Srebp-1 and CTCF motifs catalogued in Transfac 11.3, and found similar results. For the CTCF motif, we did genomic sampling (100,000 samples) to identify a threshold score corresponding to a z-score of 4.29. For Srebp-1, we used the threshold score reported by MEME (see **Figure 1**).

2.3 A Probabilistic Framework for ChIP Read Realignment

Many short reads cannot be uniquely mapped to the reference genome. Most peak finding workflows throw away these non-uniquely mapped reads, and as a consequence have low power for detecting peaks located within repeat regions. While each experiment varies, only about 60% [in house data] of the sequence reads from a ChIP-Seq experiment can be uniquely mapped to a reference genome. Therefore, a significant portion of the raw data is not utilized by the current methods. There have been proposals to address the non-uniquely mapped reads in the literature by either randomly choosing a location from a set of potential ones [50, 101] or by taking all potential alignments [76], but most peak callers are not equipped to deal with ambiguous reads.

We propose a novel peak caller designed to handle ambiguous reads directly by performing read alignment and peak-calling jointly rather than in two separate steps. In the context of ChIP-Seq studies, regions enriched during immunoprecipitation are more likely the true genomic source of sequence reads than other regions of the genome. We leverage this idea to iteratively identify the true genomic source of ambiguous reads. Under our model, the true locations of reads and binding peaks are treated as hidden variables, and we implement an algorithm, AREM, to estimate both iteratively by alternating between mapping reads and finding peaks.

2.3.1 Notations

Let $R = \{r_1, \cdots, r_N\}$ denote a set of reads from a ChIP-Seq experiment with read $r_i \in \Sigma^l$, where $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, l is the length of each read, and N denotes the number of reads. Let $S \in \Sigma^L$ denote the reference sequence to which the reads will be mapped. In real applications, the reference sequence usually consists of multiple chromosomes. For notational simplicity, we assume the chromosomes have been concatenated to form one reference sequence.

We assume that for each read we are provided with a set of potential alignments to the reference sequence. Denote the set of potential alignments of read r_i to S by $A_i = \{(l_{ij}, q_{ij}) : j = 1, \cdots, n_i\}$, where l_{ij} and q_{ij} denote the starting location and the confidence score of the j-th alignment, and n_i is the total number of potential alignments. We assume $q_{ij} \in [0, 1]$ for all j, and use it to account for both sequencing quality scores and mismatches between the read and the reference sequence. There are several programs available to generate the initial potential alignments and confidence scores.

2.3.2 Mixture model

We use a generative model to describe the likelihood of observing the given set of short reads from a ChIP-Seq experiment. Suppose the ChIP procedure results in the enrichment of K non-overlapping regions in the reference sequence S. Denote the K enriched regions (also called peak regions) by $\{(s_k, w_k) : k = 1, \cdots, K\}$, where s_k and w_k represent the start and the width, respectively, of the i-th enriched region in S. Let $E_k = \{s_k, \cdots, s_k + w_k - l\}$ denote the set of locations in the enriched region k that can potentially generate a read of length l. Let E_k^s, E_k^w denote the start and width of region k. We will use E_0 to denote all locations in S that are not covered by $\bigcup_{k=1}^{K} E_k$.

We use variable $z_i \in \{1, \cdots, n_i\}$ to denote the true location of read r_i, with $z_i = j$ representing that r_i originates from location l_{ij} of S. In addition, we use variable $u_i \in \{0, 1, \cdots, K\}$ to label the type of region that read r_i belongs to. $u_i = k$ represents that read r_i is from the non-enriched regions of S if $k = 0$, and is from k-th enriched region otherwise. Both z_i and u_i are not directly observable, and are often referred to as the hidden variables of the generative model.

Let $P(r_i|z_i = j, u_i = k)$ denote the conditional probability of observing read r_i given that r_i is from location l_{ij} and belongs to region k. Assuming different reads are generated independently, the log likelihood of observing R given the mixture model is then

$$\ell = \sum_{i=1}^{N} \log \left[\sum_{j=0}^{n_i} \sum_{k=0}^{K} P(r_i|z_i = j, u_i = k) P(z_i = j) P(u_i = k) \right],$$

where $P(z_i)$ and $P(u_i)$ represent the prior probabilities of the location and the region type, respectively, of read r_i. $P(z_i)$ is set according to the confidence scores of different alignments

$$P(z_i = j) = \frac{q_{ij}}{\sum_{k=1}^{n_i} q_{ik}}. \tag{2.1}$$

$P(u_i)$ depends on both the width and the enrichment ratio of each enriched region. Denote the enrichment ratio of the ChIP regions vs non-ChIP regions by α, which is often significantly impacted by the quality of antibodies used in ChIP experiments. We parametrize the prior distribution on region types as follows

$$P(u_i = k) = \frac{1}{(\alpha - 1)\sum_j w_j + L} \times \begin{cases} L - \sum_j w_j & \text{if } k = 0 \\ \alpha w_k & o.w. \end{cases} \tag{2.2}$$

2.3.3 Parameter estimation

The conditional probability $P(r_i|z_i = j, u_i = k)$ can be modeled in a number of different ways. For example, bell-shaped distributions are commonly used to model the enriched regions. However, for computational simplicity, we will use a simple uniform distribution to model the enriched regions. If read r_i comes from one of the enriched regions, i.e., $k \neq 0$, we assume the read is equally likely to originate from any of the potential positions within the enriched region, that is,

$$P(r_i|z_i = j, u_i = k) = \frac{1}{w_k - l + 1}\mathbf{I}_{E_k}(l_{ij}), \tag{2.3}$$

where $\mathbf{I}_A(x)$ is the indicator function, returning 1 if $x \in A$ and 0 otherwise.

If the read is from non-enriched regions, i.e., $k = 0$, we use p_i^b to model the background probability of an arbitrary read originating from location i of the reference sequence. (We assume p_i^b has been properly normalized such that $\sum_{i=1}^{L} p_i^b = 1$.) Then the conditional probability $P(r_i|z_i = j, u_i = k)$ for the case of $k = 0$ is modeled by

$$P(r_i|z_i = j, u_i = 0) = \mathbf{I}_{E_0}(l_{ij})\ p_{l_{ij}}^b. \tag{2.4}$$

Numerous ChIP-Seq studies have demonstrated that the locations of ChIP-Seq reads are typically non-uniform, significantly biased toward promoter or open chromatin regions [88]. The p_i^b's takes this ChIP and sequencing bias into account, and can be inferred from control experiments typically employed in ChIP-Seq studies.

Next we integrate out the u_i variable to obtain the conditional probability of observing r_i

given only z_i

$$P(r_i|z_i = j) = P(u_i = 0)\mathbf{I}_{E_0}(l_{ij})\ p^b_{l_{ij}} + \sum_{k=1}^{K} \frac{P(u_i = k)}{w_k - l + 1}\mathbf{I}_{E_k}(l_{ij}). \tag{2.5}$$

Note that because $E_0, E_1, \cdots, E_K$ are disjoint, only one term in the above summation can be non-zero. This property significantly reduces the computation for parameter estimation since we do not need to infer the values of u_i variables any more.

The log likelihood of observing R given the mixture model can now be written as

$$\ell(r_1, \cdots, r_n; \Theta) = \sum_{i=1}^{N} \log \left[\sum_{j=0}^{n_i} P(r_i|z_i = j)P(z_i = j) \right], \tag{2.6}$$

where $\Theta = (s_1, w_1, \cdots, s_K, w_K, \alpha)$ denotes the parameters of the mixture model. We estimate the values of these unknown parameters using maximum likelihood estimation

$$\hat{\Theta} = \arg\max_{\Theta}\ \ell(r_1, \cdots, r_n; \Theta). \tag{2.7}$$

2.3.4 Expectation-maximization algorithm

We solve the maximum likelihood estimation problem in Eq. (2.7) through an expectation-maximization (E-M) algorithm. The algorithm iteratively applies the following two steps until convergence:

Expectation step: Estimate the posterior probability of alignments under the current estimate of parameters $\Theta^{(t)}$:

$$Q^{(t)}(z_i = j|R) = \frac{1}{C} P(r_i|z_i = j, \Theta^{(t)})P(z_i = j), \tag{2.8}$$

where C is a normalization constant.

Maximization step: Find the parameters $\Theta^{(t+1)}$ that maximize the following quantity,

$$\Theta^{(t+1)} = \arg\max_{\Theta} \sum_{i=1}^{N} \sum_{j=0}^{n_i} Q^{(t)}(z_i = j|R) \log P(r_i|z_i = j, \Theta). \tag{2.9}$$

2.3.5 Implementation of E-M updates

The mixture model described above contains $2K + 1$ parameters. Since K, the number of peak regions, is typically large, ranging from hundreds to hundreds of thousands, exactly solving Eq. (2.9) in the maximization step is nontrivial. Instead of seeking an exact solution, we identify the K regions from the data by considering all regions where the number of possible alignments is significantly enriched above the background.

For a given window of size w starting at s of the reference genome, we first calculate the number of reads located within the window, weighted by the current estimation of posterior alignment probabilities,

$$f(s, w) = \sum_{i=1}^{N} \sum_{j=1}^{n_i} Q^{(t)}(z_i = j|R)\ \mathbf{I}_{[s,s+w-l]}(l_{ij}). \tag{2.10}$$

We term this quantity the foreground read density. As a comparison, we also calculate a background read density $b(s, w)$, which is estimated using either reads from the control experiment or reads from a much larger extended region covering the window. Different ways of calculating background read density are discussed in [137].

Provided with both background and foreground read densities, we then define an enrichment score $\phi(s, w)$ to measure the significance of read enrichment within the window starting at position s with width w. For this purpose, we assume the number of reads are distributed according to a Poisson model with mean rate $b(s, w)$. If $f(s, w)$ is an integer, the enrichment score is defined to be $\phi(s, w) = -\log_{10}(1 - g(f, b))$, where

$$g(x,\ \lambda) = e^{-\lambda} \sum_{k=0}^{x} \frac{\lambda^k}{k!} \tag{2.11}$$

denotes the chance of observing at least x Poisson events given the mean rate of λ. However, if $f(s, w)$ is not an integer, the enrichment score cannot be defined this way. Instead, we use a linear extrapolation to define the enrichment score $\phi(s, w) = -\log_{10}(1 - \tilde{g}(f, b))$, where function $\tilde{g}$ is defined as

$$\tilde{g}(x,\ \lambda) = g(\lfloor x \rfloor,\ \lambda) + [g(\lceil x \rceil,\ \lambda) - g(\lfloor x \rfloor,\ \lambda)]\,(x - \lfloor x \rfloor). \tag{2.12}$$

If two potential alignments of a read have the same confidence score and are located in two peak regions with equal enrichment, the update of posterior alignment probabilities in Eq. (2.8) will assign equal weight to these two alignments. This is so because we have assumed that peak regions have the same enrichment ratio as described in Eq. (2.2), which is not true as some peak regions are more enriched than others in real ChIP experiments. To address this issue, we have also implemented an update of the posterior probabilities that takes the calculated enrichment scores into account as

$$Q^t(z_i = j|R) \leftarrow \sum_{k=1}^{K} [\ \phi(E_k^s,\ E_k^w)\ P(z_i = j)\ \mathbf{I}_{E_k}(z_i)] \tag{2.13}$$

which is then normalized. In practice, we found this implementation usually behaves better than the one without using enrichment scores.

We use entropy to quantify the uncertainty of alignments associated with each read. For read i, the entropy at iteration t is defined to be

$$H_i^t = -\sum_{j=1}^{n_i} Q^t(z_i = j|R) \log Q^t(z_i = j|R). \tag{2.14}$$

We stop the E-M iteration when the relative square difference between two consecutive entropies is small, that is, when

$$\frac{\sum_{i=0}^{N}(H_i^t - H_i^{t-1})^2}{\sum_{i=0}^{N}(H_i^{t-1})^2} < \epsilon, \tag{2.15}$$

where $\epsilon = 10^{-5}$ for results reported in this paper.

AREM seeks to identify the true genomic source of multiply-aligning reads (also called multireads). Many of the multireads will map to repeat regions of the genome, and we expect repeats to be included in the K potentially enriched regions. To prevent repeat regions from garnering multiread mass without sufficient evidence of their enrichment, we impose a minimum enrichment score. Effectively, unique or less ambiguous multireads need to raise enrichment above noise levels for repeat regions to be called as peaks. The minimum enrichment score is a parameter of our model and its effect on called peaks is explored in Results.

2.4 Results

Two ChIP-Seq datasets were used in this study: 1) cohesin, a new dataset generated in house, and 2) Srebp-1, a previously published dataset [103]. We generated the cohesin dataset by performing ChIP-Seq using mouse embryonic fibroblasts and an antibody targeting Rad21 [134], a subunit of cohesin. Cohesin is an essential protein complex required for sister chromatid cohesion. In mammalian cells, cohesin binding sites are present in intergenic, promoter and 3' regions-especially in connection with CTCF binding sites [63, 98]. It was found that cohesin is recruited by CTCF to many of its binding sites, and plays a role in CTCF-dependent gene regulation [80, 124]. Cohesin has been shown to bind to repeat sequences in a disease-specific manner [134], making it a particularly interesting candidate

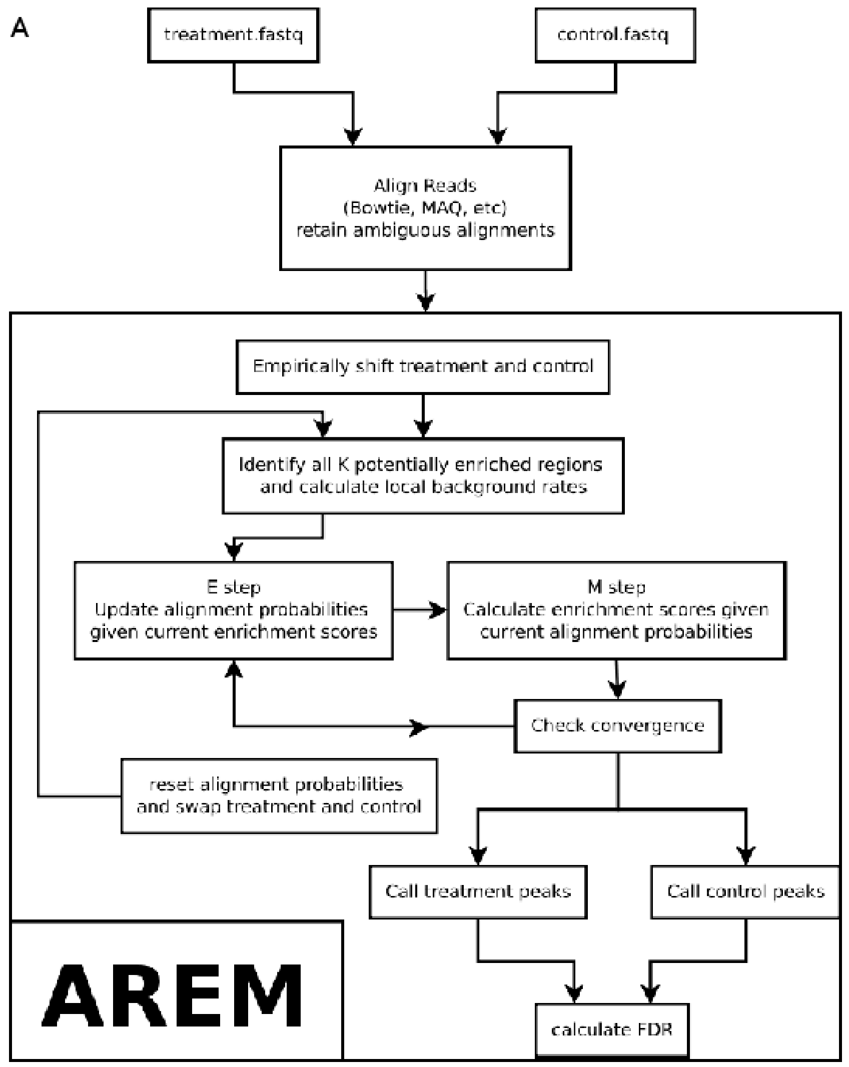

A
treatment.fastq
control.fastq
Align Reads
(Bowtie, MAQ, etc)
retain ambiguous alignments
Empirically shift treatment and control
Identify all K potentially enriched regions
and calculate local background rates
E step
Update alignment probabilities
given current enrichment scores
M step
Calculate enrichment scores given
current alignment probabilities
Check convergence
reset alignment probabilities
and swap treatment and control
Call treatment peaks
Call control peaks
calculate FDR
AREM

Figure 2.1: **AREM workflow diagram.** Given a set of multiply-mapping and singly-mapping reads, AREM shifts stranded reads towards each other, identifies all *potential* peak regions, then iterates between updating read alignment probabilities and peak enrichment scores until convergence. Once converged, AREM optimizes the enrichment score further, seeking the peak width that gives the optimal enrichment score. Finally, the process is repeated for the control data in order to determine the overall false discovery rate.

for our study.

The second dataset is Srebp-1, a transcription factor important in allostatic regulation of sterol biosynthesis and membrane lipid composition [39]. This particular dataset [103] examines the genomic binding locations for Srebp-1 in mouse liver. Regulation of expression by Srebp-1 is important for regulation of cholesterol; repeat-binding for this transcription factor has not been shown previously [39, 132]. We choose these datasets because both proteins have well characterized regulatory motifs, allowing us to directly test the validity of our peak finding method.

Building on the methodology of the popular peak-caller Model-based Analysis of ChIP-Seq (MACS) [137], we implement AREM, a novel peak caller designed to handle multiple possible alignments for each sequence read. AREM's peak caller combines an initial sliding window approach with a greedy refinement step and iteratively aligns ambiguous reads. We use two ChIP-Seq datasets in this study: Rad21 and Srebp-1. Rad21, a subunit of the structural protein cohesin, contained 7.2 million treatment reads and 7.4 million control reads (*manuscript in preparation*). Srebp-1, a regulator of cholesterol metabolism, had 7.7 million treatment reads and 6.4 million control reads [103].

Using AREM, we identify 19,935 Rad21 peaks covering more than 10 million base pairs at a low False Discovery Rate (FDR) of 3.7% and 1,474 Srebp-1 peaks covering nearly 1 million bases at a moderate FDR of 8%. For comparison, we also called peaks using MACS and SICER [133], another popular peak finding program. To compare our results, we use FDR and motif presence as indicators of *bona fide* binding sites.

2.4.1 AREM identifies additional binding sites

We seek to benchmark both AREM's peak-calling and its multiread methodology. To benchmark peak-calling, we limit all reads to their best alignment and run AREM, MACS and SICER. In the Rad21 dataset, AREM identifies 456 more peaks than MACS and 1920 more peaks than SICER but retains a similar motif presence (81.6% MACS, 82.5% SICER, 81.3% AREM) and has a lower FDR (2.8% MACS, 12.7% SICER, 1.9% AREM) (see **Table 1**). For Srebp-1, AREM identifies more than double the number of peaks compared to MACS and 816 more than SICER, though the FDR is slightly higher (4.85% MACS, 9% SICER, 8% AREM), and motif presence is slightly lower (46.6% MACS, 59% SICER, 39% AREM). In both datasets, AREM appears to be more sensitive to true binding sites, picking up more total sites with motif instances, although it trades off some specificity in Srebp-1.

To see if AREM can identify true sites that are not significant without multireads, we performed peak-calling with multireads, removing peaks that overlapped with those identified using AREM without multireads. Up to 1,546 (8.1%) and 272 (18.9%) previously unidentified peaks were called from Rad21 and Srebp-1, respectively. These new peaks have a similar motif presence compared to previous peaks but overlap with annotated repeat regions more often.

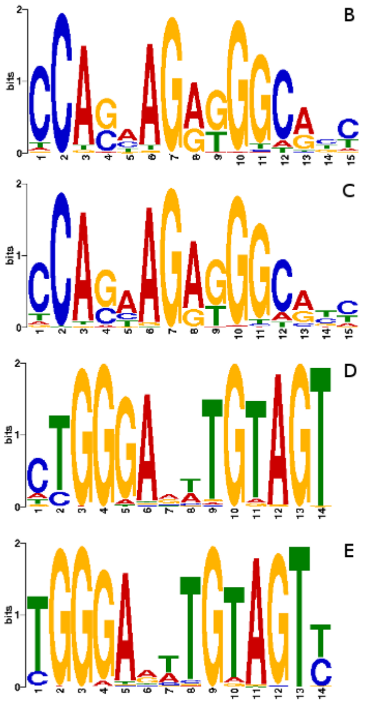


Figure 2.2: **B-E** *de novo* discovery of motifs. From top to bottom: **B)** CTCF in MACS peaks from uniquely mapping reads, **C)** CTCF in AREM's peaks with multireads, **C)** Srebp-1 in MACS peaks from uniquely mapping reads and **D)** Srebp-1 in AREM's peaks with multireads.

2.4.2 AREM's sensitivity is increased with ambiguous reads

Several methods for dealing with ambiguous reads have been proposed, including retaining all possible mappings, retaining one of the mappings chosen at random, and distributing weight equally among the mappings. The first option will clearly lead to false positives, particularly in repeat regions as the number of retained mappings increases. We compare the latter two methods to our E-M implementation, varying the number of retained reads and summarizing the results in **Table 1**. Although both random selection and fractionating reads increases the number of peaks called, our E-M method outperforms them, yielding 1546 more peaks for Rad21, and 272 for Srebp-1 with comparable quality. As the number of retained alignments increases, the disparity gets smaller. AREM shows fairly consistent results across datasets with a large increase in total number of alignments (nearly 40-fold for Rad21, over 10-fold for Srebp-1).

For a given sample, the iterations show a continued shift of the max alignment probabilities to either 1 or 0. This shift is consistent across datasets with larger numbers of max alignments (data not shown), but does depend on other parameters. What is apparent is that AREM's E-M heuristic performs well, allowing for significant shift toward a "definitive" alignment; at the same time, it does not force a shift on reads with too little information, preventing misalignment and resulting spurious peak-calling.

2.4.3 AREM is sensitive to repeat regions

An important parameter in our model is the minimum enrichment score for all K regions. Since repeat regions have such similar sequence content, many reads will share the same repetitive elements. If one of the shared repeat elements has a slightly higher enrichment score by chance, the E-M method will iteratively shift probability into that repeat region, snowballing the region into what appears to be a full-fledged sequence peak. To distinguish repetitive peaks arising by small enrichment fluctuations from true binding sites within or adjacent to repetitive elements, we impose a minimum enrichment score on all regions. Using lower threshold scores, our method may include false positives from these random fluctuations. However, true binding peaks near repetitive elements may be missed if the score is too high.

To explore the effect of varying the minimum enrichment score, we varied the minimum score from 0.1 to 2, keeping the maximum number of alignments fixed at 20. For Rad21, we see a declining number of discovered peaks ranging from 28,305 to 19,634 peaks. In addition to a decline in discovered peaks as minimum enrichment score increases, we also see a decrease in the reported FDR and the percent of peaks in repeat regions from 11.28% to 2.95% FDR and 71.56% to 59.02%. Lastly, the percent of peaks with motif increases from 63.64% to 81.12%. These additional peaks appear to be of lower quality: motifs are largely absent from them and the FDR is much higher (see **Figure 2**).

For our method, detecting peaks near repeat regions is a tradeoff between sensitivity and specificity. As the minimum score increases, the method approaches the uniform or "fraction" distribution, in which only the initial mapping quality scores (and not the enrichment) affect alignment probabilities. The fraction method is explored explicitly, showing increased power compared to unique reads only, but decreased sensitivity to true binding sites compared to other AREM runs.

On a 2.8Ghz CPU, AREM takes about 20 minutes and 1.6GB RAM to call peaks from over 12 million alignments and about 30 minutes and 6GB RAM to call peaks from nearly 120 million alignments. Each dataset takes less than 40 iterations to converge. AREM is written in Python, is open-source, and is available at http://sourceforge.net/projects/arem.

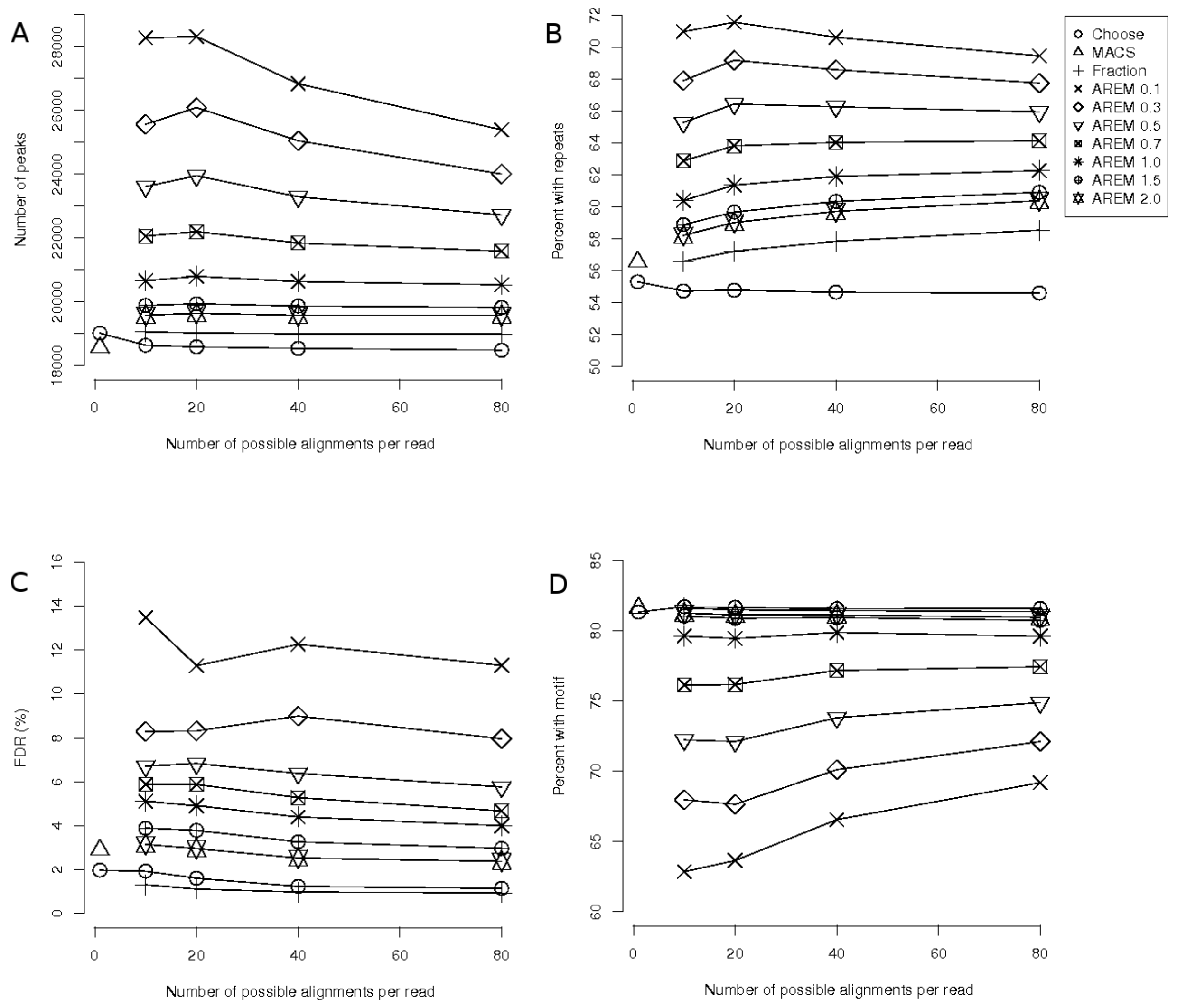


Figure 2.3: Graphs displaying varying parameters and number of possible alignments per read. **A)** Total number of peaks discovered. **B)** Percentage of peaks with repetitive sequences. **C)** False Discovery Rate. **D)** Percentage of peaks with motif.

2.5 Discussion

Repetitive elements in the genome have traditionally been problematic in sequence analysis. Since sequenced reads are short and repetitive sequences are similar, many equally likely mappings may exist for a given read. Our method uses the low-coverage unique reads near repeat regions to evaluate which potential alignments for each read are the most likely. Our method's sensitivity to repeat regions is adjustable, but increasing sensitivity may introduce false positives. Further refinement of our methodology may lead to increased specificity.

Method	# Alignments	# Peaks	Peak Bases	FDR	New Peaks	Motif	Repeat
Cohesin							
MACS	2,368,229	18,556	9,546,641	2.8%	—	81.67%	56.55%
SICER	2,368,229	17,092	17,374,108	12.71%	—	82.55%	70.42%
AREM 1	2,368,229	19,012	9,353,567	1.9%	—	81.32%	55.30%
AREM 10	7,616,647	19,881	10,225,479	3.8%	1,404	81.04%	58.88%
AREM 20	12,312,878	19,935	10,531,465	3.7%	1,517	80.88%	59.66%
AREM 40	20,527,010	19,863	10,744,836	3.2%	1,546	80.93%	60.34%
AREM 80	34,537,311	19,820	10,972,796	2.9%	1,538	80.73%	60.91%
Srebp-1							
MACS	10,482,005	721	495,968	4.85%	—	46.60%	53.95%
SICER	10,482,005	622	963,778	9.0%	—	59.00%	77.33%
AREM 1	10,482,005	1,438	880,284	8.0%	—	39.08%	53.47%
AREM 10	28,347,869	1,815	996,346	10.5%	262	39.22%	56.04%
AREM 20	44,493,532	1,748	959,646	8.0%	227	39.95%	55.97%
AREM 40	72,453,642	1,685	983,459	8.2%	248	40.34%	56.46%
AREM 80	118,744,757	1,695	987,746	7.3%	272	40.66%	56.73%

Table 2.1: **Comparison of peak-calling methods for cohesin and Srebp-1.** Three peak callers (MACS, SICER, and AREM) were run on both datasets. For AREM, the maximum number of retained alignments per read is varied (from 1 to 80). The total number of peaks and bases covered by peaks is reported as well as the FDR by swapping treatment and control. For both datasets, AREM's minimum enrichment score was fixed at 1.5 with 20 maximum alignments per read. For comparison, the motif background rate of occurence was 4.5% (CTCF) and 27% (Srebp-1) in 100,000 genomic samples, sized similarly to Rad21 MACS peaks and Srebp-1 MACS peaks, respectively.

Our results imply that functional CTCF binding sites exist within repeat regions, revealing an interesting relationship between repetitive sequence and chromatin structure. Another application of our method would be to explore the relationship between repetitive sequence and epigenetic modifications such as histone modifications. Regulation of and by transposable elements has been linked to methylation marks [44], and transposable elements have a major role in cancers [21]. Better identification of histone modifications in regions of repetitive DNA increases our understanding of key regulators of genome stability and diseases sparked by translocations and mutations.

Chapter 3

Discovering and Mapping Chromatin States Using a Tree Hidden Markov Model

3.1 Introduction

In this chapter, we shift our focus to the annotation of differences between cell types. New biological techniques and technological advances in high-throughput sequencing are paving the way for systematic, comprehensive annotation of many genomes, allowing differences between cell types or between disease/normal tissues to be determined with unprecedented breadth. Epigenetic modifications have been shown to exhibit rich diversity between cell types, correlate tightly with cell-type specific gene expression, and changes in epigenetic modifications have been implicated in several diseases. Previous attempts to understand chromatin state have focused on identifying combinations of epigenetic modification, but in cases of multiple cell types, have not considered the lineage of the cells in question.

Although identical DNA is shared amongst most cells in an organism, a key question in biology relates to how different cell types are formed, maintained, and made to perform vastly different functions. Recent studies have shown that these processes are in part mediated by the post-translational modifications of histone tails, which in turn affect chromatin accessibility and other properties of chromatin structures in a cell-type specific way [6]. There are also interactions between these modifications [102, 110], which act combinatorially to exert dynamic control over gene expression and other fundamental cellular processes[31]. Although we do not fully understand the role of epigenetic modifications, their effect in the development of disease and in defining cell type is becoming clearer. For example, epigenetic changes have been shown to be tightly correlated with gene expression [26, 53, 129], have been linked to metastasis development in certain types of cancer [48] and are shown to control recombination [7]. Epigenetic inheritance across cells and across individuals has been highlighted in recent research (see [45] for a review) and our understanding of the scope of epigenetic modifications has expanded considerably in recent years.

There is enough DNA in most human cells to stretch up to 1.8 meters if completely unwound and laid end-to-end. Normally, the long threads of DNA are ultra-compacted down to only about 90 micrometers by being wrapped around histone proteins [94]. The histones act like miniature spools for the DNA threads, and can be further condensed by forming chemical interactions with other histones. These interactions occur through modifications to the tails of the histones. Both DNA compaction and expansion need to be efficient since different portions of the DNA must be made available at different times and in different cell types for replication and transcription. Since these chemical modifications can directly affect levels of transcription and overall cell state but aren't modifications to the letter sequence of the DNA, they fit under the broad category of epigenetics.

As outlined in section 2.1, chromatin immunoprecipitation coupled with high-throughput sequencing (ChIP-seq) has emerged as a cost-effective method for determining epigenetic modifications. Although initially used as a high-resolution transcription factor binding site discovery mechanism (see [88, 91] for review), ChIP-seq has recently been used to target these histone tail modifications and is proving to be particularly cost-effective method for epigenomic annotation. Thanks to the ENCODE project [53], hundreds of ChIP-seq datasets are now publicly available and the process of integrating species-specific and cell-type specific binding site information, gene expression, and chromatin state is now underway. These high-throughput datasets provide an unbiased, comprehensive view of the function of different genomic regions and enable comparison between multiple cell types and different species.

3.1.1 Contributions

In this chapter, we present contributions centered on automatically determining epigenomic anotations and contrasting them across multiple species.

- We expand the HMM methodology of Ernst et al. [29] (called ChromHMM) for automatic epigenetic segmentation and annotation. Instead of independent models for each cell type, we generalize the model by connecting the hidden nodes in a lineage. The resulting Bayesian network uses epigenetic modifications to simultaneously model 1) chromatin mark combinations that give rise to different chromatin states and 2) propensities for transitions between chromatin states through differentiation or disease progression.

- Since exact inference in the proposed model is computationally expensive, we explore several approximate variational inference methods including mean field, structured mean field and loopy belief propagation, and explore the accuracy of each method using synthetic and real data. We show that our model exhibits several desirable features including improved accuracy of inferring chromatin states, improved handling of missing data, and linear scaling with dataset size.

- We show that the structured mean field (SMF) approximation is closest to the exact inference solution. Using SMF, we perform genome-wide prediction on two genomes (mouse and human) using two large two Encyclopedia of DNA Elements (ENCODE) datasets [29]. The genome is clustered into 18 distinct chromatin states and we show improved accuracy of the inferred state over the previous ChromHMM.

These contributions are released as an open-source software package called TreeHMM, available at `https://github.com/uci-cbcl/tree-hmm`. Portions of this chapter were published as part of [10].

3.1.2 Chapter Outline

The remainder of this chapter is structured as follows: Section 3.1.3 surveys related work, including ChromHMM, which we compare favorably to in section 3.4.6. Our improved graphical model is presented in 3.2 as well as some implementation details surrounding data preprocessing in sections 3.2.4 - 3.2.5. Section 3.3 presents three variational approximations to exact inference in our graphical model and goes into detail about their derivations, normalization, inference, and learning steps.. Section 3.4 presents the results of our method, with the different approximations compared on artificial data and human data in sections 3.4.1 through 3.4.3. Section 3.4.5 presents our results decoding the the entire human and mouse genomes using the SMF approximation and section 3.4.6 presents our comparison with ChromHMM. Finally, section 3.5 includes a brief discussion of the methods and possible future work.

List of abbreviations

MF: Mean Field; SMF: Structured Mean Field; LBP: loopy belief propagation; BIC: Bayes Information Criterion

3.1.3 Related Work

Several computational approaches have been used to tackle the important problem of genome annotation using these high-throughput epigenetic datasets. In particular, methods that integrate histone modification data can be segregated into two general approaches: one approach searches near known genomic annotations to identify characteristic marks of particular classes of regions, such as promoters and enhancers, and subsequently uses the learned characteristics to find new instances of the class [40, 42, 73]. The other approach learns the characteristic patterns of histone marks *de novo* using unsupervised methods, "rediscovering" and predicting genomic features associated with mark combinations. Methods for identifying these patterns have included clustering [46, 116], a dynamic Bayesian network [41], and hidden Markov models (HMM) [28, 53, 128]. These methods differ mostly in how they model the chromatin mark signal intensity. Some determine a characteristic signal shape while others focus on modeling the mark signal using non-parametric histograms, multivariate normal distributions, or binary presence and mark co-occurrence. Each of these methods focuses on modeling the histone mark combinations; none explicitly incorporate the *lineage* information by which the data are related.

Here, we expand the HMM methodology of Ernst et al. [29] (called ChromHMM), who originally analyzed nine transcription factors (TF) or histone modifications (plus control) performed in nine different human cell types. Their multivariate HMM model concatenated several cell types to form a single chain with the goal of learning a global set of histone mark combinations and left as secondary all comparative analysis between cell types. We generalize the model to more closely reflect biological reality: chromatin remodeling occurs as cells progress through several stages of differentiation. We expect many genomic regions to be correlated across a lineage since cell types diverged from a common progenitor are likely to share the chromatin changes that took place in that progenitor. To capture this reality, we simultaneously model both the genomic localization of histone marks and the chromatin dynamics along a lineage by explicitly aligning each cell type and connecting their internal, hidden nodes vertically in a tree structure. Our model learns both histone modifications' association with chromatin state and state transitions between cell types, capturing epigenetic changes that occur through differentiation or disease progression. Our method effectively pools information across species, and we expect it to show improved accuracy of genome segmentation over the previous HMM approach which does not incorporate cell lineage information.

3.2 Tree Hidden Markov Model

3.2.1 Model Description and Notation

We propose a tree hidden Markov model (TreeHMM) to discover and map chromatin states using the observed chromatin modification data. We begin by introducing some notation. We denote the chromatin modification of type l at position t of cell type i as $x^i_{t,l}$, which can take binary values, i.e. $x^i_{t,l} \in \{0, 1\}$. Subsequently we denote all the histone marks at position (i, t) to be $\mathbf{x}^i_t = (x^i_{t,1}, \dots, x^i_{t,L})$, which is a vector of length L and $X = \{\mathbf{x}^i_t : i = 1, \dots, I; t = 1, \dots, T\}$ to be the collection of all observed data. We further introduce a hidden variable z^i_t to denote the underlying chromatin state at chromosomal position t of cell type i. We assume z^i_t's are discrete taking K possible values, i.e., $z^i_t \in \{1, \dots, K\}$ for all t and i. Let $Z = \{z^i_t : i = 1, \dots, I; t = 1, \dots, T\}$ denote the collection of all hidden chromatin state variables. We assume that these chromatin state variables are the key determinant of the observed chromatin modifications, and that $\mathbf{x}_t^i$'s are independent of each other conditioned on Z, i.e., $\mathbb{P}(X|Z) = \prod_{i=1}^{I} \prod_{t=1}^{T} \mathbb{P}(\mathbf{x}_t^i | z_t^i)$.

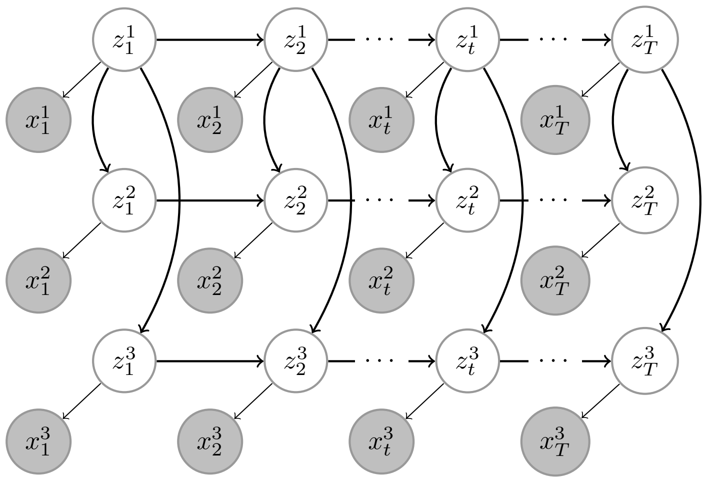


Figure 3.1: Example graphical model for a tree-structured HMM with three cell types. Hidden state variables representing chromatin states (white) are connected horizontally in a chain as well as vertically in a tree structure. Each chain in the graph represents a certain cell type. For example, the top chain represents the root cell type (e.g., ES cells). Observed nodes (grey) represent chromatin modifications and are connected only to the hidden variables.

We assume the I cell types are related to each other through a lineage tree $\mathcal{T}$ and use $\pi(i)$ to denote the parent node of the cell type i within the lineage tree $\mathcal{T}$. The conditional dependencies among the variables are modeled by a Bayesian network as shown in Figure 3.1 with the chromatin state variables at neighboring positions of each cell type linked as a chain (referred to as horizontal connections) and the state variables of different cell types at the same chromosomal position connected according to the lineage tree $\mathcal{T}$ (referred to as vertical connections). The horizontal connections capture the spatial correlation between chromatin states, i.e., the tendency of histone modifications to spread and cluster spatially across the genome, allowing for example large inactivated regions and short “poised” regions. The lineage relation is modeled by vertical connections between the same locations of different chains, and captures temporal changes in chromatin states during differentiation or disease progression over the cell lineage. Given the conditional dependency specification, the joint distribution of the chromatin state variables can then be written as

$$\mathbb{P}(Z) = \prod_{i=1}^{I} \prod_{t=1}^{T} \mathbb{P}(z_t^i | z_{t-1}^i, z_t^{\pi(i)}) \tag{3.1}$$

where by definition $z_{t-1}^i = \emptyset$ if $t = 1$ and $z_t^{\pi(i)} = \emptyset$ if node i is the root cell type. As a notation, we also use $\pi(i,t)$ to denote the parent nodes of node (i,t) in the model, and use $z_{\pi(i,t)}$ to denote the state variables at these parent nodes if they exist.

3.2.2 Parameters

The TreeHMM model presented above requires us to specify two sets of conditional distributions. One is the emission probabilities $P(\mathbf{x}_t^i | z_t^i)$, that is, the probability of observing chromatin modification vector $\mathbf{x}_t^i$ conditioned on chromatin state z_t^i. For simplicity, we assume different chromatin modification marks are independent of each other conditioned on the chromatin state, and use $e_l^k = \mathbb{P}(x_{t,l}^i = 1 | z_t^i = k)$ to denote the probability of observing mark l at position t of cell type i conditioned on the underlying state being k.

The second set of conditional probabilities we need to specify are the transition probabilities among chromatin states, that is, $\mathbb{P}(z_t^i | z_{t-1}^i, z_t^{\pi(i)})$. When $t > 1$ and $\pi(i)$ is not empty, we will use a $K \times K \times K$ matrix to specify $\mathbb{P}(z_t^i | z_{t-1}^i, z_t^{\pi(i)})$. However, when one of the conditioned variable is non-existent, we use $K \times K$ matrix to specify the transition probability. More specifically, the state transition probabilities are modeled as

$$\begin{aligned}
\theta_{bc}^a &= \mathbb{P}(z_t^i = a | z_t^{\pi(i)} = b, z_{t-1}^i = c) && t \neq 1,\ i \text{ is not root} \\
\alpha_b^a &= \mathbb{P}(z_t^i = a | z_{t-1}^i = b) && t \neq 1,\ i \text{ is root} \\
\beta_b^a &= \mathbb{P}(z_t^i = a | z_t^{\pi(i)} = b) && t = 1,\ i \text{ is not root} \\
\gamma^a &= \mathbb{P}(z_t^i = a) && t = 1,\ i \text{ is root.}
\end{aligned}$$

We will also use $\Theta = \{\theta^a_{bc}, \alpha^a_b, \beta^a_b, \gamma^a, e^a_l | (a,b,c) \in 1,\ldots,K; l \in 1,\ldots,L\}$ to denote the collection of all parameters associated with the model.

3.2.3 Model Description and Parametrization

We use a Bayesian network to model the chromatin states across the genome of I different cell types. The genomic location is divided into total of T fixed-size bins. Each bin is associated with a hidden variable (node) representing the underlying chromatin state and several observed nodes denote the measured chromatin markers. Nodes within one cell type are connected horizontally while cell lineage is modeled by connecting nodes at the same horizontal location in a lineage tree. The resulting model is a TreeHMM model as shown in Fig. 1 of the main article. Each node can be indexed by (i,t) with $i \in \{1,2,\ldots,I\}$ indicating the cell type and $t \in \{1,2,\ldots,T\}$ indicating the location. We denote the hidden variables as z^i_t and the observed variables as x^i_t. The possible values of the hidden and observed variables can take are

$$z^i_t \in \{1,\ldots,K\}, x^i_t \in \{0,1\}^L$$

where K is the number of chromatin states and L is the number of different epigenetic markers.

Additionally, we will define the following functions for notational simplicity:

$$\begin{aligned}
\mathrm{pa}(i,t) &= \{(j,s)|\text{ node }(j,s)\text{ is a parent of }(i,t)\} \\
&= \begin{cases} \emptyset & \text{if } i \text{ is the root and } t=1 \\ \{(i,t-1)\} & \text{if } i \text{ is the root and } t>1 \\ \{(\mathrm{pa}(i),t),(i,t-1)\} & \text{otherwise} \end{cases}
\end{aligned}$$

and

$$\mathrm{pa}(i) = \{j|j \text{ is the parent of cell type } i \text{ in the tree}\}$$

Below we define the parameters of our model, which specify the transition probabilities $\mathbb{P}(z_t^i|\mathrm{pa}(z_t^i))$ and emission probabilities $\mathbb{P}(x_t^i|z_t^i)$.

Transition Probabilities

The probability of observing state z_t^i conditional on the parent states $\mathrm{pa}(z_t^i)$ is given by

$$\mathbb{P}(z_t^i|\mathrm{pa}(z_t^i)) = \begin{cases} \theta_{mn}^k & \equiv \mathbb{P}(z_t^i = k|z_t^{\mathrm{pa}(i)} = m, z_{t-1}^i = n) & \text{if } t>1,\ i \text{ is not root} \\ \alpha_m^k & \equiv \mathbb{P}(z_t^i = k|z_{t-1}^i = m) & \text{if } t>1,\ i \text{ is root} \\ \beta_m^k & \equiv \mathbb{P}(z_t^i = k|z_t^{\mathrm{pa}(i)} = m) & \text{if } t=1,\ i \text{ is not root} \\ \gamma^k & \equiv \mathbb{P}(z_t^i = k) & \text{if } t=1,\ i \text{ is root} \end{cases}$$

The total number of parameters in each transition matrix are: $|\theta| = K^2 \times K$, $|\alpha| = K \times K$, $|\beta| = K \times K$, and $|\gamma| = K$.

Emission Probabilities

We treat each histone mark as independent variables, the probability of observing lth histone mark given certain cell state k is given by:

$$e_l^k \equiv \mathbb{P}(x_{t,l}^i = 1 | z_t^i = k) \qquad \forall i = 1 \dots I, t = 1 \dots T$$

where $x_{t,l}^i$ represents the lth histone mark, which takes a binary value of $\{0, 1\}$. Thus the emission probability matrix e has $K \times L$ parameters (or K vectors with dimension $1 \times L$).

We use variational EM algorithm to do model learning. The Variational EM algorithm minimizes the free energy

$$\begin{aligned} F &= -\sum_Z \mathbb{Q}(Z) \log \frac{\mathbb{P}(X, Z; \Theta)}{\mathbb{Q}(Z)} && (3.2) \\ &= \mathbb{E}_{\mathbb{Q}(Z)}[\log \mathbb{Q}(Z) - \log \mathbb{P}(X, Z; \Theta)] && (3.3) \end{aligned}$$

under some approximate form of hidden variable distribution. The algorithm iterates between two procedures - expectation (or inference) and maximization (or learning). Below we derive the update formula of the E-step and M-step for the mean-field and structured mean-field approximations.

3.2.4 Incorporating Missing Markers and Hidden Cell Types

Many additional histone modifications are available beyond the nine included in the ENCODE dataset. Most of the additional marks are only available for a small number of cell types. These markers, though excluded from the current analysis, could be incorporated to provide additional model refinement. Since each mark is treated as an independent variable, if a certain marker l is absent in some of the cell types, we simply remove the lth emission term in (3.12). While the inference step and parameter learning of the transition matrices follow the exact procedure, the estimation of emission parameters in the M-step is modified as

$$e_l^k \propto \sum_{i,t} q(z_t^i = k) I(x_{t,l}^i = 1) I(M_l^i = 1),$$

where I is the indicator function and M is a binary matrix with M_l^i indicating the availability of lth marker .

In the current approach we use simple, biologically-motivated tree structures for the human dataset. In reality, cells differentiate through multiple steps, and some of the intermediate cell types may not have available experimentally derived measurements. Also, other statistically motivated tree structures such as those created by hierarchical clustering could be used. Adding unobserved cell types has both biological and modeling significance. Computationally, hidden cell types can be treated as a special case of absent markers by considering all of its markers as missing. The inference step and parameter learning are similar to the above case of absent markers.

3.2.5 Data Preprocessing

As a preprocessing step, we create a histogram of mapped reads by dividing the genome into 200bp non-overlapping bins and counting the number of mapped reads whose middle base fell into each bin. All replicates, if any, were added to the histogram and the histogram was then binarized using a threshold corresponding to a Poisson p-value of 10^{-4}, similar to [29]. We further segmented the genome into regions with and without chromatin marks by applying a smoothing filter to the raw count data, retaining regions that contained mapped reads. Further data processing details can be found in Supplemental material section 3.4.2, and all preprocessing methods are available as part of the released source code.

Our model's preprocessing and parameterization are very similar to the multivariate HMM methodology of [29], however Ernst's implementation suffered from a very slow runtime on our processed data, which contains many regions to facilitate parallel inference. We re-implemented the method as described [28] and use this implementation for comparison in later sections. The implementation is available in the released source code.

3.3 Variational Inference

Given the TreeHMM model described above and the set of observed chromatin modification data X, our goal is to: 1) estimate the parameters of the model, and 2) infer the underlying hidden state at each chromosomal location of each cell type. For parameter learning, we will use the maximum likelihood method, that is, we seek to find the optimal parameter set Θ^* that maximizes the log likelihood function

$$\log \mathcal{L}(\Theta; X) = \log \mathbb{P}(X; \Theta) = \log \sum_{Z} \mathbb{P}(Z; \Theta)\mathbb{P}(X|Z; \Theta) \tag{3.4}$$

Note that in the above notation, we put Θ into the distributions to emphasize the dependency of the distributions on the parameters. However, we will also the simplified notation $\mathbb{P}(Z|X)$ or $\mathbb{P}(X)$ when the context is clear. After finding the optimal parameters, we infer the underlying chromatin states using posterior inference, to calculating the posterior probability of each chromatin state conditioned on the observed data, $\mathbb{P}(z_t^i|X; \Theta)$.

We explore various inference methods for the TreeHMM model, including exact methods and approximate methods. For exact inference, we provide two implementations: first, we generate a lattice for the Graphical Models Toolkit (GMTK) [11], which provides an efficient framework for exact inference and learning using the junction tree algorithm [25]. We also provide a custom library which implements a "cliqued" method in which each slice t of the model has all its nodes in that slice treated as if they were part of a single "cliqued" node that has K^I states. In this cliqued node representation, we can apply standard HMM methodology to do inference and learning. The state space of the cliqued inference method grows exponentially with I, but we found it to be faster than the GMTK implementation for small trees. Both implementations gave the same results in our testing.

Since the TreeHMM model contains undirected cycles, exact inference methods such as junction tree and the "cliqued" method quickly become intractable in computational time and memory consumption when the number of nodes I or the number of inferred states K increases. Therefore, we introduce several approximate inference methods to solve the inference and learning problem presented above. We focus on variational methods since they are usually computationally efficient and scale well with size of the dataset[119]. The overall strategy of variational methods is to find an easier-to-handle surrogate distribution of the states $\mathbb{Q}(Z)$ that can be used to approximate the true posterior distribution $\mathbb{P}(Z|X)$. This is done through the venue of the free energy function

$$F = -\sum_{Z} \mathbb{Q}(Z) \log \frac{\mathbb{P}(X, Z; \Theta)}{\mathbb{Q}(Z)} = \mathbb{E}_{\mathbb{Q}}[\log \mathbb{Q}(Z)] - \mathbb{E}_{\mathbb{Q}}[\log \mathbb{P}(X, Z; \Theta)] \tag{3.5}$$

By Jensen's inequality, F is always lower bounded by the negative log likelihood function, i.e. $F \geq -\log \mathcal{L}(\Theta; X)$, with equality holding if and only if $\mathbb{Q}(Z) = \mathbb{P}(Z|X)$. The goal of the variational inference is to find a $\mathbb{Q}$ distribution (usually under some approximate form) that minimizes the free energy function. We will consider three different forms of surrogate distributions and briefly describe variational inference for each of them. Details of the derivations are given in section 3.2.3.

Algorithm 1 Variational expectation-maximization algorithm

```
Initialize $\mathbb{Q}_0(Z) = \prod_i q_i(\mathbf{z_i})$, $\Theta_0 = (\theta_0, \alpha_0, \beta_0, \gamma_0, e_0)$
$s = 1$
repeat
  (E-step)
  repeat
    for $i = (1, \ldots, I)$ do
      $\log q_i(\mathbf{z_i}) \sim \mathbb{E}_{\{q_j\}|j \neq i} \log \mathbb{P}(X, Z; \Theta_{s-1})$
    end for
  until
    $\left| \frac{F(\mathbb{Q}_{new}(Z), \Theta_{s-1}) - F(\mathbb{Q}(\Theta_{s-1}, Z))}{F(\Theta_{s-1}, \mathbb{Q}(Z))} \right| < \epsilon$
  return $\mathbb{Q}_s(Z) = \prod_i q_i(\mathbf{z_i})$
  (M-step)
  Update $\Theta_s$=arg min$_\Theta$ $F(\Theta, \mathbb{Q}_s(Z))$ .
  Set $s = s + 1$.
until
   Convergence
```

3.3.1 Mean field (MF) variational inference

In the mean field variational method, we consider the surrogate distribution to be the product of the marginal distributions of each individual state variable

$$\mathbb{Q}(Z) = \prod_{i=1}^{I} \prod_{t=1}^{T} q(z_t^i) \tag{3.6}$$

where $q(z_t^i)$ represents the marginal distribution of z_t^i. For notational simplicity, we also use q_{it} as an abbreviation of $q(z_t^i)$. In this case, the free energy becomes

$$F = \sum_{i=1}^{I} \sum_{t=1}^{T} \mathbb{E}[\log q(z_t^i) - \log \mathbb{P}(x_t^i | z_t^i)] - \mathbb{E}[\log \mathbb{P}(z_t^i | z_{\pi(i,t)})] \tag{3.7}$$

where the expectation is with respect to $\mathbb{Q}$, as will always be the case in the remainder of this paper.

To find the optimal $\mathbb{Q}$ that minimizes the free energy, we use a coordinate descent method - alternatively updating each component q_{it} while keeping all other components fixed. To update q_{it} we collect the terms in F that involve q_{it},

$$F_{it} = \mathbb{E}[\log q(z_t^i) - \log \mathbb{P}(x_t^i|z_t^i)] - \mathbb{E}[\log \mathbb{P}(z_t^i|z_{\pi(i,t)})] - \sum_{\{(j,s):(i,t)\in\pi(j,s)\}} \mathbb{E}[\log \mathbb{P}(z_s^j|z_{\pi(j,s)})].$$

The last term involves nodes that are children of (i,t). The update formula for q_{it} is thus given by $q(z_t^i) \sim \exp\{\phi(z_t^i)\}$, up to a normalizing constant, where

$$\phi(z_t^i) = \log \mathbb{P}(x_t^i|z_t^i) + \mathbb{E}_{q_{\pi(i,t)}}[\log \mathbb{P}(z_t^i|z_{\pi(i,t)})] + \sum_{\{(j,s):(i,t)\in\pi(j,s)\}} \mathbb{E}_{q_{js},q_{\pi(j,s)\setminus(i,t)}}[\log \mathbb{P}(z_s^j|z_{\pi(j,s)})].$$

The (j,s) nodes in the last term are all children of node (i,t), but the expectation involves all the parents of (j,s) except (i,t).

3.3.2 Mean-field derivation and normalization

The mean-field approximation assumes the following factorized form of $\mathbb{Q}(Z)$

$$\mathbb{Q}(Z) = \prod_i \prod_t q(z_t^i) \tag{3.8}$$

where $q(z_t^i)$ represents the distribution of hidden variable z_t^i of node (i,t).

Under the assumption (3.8), the first term of (3.2) becomes

$$\begin{aligned}\mathbb{E}_{\mathbb{Q}(Z)}[\log \mathbb{Q}(Z)] &= \sum_i \sum_t \sum_Z \left(\prod_{i'} \prod_{t'} q(z_{t'}^{i'})\right) \log q(z_t^i) \\ &= \sum_i \sum_t \sum_{z_t^i} q_{it} \log q_{it}.\end{aligned} \tag{3.9}$$

For the last step of (3.9) and below we abbreviate $q(z_t^i)$ as q_{it} for notational simplicity. The derivation is easily attained by noting that the summation over $\{z_{t'}^{i'}\}$ for all $(i',t') \neq (i,t)$ yields 1. This is an observation that we will use frequently in later derivations.

The second term in the free energy

$$\begin{aligned}\mathbb{E}_{\mathbb{Q}(Z)}[\log \mathbb{P}(X, Z; \Theta)] =& \mathbb{E}_{\mathbb{Q}(Z)} \left[\sum_i \sum_t (\log \mathbb{P}(z_t^i|\text{pa}(z_t^i)) + \log \mathbb{P}(x_t^i|z_t^i)) \right] \qquad (3.10)\\
=& \sum_i \sum_t \sum_z \left(\prod_{(i't')} q_{i't'} \right) \left[\log \mathbb{P}(z_t^i|\text{pa}(z_t^i)) + \log \mathbb{P}(x_t^i|z_t^i) \right] \qquad (3.11)\\
=& \sum_i \sum_t \sum_{z_t^i, \text{pa}(z_t^i)} \left(q_{it} \prod_{(i't') \in \text{pa}(i,t)} q_{i't'} \right) \left[\log \mathbb{P}(z_t^i|\text{pa}(z_t^i)) + \log \mathbb{P}(x_t^i|z_t^i) \right]\\
=& \sum_i \sum_t \left[\sum_{z_t^i, \text{pa}(z_t^i)} \left(q_{it} \prod_{(i't') \in \text{pa}(i,t)} q_{i't'} \right) \log \mathbb{P}(z_t^i|\text{pa}(z_t^i)) + \sum_{z_t^i} q_{it} \log \mathbb{P}(x_t^i|z_t^i) \right]\end{aligned}$$

where, assuming there are no missing marks, the emission probability $\mathbb{P}(x_t^i|z_t^i)$ is given by

$$\mathbb{P}(x_t^i|z_t^i = k) = \prod_l \left(I(x_{t,l}^i = 1) e_l^k + (1 - I(x_{t,l}^i = 1))(1 - e_l^k) \right) \qquad (3.12)$$

For the expectation step (E-step), we isolate the terms in (3.9) and (3.10) that involve q_{it} :

$$\begin{aligned}F(q_{it}) =& \sum_{z_t^i} q_{it} \log q_{it} - \sum_{z_t^i, \text{pa}(z_t^i)} \left(q_{it} \prod_{(i',t') \in \pi(i,t)} q_{i't'} \right) \left[\log \mathbb{P}(z_t^i|\text{pa}(z_t^i)) + \log \mathbb{P}(x_t^i|z_t^i) \right]\\
& \qquad (3.13)\\
& - \sum_{z_{t'}^i, \text{pa}(z_{t'}^i) | t = t'+1} \left(q_{it'} q_{it} q_{\text{pa}(i),t'} \right) \log \mathbb{P}(z_{t'}^i|\text{pa}(z_{t'}^i))\\
& - \sum_{z_{t'}^{i'}, \text{pa}(z_{t'}^{i'}) | (i,t) = (\text{pa}(i'),t')} (q_{i't'} q_{it} q_{i',t'-1}) \log \mathbb{P}(z_{t'}^{i'}|\text{pa}(z_{t'}^{i'})).\end{aligned}$$

The above equation can be written as $\sum_{z_t^i}(q_{it}\log q_{it} - q_{it}\log\phi(z_t^i))$ with

$$\begin{aligned}
\log\phi(z_t^i) = &\sum_{\mathrm{pa}(z_t^i)}\left(\prod_{(i',t')\in\mathrm{pa}(i,t)} q_{i't'}\right)\left[\log\mathbb{P}(z_t^i|\mathrm{pa}(z_t^i)) + \log\mathbb{P}(x_t^i|z_t^i)\right] \\
&+ \sum_{z_{t'}^{i'}, z_{t'}^{\mathrm{pa}(i')}|t=t'+1}\left(q_{it'}q_{\mathrm{pa}(i),t'}\right)\log\mathbb{P}(z_{t'}^i|\mathrm{pa}(z_{t'}^i)) \\
&+ \sum_{z_t^{i'}, z_{t-1}^{i'}|i=\mathrm{pa}(i')}(q_{i't}q_{i',t-1})\log\mathbb{P}(z_{t'}^{i'}|\mathrm{pa}(z_{t'}^{i'})).
\end{aligned} \tag{3.14}$$

Subsequently q_{it} is obtained by normalizing $\phi(z_t^i)$,

$$q_{it}(z_t^i) = \phi(z_t^i)/\sum_{z_t^i}\phi(z_t^i)$$

The M-step seeks parameters that minimize the free energy F under the constraints $\sum_a \theta_{bc}^a = 1, \sum_a \alpha_b^a = 1, \sum_a \beta_b^a = 1, \sum_a \gamma^a = 1.$ where

$$\begin{aligned}
F &= \mathbb{E}_Q[-\log\mathbb{P}(X,Z;\Theta)] + C \\
&= -\sum_{i,t}(q_{it}\prod_{(i',t')\in\mathrm{pa}(i,t)} q_{i't'})\left[\log\mathbb{P}(z_t^i|\mathrm{pa}(z_t^i)) + \log\mathbb{P}(x_t^i|z_t^i)\right] + C,
\end{aligned} \tag{3.15}$$

where the entropy term, denoted as C, is now a constant given fixed $\mathbb{Q}(Z)$. We can group terms associated with each parameter matrix and we can easily see that the parameter values that maximize F are given by the average frequency of occurrence of each child-parent state.

More specifically, the parameter updates are given by

$$\theta^k_{mn} \propto \sum_{i>1,t>1} q(z^i_t = k) q(z^{\mathrm{pa}(i)}_t = m) q(z^i_{t-1} = n), \tag{3.16}$$

$$a^k_m \propto \sum_{i=1,t>1} q(z^i_t = k) q(z^i_{t-1} = m), \tag{3.17}$$

$$\beta^k_m \propto \sum_{i>1,t=1} q(z^i_t = k) q(z^{\mathrm{pa}(i)}_t = m), \tag{3.18}$$

$$\gamma^k \propto q(z^1_1 = k), \tag{3.19}$$

$$e^k_l = \frac{\sum_{i,t} q(z^i_t = k) I(x^i_{t,l} = 1)}{\sum_{i,t} q(z^i_t = k)} \tag{3.20}$$

where $I(\cdot)$ is the indicator function and $i = 1$ corresponds to the root species.

3.3.3 Structured mean field(SMF) variational inference

In the structured mean field variational method, we consider the surrogate distribution to be the product of the marginal distributions of disjoint sets of state variables. Let $\mathbf{z}_i = \{z^i_t : t = 1, \ldots, T\}$ denote the set of all state variables within cell type i, corresponding to the state variables within each horizontal chain of the TreeHMM model. We consider the $\mathbb{Q}$ to be of the following form

$$\mathbb{Q}(Z) = \prod_{i=1}^{I} q_i(\mathbf{z}_i), \tag{3.21}$$

written as the product of marginal distributions of $\mathbf{z}_i$ variables. In this case, the free energy becomes

$$F = \sum_{i=1}^{I} \left[\mathbb{E}[\log q_i(\mathbf{z}_i)] - \sum_{t=1}^{T} (\mathbb{E}[\log \mathbb{P}(z^i_t | z_{\pi(i,t)})] + \mathbb{E}[\log \mathbb{P}(x^i_t | z^i_t)]) \right]. \tag{3.22}$$

To find the optimal distribution $\mathbb{Q}$ that minimizes the free energy, we again alternatively optimize each marginal distribution component while keeping others fixed. To update $q_i(\mathbf{z}_i)$, we collect the terms in F that involve $q_i(\mathbf{z}_i)$,

$$F_i = \mathbb{E}_{q_i}[\log q_i(\mathbf{z}_i) - \sum_{t=1}^{T} \big(\log f_{it}(z_t^i, z_{t-1}^i) + \log \mathbb{P}(x_t^i | z_t^i)\big)], \tag{3.23}$$

where we have defined $f_{it}(z_t^i, z_{t-1}^i) = \exp\{\mathbb{E}_{q_{\pi(i)}}[\log \mathbb{P}(z_t^i | z_{\pi(t,i)})] + \sum_{j:i=\pi(j)} \mathbb{E}_{q_j}[\log \mathbb{P}(z_t^j | z_{\pi(j,t)})]\}$. Since f_{it} only involves expectations with respect to the distributions other than q_i, it is a fixed function of z_t^i and z_{t-1}^i during the update of the $q_i(\mathbf{z}_i)$. If the f_{it} functions can be normalized to be conditional probability distributions, then Equation (3.23) shares the exact form of the free energy of a hidden Markov model with transmission probabilities specified by f_{it} and emission probabilities specified by $\mathbb{P}(x_t^i | z_t^i)$. As such, the optimal q_i minimizing the free energy is the same as the posterior probabilities of the states in the hidden Markov model, which can be efficiently calculated using the forward-backward algorithm [27]. The details of how to normalize the f_{it} functions to be proper transition probabilities is shown in the next section, 3.3.4.

3.3.4 Structured mean-field derivation and normalization

In structured mean-field approximation, we assume that $\mathbb{Q}(Z)$ takes the following form

$$\mathbb{Q}(Z) = \prod_{i=1}^{I} q_i(\mathbf{z}_i),$$

where $\mathbf{z}_i$ is the group of hidden variables within ith chain.

We can write F as

$$
\begin{aligned}
F &= \mathbb{E}_Q[\log \mathbb{Q}(Z) - \log \mathbb{P}(X, Z; \Theta)] && (3.24)\\
&= \sum_{i=1}^{I} \mathbb{E}_Q[\log q_i(Z_i)] - \sum_{i=1}^{I}\sum_{t=1}^{T} \left(\mathbb{E}_Q[\log \mathbb{P}(z_t^i|\mathrm{pa}(z_t^i))] + \mathbb{E}_Q[\log \mathbb{P}(x_t^i|z_t^i)]\right)\\
&= \sum_{i=1}^{I} \mathbb{E}_{q_i}[\log q_i(\mathbf{z}_i)] - \sum_{i=1}^{I}\sum_{t=1}^{T} \left(\mathbb{E}_{q_i, q_{\mathrm{pa}(i)}}[\log \mathbb{P}(z_t^i|\mathrm{pa}(z_t^i))] + \mathbb{E}_{q_i}[\log \mathbb{P}(x_t^i|z_t^i)]\right).
\end{aligned}
$$

Again we isolate the terms that involves $q_i(\mathbf{z}_i)$

$$
\begin{aligned}
F &= \mathbb{E}_{q_i}[\log q_i(\mathbf{z}_i)] - \sum_{t=1}^{T} \left(\mathbb{E}_{q_i, q_{\mathrm{pa}(i)}}[\log \mathbb{P}(z_t^i|\mathrm{pa}(z_t^i))] + \mathbb{E}_{q_i}[\log \mathbb{P}(x_t^i|z_t^i)]\right) && (3.25)\\
&- \sum_{i'|i\in\mathrm{pa}(i')} \sum_{t=1}^{T} \mathbb{E}_{q_{i'}, q_i}[\log \mathbb{P}(z_{t'}^{i'}|\mathrm{pa}(z_{t'}^{i'}))] + const\\
&= \mathbb{E}_{q_i}[\log q_i(\mathbf{z}_i))] - \sum_{t=1}^{T} \mathbb{E}_{q_i}\left[\log f_{it}(z_t^i, z_{t-1}^i) + \log \mathbb{P}(x_t^i|z_t^i)\right],
\end{aligned}
$$

where we have defined

$$
f_{it}(z_t^i, z_{t-1}^i) \equiv \exp\left(\mathbb{E}_{q_{\mathrm{pa}(i)}}[\log \mathbb{P}(z_t^i|\mathrm{pa}(z_t^i))] + \sum_{i'|i\in\mathrm{pa}(i')} \mathbb{E}_{q_{i'}}[\log \mathbb{P}(z_t^{i'}|\mathrm{pa}(z_t^{i'}))]\right).
$$

Equation (3.25) from above shares the form of the free energy of a hidden Markov model if $f_{it}(z_t^i, z_{t-1}^i)$ has the normalization property of a transition matrix. To normalize it, we use the procedure outlined in 4, starting from the last node in the chain:

Algorithm 2 Strucutred Mean Field: Normalization Algorithm

```
**Initialize** $g(z_T^i) = [1, \dots, 1]$
**for** $t = (T, \dots, 2)$ **do**
    $\tilde{f}(z_t^i, z_{t-1}^i) = g(z_t^i) f(z_t^i, z_{t-1}^i)))$
    $g(z_{t-1}^i) = \sum_{z_t^i} \tilde{f}(z_t^i, z_{t-1}^i)$
    $\tilde{f}(z_t^i, z_{t-1}^i) = \tilde{f}(z_t^i, z_{t-1}^i) / g(z_{t-1}^i)$
**end for**
```

For $t = 1$, we get $\log(f_{i1}(z_1^i) g(z_1^i))$, which we can normalize to get the prior distribution of variable z_1^i.

$$\tilde{f}(z_1^i) = \frac{f_{i1}(z_1^i) g(z_1^i)}{\sum_{z_1^i} f_{i1}(z_1^i) g(z_1^i)}$$

where $f_{i1}(z_1^i) = \exp\left(\mathbb{E}_{q_{\mathrm{pa}(i)}}[\log \mathbb{P}(z_1^i | z_1^{\mathrm{pa}(i)})] + \sum_{i' | i \in \mathrm{pa}(i')} \mathbb{E}_{q_{i'}}[\log \mathbb{P}(z_1^{i'} | z_1^i)]\right)$.

Notice that now $\tilde{f}$ is properly normalized. Next we can use the forward-backward algorithm to infer the posterior distribution $q_i(z^i)$, thus the marginal $q(z_t^i)$ and $q(z_t^i, z_{t-1}^i)$. Omitting index i and defining the forward and backward messages $a_t^k = \mathbb{P}(x_1, \dots, x_t, z_t = k)$ and $b_t^k = \mathbb{P}(x_{t+1}, \dots, x_T | z_t = k)$, we can then perform forward-backward algorithm as below:

Algorithm 3 Strucutred Mean Field: Forward Algorithm

```
**Initialize** $a_1^k = e_k(x_1) \tilde{f}(z_1 = k)$
**for** $t = (2, \dots, T)$ **do**
    $a_t^k = e_k(x_t) \sum_j a_{t-1}^j \tilde{f}(z_{t-1} = j, z_t = k)$.
**end for**
```

where $\tilde{f}(z_1 = k), \tilde{f}(z_{t-1} = j, z_t = k)$ are the normalized prior and transition matrices calculated previously. $e_k(x_t) = \mathbb{P}(x_t | z_t = k) = \prod_{l=1}^{L} [(1 - e_l^k) I(x_{t,l} = 0) + e_l^k I(x_{t,l} = 1)]$.

Algorithm 4 Strucutred Mean Field: Backward Algorithm

```
Initialize $b_T^k = (1, 1, \ldots, 1)$
for $t = (T-1, \ldots, 1)$ do
    $b_t^k = \sum_j \tilde{f}(z_t = k, z_{t+1} = j) e_l(x_{t+1}) b_{t+1}^j$
end for
```

The likelihood is given by $\mathbb{P}(x) = \sum_{z_T} \mathbb{P}(x, z_T) = \sum_k a_T^k$. We can calculate the posterior distribution of hidden variables $\mathbb{P}(z_t = k|x)$ and $\mathbb{P}(z_t = k, z_{t+1} = j|x)$ as

$$\begin{aligned} q(z_t = k|x) &= \frac{a_t(z_t = k) b_t(z_t = k)}{\mathbb{P}(x)}, \\ q(z_t = k, z_{t+1} = j|x) &= \frac{a_t(z_t = k)\tilde{f}(z_t = k, z_{t+1} = j) e_l(x_{t+1}) b_t(z_{t+1} = j)}{\mathbb{P}(x)}. \end{aligned}$$

The M-step is similar to the mean-field case and is given below.

$$\begin{aligned} \theta_{mn}^k &\propto \sum_{i>1, t>1} q(z_t^{\mathrm{pa}(i)} = m) q(z_{t-1}^i = n, z_t^i = k), \\ \alpha_m^k &\propto \sum_{i=1, t>1} q(z_{t-1}^i = m, z_t^i = k), \\ \beta_m^k &\propto \sum_{i>1, t=1} q(z_t^{\mathrm{pa}(i)} = m) q(z_t^i = k), \\ \gamma^k &\propto q(z_1^1 = k), \\ e_l^k &= \frac{\sum_{i,t} q(z_t^i = k) I(x_{t,l}^i = 1)}{\sum_{i,t} q(z_t^i = k)} \end{aligned}$$

3.3.5 Loopy belief propagation (LBP)

The third inference method we used is loopy belief propagation. Belief propagation is a message passing algorithm commonly used in probabilistic graphical models. The algorithm is exact for tree and poly-tree structured graphs. For general graphs that contain cycles or loops, it is an approximate algorithm also called loopy belief propagation. In this case, the algorithm is not guaranteed to converge nor is the approximate free-energy a bound of the log-likelihood. Nevertheless, it has shown empirical success in some cases [78]. Loopy belief propagation can be also viewed as a variational method with the $\mathbb{Q}$ distribution taking the Bethe approximation form upon convergence [131]. Here we use Pearl's belief propagation algorithm which is directly applicable to the Bayesian network representation. For each node, the algorithm tracks two types of messages - one from its children and the other from its parents, both of which are $1 \times K$ vectors in our case (observed nodes are treated as evidence and no associated messages are incorporated). At the start of the algorithm, we initialize all the messages to unity vectors. During each iteration, all messages are updated simultaneously based on the messages from the previous iteration. The procedure is repeated until convergence. We refer readers to [23] for the details of the algorithm.

3.3.6 Parameter Learning

Above we have introduced different inference methods. To do parameter learning, we use a variant of the expectation-maximization (EM) algorithm called variational EM algorithm. Like the EM algorithm, the variational EM algorithm iterates between two steps: an expectation and a maximization step. The expectation step (E-step) is performed by the inference methods, during which we calculate $\mathbb{Q}(Z)$ in the approximate forms outlined above with fixed parameter values. In the maximization step (M-step), we seek parameter values that minimize F (or maximize $-F$) under $\mathbb{Q}(Z)$.

Consider the free energy F as a function of Θ, the variational maximization step seeks the parameters that minimize F given the current hidden variable distribution $\mathbb{Q}(Z)$, i.e.

$$\hat{\Theta} = \arg\min_{\Theta} F(\Theta, \mathbb{Q}(Z)).$$

The above optimization can be solved explicitly. As a result, the state transition parameters are calculated as $\theta^a_{bc} \propto \sum_{i>1}\sum_{t>1} \mathbb{Q}(z^i_t = a, z^{\pi(i)}_t = b, z^i_{t-1} = c)$, $\alpha^a_b \propto \sum_{t>1} \mathbb{Q}(z^1_t = a, z^1_{t-1} = b)$, $\beta^a_b \propto \sum_{i>1} \mathbb{Q}(z^i_1 = a, z^{\pi(i)}_1 = b)$, $\gamma^a \propto \mathbb{Q}(z^1_1 = a)$ up to a normalization constant, where $\mathbb{Q}(\cdot)$ denotes the marginal distribution of the variables inside the brackets. The emission parameters are given by $e^a_l = \frac{\sum_{i,t} \mathbb{Q}(z^i_t = a) I(x^i_{t,l}=1)}{\sum_{i,t} \mathbb{Q}(z^i_t = a)}$ where $I(\cdot)$ is the indicator function. The variational EM algorithm for the SMF case is outlined in Algorithm 1 1. Notationally, we have considered the entire genome as a single chunk. In practice, we break up the genome into many smaller chunks to allow more efficient, parallel execution and to reduce memory consumption, at the cost of computational artifacts at chunk borders.

3.4 Results

3.4.1 Artificial Data

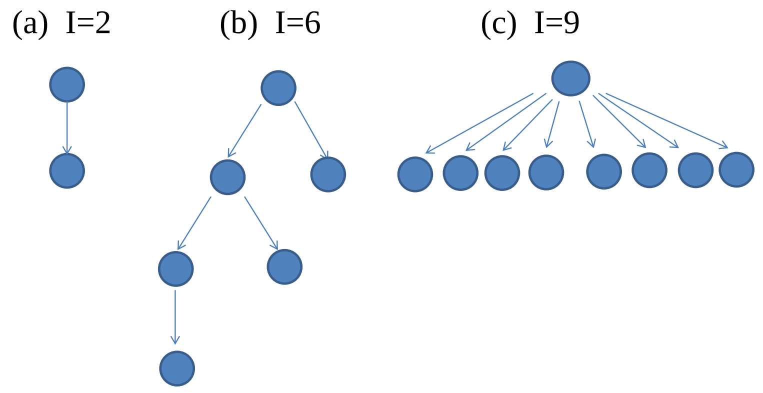


Figure 3.2: **Lineage trees used for artificial data**

We assess the accuracy of different approximations on artificial datasets generated under the TreeHMM model described above. We tested cases with different tree structures ($I \in \{2, 6, 9\}$) and number of states $K \in \{5, 10\}$ .For the cases of $I = 2$, we consider a lineage where species are connected in a line structure. For the case of $I = 9$, we adopt a tree structure with one being root and others being direct children. And for the case of $I = 6$, we have used a relatively deep tree, as depicted in Fig. 3.2. We also set the number of markers to be $L = 10$ respectively to mimic the real data. Parameters in α, β, γ and e are generated randomly between $[0, 1]$ and normalized to be a conditional probability table (except e). Sub-matrices $\theta(k, :, :)$ are generated by adding perturbations to α. We specify the model in the Bayes Net Toolbox (BNT) [77] and use the *sample_bnet* function in BNT to generate values for the observed nodes using the "true" parameter values. We generate datasets consisting of $T = 10^4$ slices for $K = 5$ and $T = 10^5$ for $K = 10$.

We run each variational EM algorithms on the generated artificial datasets. Since EM algorithm could reach local optimal solutions, we run each algorithm 5 times with random initializations for each dataset and take the result with the lowest free energy. We use the root mean squared error (RMSE) of the elements in each parameter matrix to measure the accuracies of the inferred parameters. To test the consistency of the learning algorithms, we generated five datasets using different parameters for each (K, I) pair, and record the averaged result of RMSE of inferred parameters w.r.t. the true ones in each parameter matrix.

From results summarized in Table 3.1, we observe that the SMF approximation consistently outperforms LBP and MF. Although LBP and MF can give good estimates of the emission matrix e for some simple cases (e.g. $K = 5$), they usually give a large error in the learned α and θ parameters, indicating that the two methods are less accurate in inferring joint marginals of hidden variables. We also observe that the performance of LBP decreases with increasing number of leaf cell types (e.g., percent error of $\alpha = 4.5\%$ for $I = 2$ versus

26% for $I = 9$ in $K = 5$ cases), indicating that LBP is sensitive to the tree structures and tends to infer inaccurate marginals when nodes in the graph have many connected nodes. Interestingly it performs well in I=6 case of relatively deep lineage tree. In contrast, SMF consistently gives accurate estimations for all the cases. SMF's advantage is especially apparent for estimating α and θ, which involve the joint marginal. For example, the percent error of $\theta = 9.4\%$ versus 63% for MF and 168% for LBP in the $(I, K) = (9, 10)$ case. Also we can see that, as expected, more data is needed to achieve the same accuracy for the inferred parameters as the number of states K (and therefore the number of parameters) increases.

Table 3.1: Accuracy of different algorithms in recovering parameter values using artificial data.

		RMSE (percent*)		
(K, I, T)	Approx	e	α	θ
(5, 2, 10K)	SMF	0.008 (1.5%)	0.009 (4.6%)	0.022 (11%)
	MF	0.03 (6.7%)	0.03 (17.1%)	0.04 (22%)
	LBP	0.011 (2.1%)	0.009 (4.5%)	0.28 (140%)
(10, 2, 100K)	SMF	0.005 (1.0%)	0.005 (4.8%)	0.013 (13%)
	MF	0.08 (15%)	0.055 (55%)	0.06 (59%)
	LBP	0.06 (12%)	0.026 (26%)	0.17 (170%)
(5, 9, 10K)	SMF	0.004 (0.8%)	0.009 (4.7%)	0.008 (4.2%)
	MF	0.04 (0.9%)	0.03 (18%)	0.04 (19%)
	LBP	0.022 (4.4%)	0.10 (52%)	0.30 (150%)
(10, 9, 100K)	SMF	0.009 (1.8%)	0.008 (8.3%)	0.009 (9.4%)
	MF	0.06 (23%)	0.05 (65%)	0.05 (63%)
	LBP	0.14 (26%)	0.20 (196%)	0.17 (168%)
(5, 6, 10K)	SMF	0.005 (1.0%)	0.015 (7.5%)	0.021 (11%)
	MF	0.04 (8.0%)	0.16 (80%)	0.16 (80%)
	LBP	0.04 (8.0%)	0.014 (70%)	0.08 (40%)

SMF: structured mean field, MF: mean field, LBP: loopy belief propagation

*Each value inside the bracket is the percent error relative to the mean value of the corresponding parameter matrix elements.

3.4.2 Data Processing for ENCODE Dataset

We preprocessed the datasets by dividing the genome into 200-bp non-overlapping bins and then binarized the reads within each bin, similar to [29]. For binarization, we assign value 1 if the total number of reads located within the bin is above the threshold corresponding to a p-value of 10^{-4} under a Poisson model, where the Poisson rate λ is the number of reads in all replicates of an experiment divided by the length of the genome.

To reduce computational cost, we segmented the genome into regions with and without chromatin marks and only use the regions with sufficient reads present. To do this, binned read counts across all species and all marks were summed together into a single track and convolved using a 1-D Gaussian kernel acting over $\sigma = 40$kb. Only regions with at least 0.5 smoothed reads across at least 5kb were retained as having sufficient signal to include in training. In total, these segments covered 54.8% of the genome and inference proceeded on each segment in parallel.

We used the same human ENCODE dataset reported in [29] which contains ChIP-seq profiles for nine human cell types including human embryonic stem cells (H1 ES), erythrocytic leukaemia cells (K562), B-lymphoblastoid cells (GM12878), hepatocellular carcinoma cells (HepG2), umbilical vein endothelial cells (HUVEC), skeletal muscle myoblasts (HSMM), normal lung fibroblasts (NHLF), normal epidermal keratinocytes (NHEK), and mammary epithelial cells (HMEC). For each cell type, ten different markers are used including eight histone modifications (H3K27me3, H3K36me3, H4K20me1, H3K4me1, H3K4me2, H3K4me3, H3K27ac, and H3K9ac), one transcription factor closely related to chromatin dynamics (CTCF), and a control data set (whole cell extract). Altogether, the dataset contains 90 ChIP-seq profiles, which were downloaded from the ENCODE website [84].

Since the cell types in the ENCODE data represent very diverse, distinct cell types, we used a simple lineage tree structure with the H1 ES cell type forming the tree root and all other cell types connecting to it directly as leaves. ES cells exhibit unique epigentic biology [9], however hierarchical clustering of the observed marks reveals that each mark exhibits substantial correlation between all cell types (see Supplemental Figure 3.12). Further, TreeHMM can incorporate information from marks that are only available in certain cell types and can be adapted to more interesting tree structures by including additional latent cell types. Although the current choice of tree structure may be an oversimplification of the underlying biology, we are mostly focusing on the *methodology* for approximate inference in TreeHMM; we explored the performance on artificial data with more interesting tree structures in Supplemental Material section 3.4.1. Finally, we note that while exact inference methods scale exponentially in the tree width, the approximate inference methods developed here scale linearly with I, allowing deeper lineages and more complex tree structures to be examined eventually.

3.4.3 Comparing Approximate Inference Methods

To determine the accuracy of our approximate inference methods, we apply the TreeHMM model to the human ENCODE dataset described above using the following scheme: Exact inference and learning are used to define a set of parameters at each iteration. Each of the approximate inference methods performs inference on the parameters' values to get the free energy. We apply this procedure on a randomly selected 2MB region with 3 cell types (H1 ES, K562, GM12878) using $K = 5$. Figure 3.3 shows the log likelihood of the exact inference and the corresponding free energy of different inference methods during exact EM iterations. We observe that the SMF approximation gives the highest negative free energy in this test dataset. The closeness between SMF free energy and the exact log likelihood indicates that the SMF method captures the majority of correlation between variables. Notably, the free energy curves of MF approximation and LBP fluctuate widely as the parameters are refined by the exact algorithm, indicating inconsistency in the free energy landscapes of these approximations and the true one. We also experiment with parameter recovery in several artificial datasets with different tree structures (Supplemental material section 3.4.1), and observe that SMF typically outperforms the other approximate methods. As SMF seems to be the most accurate approximation in both the artificial and real data cases, we proceed with the SMF approximation in the following real data genomic segmentation and prediction problems.

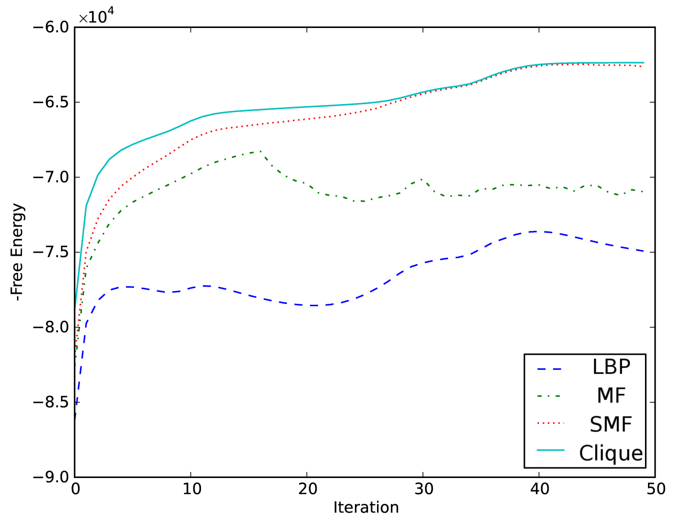


Figure 3.3: Free energy for approximate inference methods. Free energy for different inference methods are compared, with parameters learned using exact inference. The test dataset is restricted to a 2MB region of chromosome 22 with only three cell types and $K = 5$. The approximate methods use the parameters (learned by the exact method) and only perform inference steps. Note that for the exact algorithm (clique), the free energy equals the negative log-likelihood.

3.4.4 Model Complexity for Human ENCODE dataset

We seek to determine the model complexity, i.e. the number of chromatin states K, that is best supported by the human histone data. We expanded SMF inference to include all of chromosome 22 and all nine cell types, and varied the number of hidden states. Supplemental Figure 3.4 shows the free energy together with the complexity-penalized score according to Bayesian information criterion (BIC), assuming SMF free energy to be a close approximation to the true likelihood. To proceed with whole-genome analysis, we chose $K = 18$ where the maximum BIC score is achieved in this smaller dataset. We note that the value is close to the number of states (15) selected by [29] which was chosen partly by post-analysis of the assigned states.

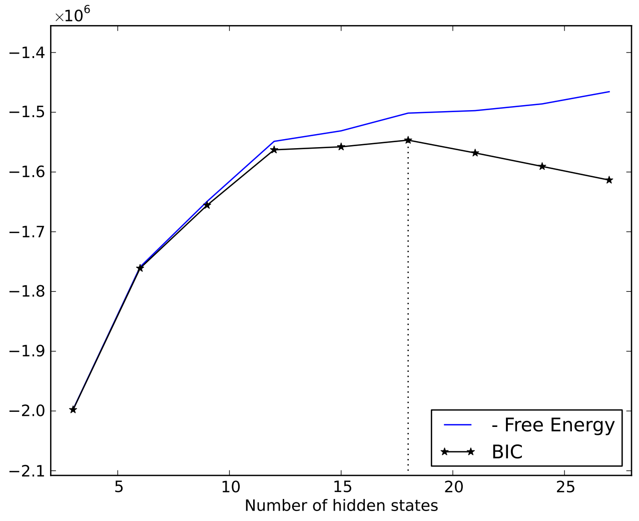


Figure 3.4: **Choosing model complexity (the optimal number of states K).** We ran the SMF variational EM algorithm on chromosome 22 with a range of K values . The final free energy for different K and the complexity-penalized likelihood score (BIC) is shown, revealing that a model with between 15 and 20 states are well-supported by the data.

3.4.5 TreeHMM on Complete Genomes using the SMF Approximation

We next apply the TreeHMM model's SMF approximation to the complete genomic histone data. We use the Bayes Information Criterion, a complexity-penalized likelihood, to determine the optimal number of states $K = 18$ (see Supplemental material section 3.4.4). After running several random initializations of the EM algorithm to convergence, we report the one with highest final likelihood. Figure 3.5 shows the learned states' characteristic chromatin modification co-occurrence patterns (the emission matrix e) and their enrichment in different genomic regions. Although states are learned *de novo* based only on the chromatin markers, many marker co-occurrences correspond to previous biological observations (e.g. H3K4 di- and tri-methylation in promoter regions and H3K4 mono- and di-methylation in enhancer regions [8]). We have annotated the likely function of each state (Figure 3.5) based on its genomic localization and concordance with previously reported findings [29]. The states show distinct enrichment patterns in different genomic locations. Several of the states (3, 8, and 11) are strongly enriched (8-17 fold) in the $\pm$2kb TSS region. Other states (7, 13, and 15) are enriched (2-3 fold) in coding genes. The coverage of each chromatin state region also varies widely, as shown in Table 3.3. The promoter and enhancer states cover a relatively small portion of the genome, e.g. $\sim 1.1\%$ for both active promoter and strong enhancer regions while low signal regions combine for around 75% of the genome. The state distribution also shows some cell-type specific properties, e.g., enhancer states 5, 10 and 11 are largely depleted in H1 ES cells, while other enhancer states are not (one being 2 fold enriched), indicating different functional roles of the learned enhancer states.

To explore the cell-type specificity of our learned states, we performed K-means clustering regions assigned to each state in any cell type. We show three of the states in Figure 3.13, including the insulator regions (state 14), strong enhancer regions (state 5) and active promoter regions (state 3). We can see that the distribution of different states across cell types differs drastically. Almost half of all insulator sites (state 14) are shared amongst all nine cell types or are only missing in one or two cell types. Many of the remainder are specific to a single cell type. Likewise, some active promoter regions (state 3) are shared amongst all or most cell types, but many more of the promoter regions are cell-type specific. Finally, the strong enhancer regions (state 5) are almost entirely cell-type specific. These overall patterns of cell-type specificity are captured by the learned transition matrices α and θ, which are shown in Supplemental Figures 3.7 and 3.8.

Several states are dominated by their vertical component in the θ transition matrix, including the states localizing to TSS's (states 2, 3, 8, 10, and 11), copy number variant/repeat regions (state 4), and the insulator state marked by CTCF (state 14). Other states have weak vertical components: consistent with the cell-type specificity of enhancers and chromatin remodeling, three of the enhancer regions (states 1, 5 and 6) and the polycomb repressed regions (state 17) show little to no vertical correlation. In particular, enhancer state 1 does not show the vertical correlation that might be expected given its propensity for TSS regions (4.24 fold enrichment).

3.4.6 Comparison with ChromHMM

We compare our result with ChromHMM - a similar method based on hidden Markov model described in [29] that does not utilize lineage information. We ran the HMM on the same histone data, treating each cell type's segment as a separate chain with inference performed in parallel but with tied parameters. We set number of states to be the same as in the TreeHMM result for consistency.

The learned emission probability matrix from ChromHMM together with the confusion matrix between the assigned states of the two results is shown in Supplemental Figure 3.9 (right panel). Comparing the emission matrix from two methods (Fig. 3.5 and Supplemental Figure 3.9 (left panel)), we observe similar co-occurrence patterns of markers. But as revealed by the confusion matrix, there is a substantial set of regions that are assigned different states due to the lineage constraint introduced in our model. For example, the weak promoter state (state 8) overlaps with ChromHMM 's inactive promoter and enhancer states (2 and 8). Also ChromHMM exhibits two repetitive states (similar to [29]) while there is only one such state in the TreeHMM result. To assess the accuracy of our methods, we tested our predicted states' overlap with several human ES-cell-specific ChIP-seq datasets.

We use a recent series of ChIP-seq datasets of transcription factor binding in H1-ES cells

Promoters				
Factor	TreeHMM		ChromHMM	
	All	Unique	All	Unique
Taf1	32,069 (41.6x)	1,489 (15.2x)	35,082 (26.0x)	4,502 (6.7x)
Oct4	4,980 (23.8x)	231 (8.7x)	6,932 (19x)	2,183 (12x)
Klf4	2,622 (18.1x)	105 (5.7x)	3,819 (15.1x)	1,302 (10.3x)
p300	141 (1.0x)	16 (0.9x)	1,597 (6.4x)	1,472 (11.8x)
Nanog	1,556 (1.5x)	227 (1.7x)	8,650 (4.7x)	7,321 (7.7x)
Sox2	412 (1.6x)	63 (2.0x)	2,509 (5.7x)	2,160 (9.8x)

Enhancers				
Factor	TreeHMM		ChromHMM	
	All	Unique	All	Unique
Taf1	8,095 (2.5x)	4,293 (4.4x)	5,611 (2.2x)	1,809 (5.3x)
Oct4	3,914 (4.5x)	2,060 (7.8x)	2,274 (3.3x)	420 (4.5x)
Klf4	2,143 (3.6x)	1,294 (7.1x)	1,003 (2.1x)	154 (2.4x)
p300	7,253 (12.2x)	1,517 (8.4x)	5,861 (12.2x)	125 (2.0x)
Nanog	39,829 (9.1x)	7,941 (6.0x)	33,561 (9.6x)	1,673 (3.5x)
Sox2	9,786 (9.4x)	2,185 (6.9x)	7,952 (9.5x)	351 (3.1x)

Table 3.2: H1-ES ChIP-seq enrichment in predicted promoter and enhancer regions.

[62] including Taf1, p300, Nanog, Klf4, Oct4, and Sox2. Among those, Taf1 is part of the machinery that recruits Polymerase II to the transcription start site and we expect its enrichment in promoter regions. p300 is a transcription factor (TF) that interacts with many other TFs in enhancer regions and we expect its presence in predicted enhancer regions. The other TFs in this dataset are important in maintaining stem-cell state, but a preference for promoter vs. enhancer has not been established. We investigated the overlap of ChIP-seq peaks in these datasets with our predicted states. For each method, we pooled the "enhancer" states (states 1, 5, 6, 10 and 11 in both methods) and report the fraction of sites overlapping called peaks for each transcription factor in Table 3.2. Similar results are reported for "promoter" regions (states 2, 3 and 8 in both methods).

As shown in Table 3.2, Taf1 shows strong enrichment in the promoter regions annotated by both ChromHMM and TreeHMM methods (26 and 41.6 fold enrichment over background, respectively). Although the two methods identify a similar number of active promoters (136,702 for TreeHMM vs. 239,792 by ChromHMM), a larger fraction of TreeHMM's predicted promoter overlaps with Taf1 binding sites than ChromHMM (32,069 or 23.5% of sites predicted by TreeHMM vs. 35,082 or 18.5% of sites predicted by ChromHMM). The enhancer regions predicted by the two methods with similar fold enrichment (12.2 and 12.3 fold) in p300 ChIP-seq binding peaks, but 24% more sites are correctly predicted by TreeHMM

(7,253 vs. 5,861). An interesting observation is that Oct4 and Klf4 both show preference for promoter regions over enhancer regions and in these cases, ChromHMM captures more of the ChIP-Seq binding sites but at the cost of calling many more total sites (23.8 vs. 19 fold enrichment of Oct4; 18.1 vs. 15.1 fold enrichment of Klf4). Distinctly, Nanog and Sox2 show a strong preference for enhancer regions. For these predictions, more ChIP binding sites (19% more for Nanog, 23% more for Sox2) are captured by TreeHMM at similar enrichment levels. These results indicate TreeHMM's lower false negative rate for enhancer regions and lower false positive rate for promoter regions.

We also investigated the enrichment of the CTCF motif in predicted insulator regions. CTCF is a transcription factor often associated with insulator regions and helps segment different regions of the genome. Its DNA binding motif is well-defined and it is often found in ChIP-seq determined CTCF binding sites [54]. To compare the accuracy of the predicted insulator regions, we searched through these regions for the presence of the CTCF motif. Among the 211,282 insulator regions predicted by TreeHMM in at least one cell type, 33,002 (15.6%) contained the CTCF motif. ChromHMM identified nearly twice as many insulator regions (388,788), of which only 43,111 (11.1%) contained the CTCF motif. Of the 186,735 sites uniquely predicted by ChromHMM, only 5.7% actually contain the CTCF motif compared to 7.2% of the 9,230 sites uniquely predicted by TreeHMM, indicating that many of the additional sites predicted by ChromHMM are likely to be spurious. In this case, TreeHMM offers increased accuracy and fewer spurious sites (false positives) over ChromHMM at the cost a few missed sites (false negatives).

3.5 Discussion

We have here presented a tree hidden Markov model for identifying chromatin state based on measurements from multiple cell types in a principled way. The major improvement over the

State	mean	H1 ES	K562	GM12878	HePG2	HUVEC	HSMM	NHLF	NHEK	HMEC
1	1.0	2.02	0.72	0.95	0.17	0.87	0.94	0.84	1.21	1.28
2	0.7	0.39	0.59	1.15	2.84	0.42	1.1	1.14	0.68	0.7
3	1.1	0.58	0.97	1.17	2.04	0.58	0.88	1.04	0.91	0.83
4	0.1	0.84	2.21	0.85	0.87	0.9	0.82	0.82	0.84	0.85
5	1.5	0.4	0.53	0.78	0.05	1.54	1.46	1.15	1.43	1.66
6	4.0	1.08	1.05	0.88	1.36	0.81	0.79	0.89	0.94	1.19
7	4.3	0.24	1.01	1.2	1.12	0.89	1.67	0.98	0.98	0.91
8	0.7	1.26	0.56	1.02	1.85	0.7	0.99	0.97	0.81	0.84
9	1.3	2.93	0.99	0.88	0.61	0.91	0.91	0.61	0.51	0.66
10	0.5	0.09	0.48	0.8	3.54	0.36	1.06	1.21	0.73	0.74
11	1.1	0.18	1.57	1.46	0.15	1.56	1.0	0.68	1.48	0.93
12	11	5.29	0.42	0.38	0.47	0.37	0.45	0.53	0.48	0.61
13	0.8	0.12	1.61	1.14	0.9	0.86	1.7	0.95	1.03	0.69
14	1.3	1.04	1.19	0.95	0.95	0.96	0.96	1.14	1.04	0.76
15	1.6	2.09	1.26	0.36	1.65	1.06	0.78	0.41	0.9	0.48
16	15	0.76	0.95	0.98	0.99	0.96	1.19	1.07	1.11	1.0
17	6.5	0.35	1.43	0.46	1.82	1.43	0.88	1.01	1.08	0.54
18	48	0.21	1.08	1.23	0.95	1.13	1.03	1.13	1.08	1.17
	(%)					(fold)				

Table 3.3: **Average coverage and cell type-specific enrichment of learned chromatin states.** The "mean" column gives the average percent coverage, or the fraction of the segmented genome which is covered by the state. The following columns show the enrichment of each state in each cell type relative to the across-species mean (column 2).

previous HMM approach is the incorporation of cell lineage explicitly in the model. While previous methods have focused only on the marks present at a particular region in a particular cell type, we pool information across the same genomic location at different cell types. This allows increased discernment in regions of uncertainty. Although model learning in our proposed model is intractable except in the smallest cases, we developed several approximate methods and demonstrated their accuracy using the ENCODE histone modification data for

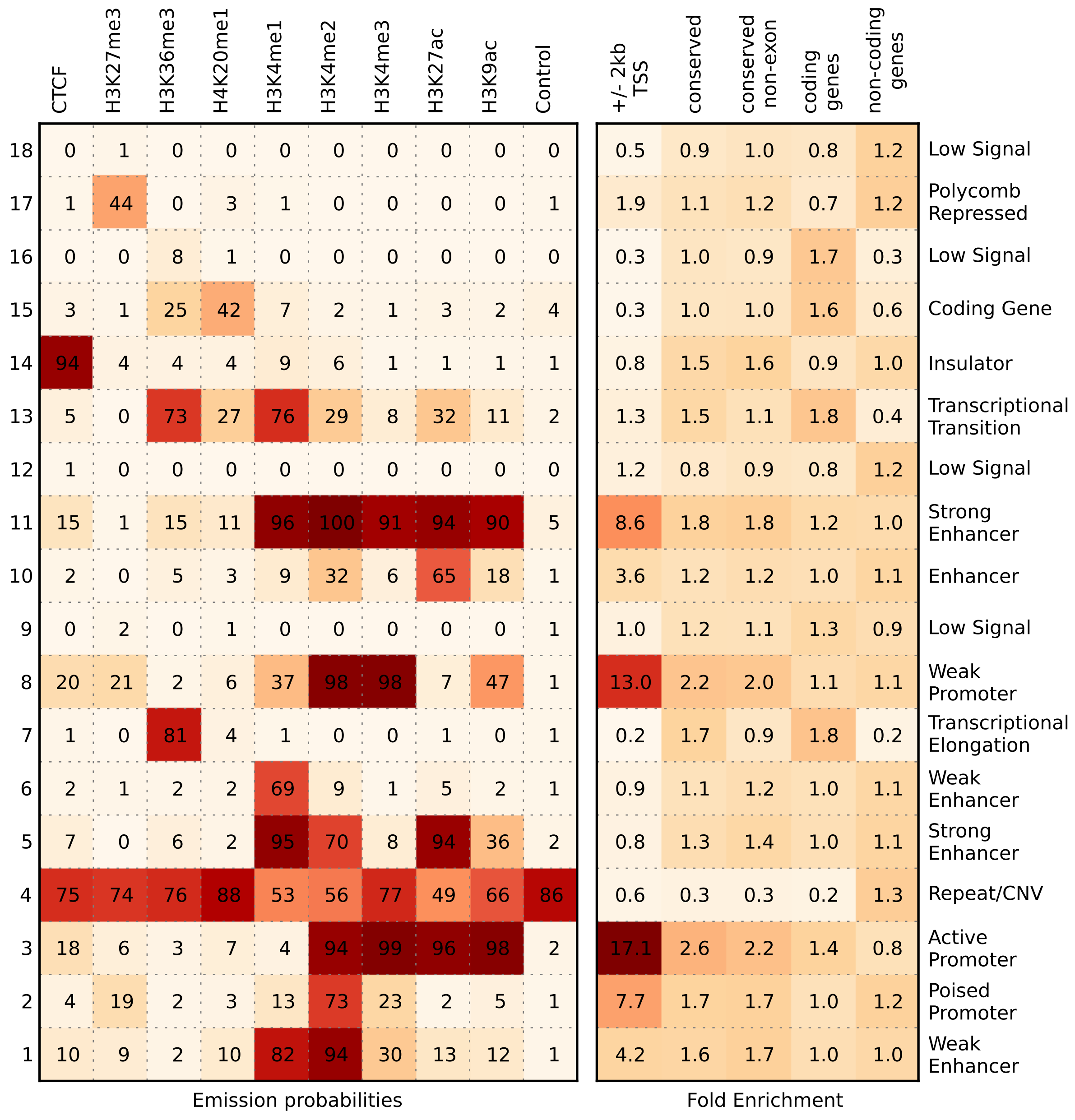
Table 3.4: Measured histone markers of Bone Marrow, G1E and G1E-ER4 - mouse dataset.

	H3K27ac	H3K27me3	H3K36me3	H3K4me1	H3K4me3	H3K9ac	H3K9me3
Bone Marrow	●			●	●		
G1E		●	●	●	●	●	●
G1E-ER4		●	●	●	●	●	●

nine different cell types. Interestingly, we found strong correlations along cell lineages and show that in many cases the information gained from lineage correlations increases state inference accuracy. Inherent to our method is the discovery of states that are more likely to change during differentiation or disease progression. This information allows more accurate prediction and allows accurate delineation between housekeeping genes present in all cell types and genes regulated in a lineage-specific fashion.

In this work, we have focused on developing approximate methods for doing inference and learning in the general framework. Our implementation is general and can deal with missing marks and missing species (discussed in section 3.2.4). With the capabilities of the model, there can be many further improvements including incorporating more cell types with incomplete measurements, modifying the lineage tree to include hidden nodes, and incorporating heterogenous data beyond histone marks. By pooling information from similar cell types and learning combinations of marks, it should be possible to infer cell state without a full spectrum of histone modifications measurements. We plan on exploring the rapidly increasingly heterogenous datasets to gain further insight into role of chromatin modifications in determining epigenetic states and their relationship with disease phenotype. Another possible application of the framework is to look into cross-species correlation of histone modification [56] to gain insight into inter-species conservation or divergence of epigenetic mechanisms.

Understanding epigenetic factors' associations with cell state is a primary step towards proper context for biological systems. Histone modifications play an essential role in regulating and maintaining gene expression and determining cell state. We have developed a novel graphical model for determining chromatin state from epigenetic modifications. Our method explicitly models transitions between cell types during differentiation or disease progression by considering cell lineage relationship. Although performing exact inference in our model is intractable, we develop highly accurate approximate inference methods that scale well with dataset size. By utilizing information from several cell types, our method can infer epigenetic state more accurately and has the ability to incorporate tendency of transitions between cell states in a more principled way. These cross-cell type correlations may be especially useful in datasets where the complete battery of experiments have not been performed in all cell types.



	CTCF	H3K27me3	H3K36me3	H4K20me1	H3K4me1	H3K4me2	H3K4me3	H3K27ac	H3K9ac	Control	+/- 2kb TSS	conserved	conserved non-exon	coding genes	non-coding genes	
18	0	1	0	0	0	0	0	0	0	0	0.5	0.9	1.0	0.8	1.2	Low Signal
17	1	44	0	3	1	0	0	0	0	1	1.9	1.1	1.2	0.7	1.2	Polycomb Repressed
16	0	0	8	1	0	0	0	0	0	0	0.3	1.0	0.9	1.7	0.3	Low Signal
15	3	1	25	42	7	2	1	3	2	4	0.3	1.0	1.0	1.6	0.6	Coding Gene
14	94	4	4	4	9	6	1	1	1	1	0.8	1.5	1.6	0.9	1.0	Insulator
13	5	0	73	27	76	29	8	32	11	2	1.3	1.5	1.1	1.8	0.4	Transcriptional Transition
12	1	0	0	0	0	0	0	0	0	0	1.2	0.8	0.9	0.8	1.2	Low Signal
11	15	1	15	11	96	100	91	94	90	5	8.6	1.8	1.8	1.2	1.0	Strong Enhancer
10	2	0	5	3	9	32	6	65	18	1	3.6	1.2	1.2	1.0	1.1	Enhancer
9	0	2	0	1	0	0	0	0	0	1	1.0	1.2	1.1	1.3	0.9	Low Signal
8	20	21	2	6	37	98	98	7	47	1	13.0	2.2	2.0	1.1	1.1	Weak Promoter
7	1	0	81	4	1	0	0	1	0	1	0.2	1.7	0.9	1.8	0.2	Transcriptional Elongation
6	2	1	2	2	69	9	1	5	2	1	0.9	1.1	1.2	1.0	1.1	Weak Enhancer
5	7	0	6	2	95	70	8	94	36	2	0.8	1.3	1.4	1.0	1.1	Strong Enhancer
4	75	74	76	88	53	56	77	49	66	86	0.6	0.3	0.3	0.2	1.3	Repeat/CNV
3	18	6	3	7	4	94	99	96	98	2	17.1	2.6	2.2	1.4	0.8	Active Promoter
2	4	19	2	3	13	73	23	2	5	1	7.7	1.7	1.7	1.0	1.2	Poised Promoter
1	10	9	2	10	82	94	30	13	12	1	4.2	1.6	1.7	1.0	1.0	Weak Enhancer

Figure 3.5: Learned chromatin states with the associated chromatin modification and enrichment in distinct genomic regions - human data. Left panel: the probability of observing each histone mark in each of 18 hidden states is summarized. Right panel: fold enrichment of hidden states in various genomic regions reveals strong positional preferences of learned chromatin states.

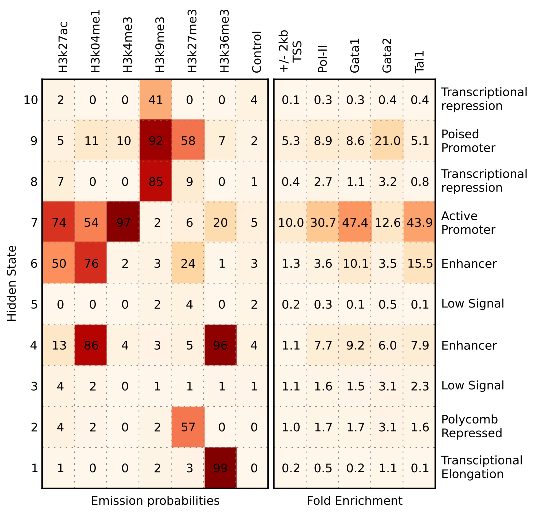


Figure 3.6: Learned chromatin states with the associated chromatin modification and enrichment in distinct genomic regions - mouse data. Left panel: The probability of observing histone marks in each inferred state is summarized. Right panel: fold enrichment of hidden states in various genomic regions reveals strong positional preferences of chromatin states. The values are average enrichments across all 3 cell types.

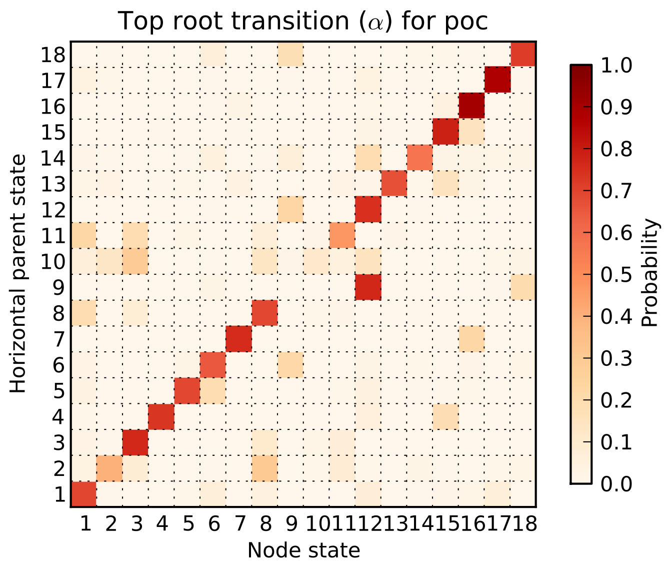


Figure 3.7: Learned transition matrix α for the root cell type (H1 ES).

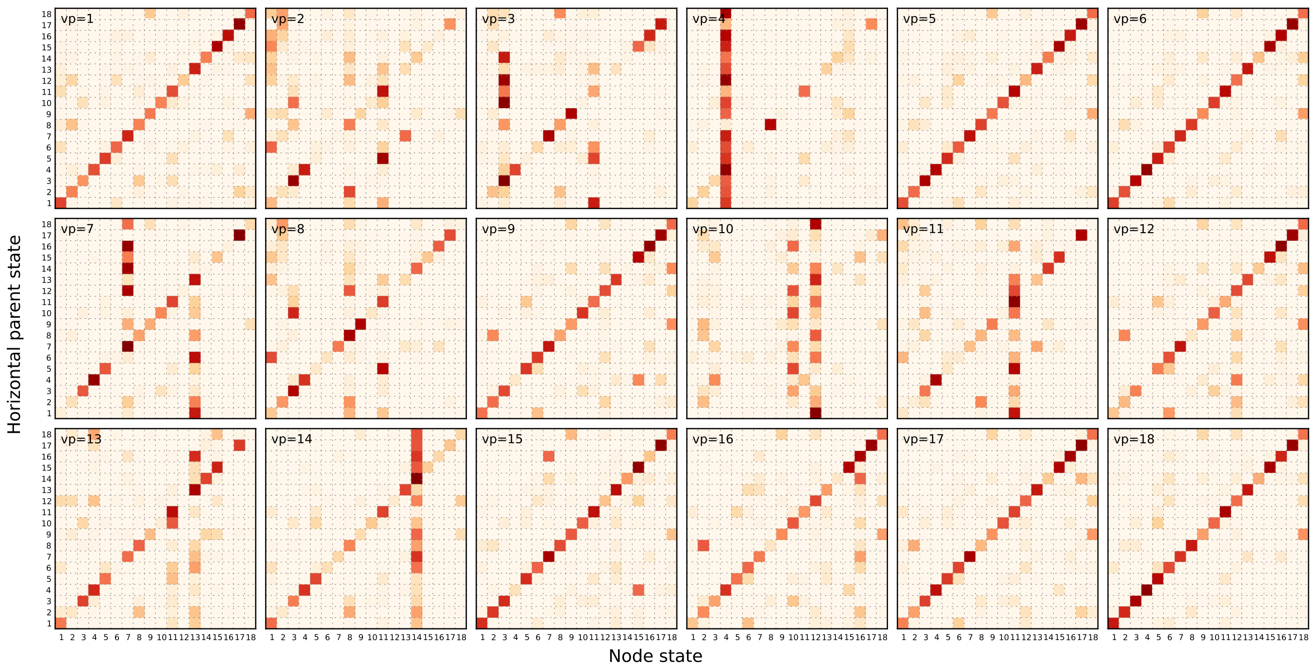


Figure 3.8: **Transition matrix θ shown as specific submatrices for each vertical parent state** Each sub-matrix is associated with vertical parent state 1 to 18 indicated at the top-left. A strong diagonal in the transition matrix indicates spatial persistence of states, while a strong vertical line indicates persistance from the corresponding vertical state across cell types.

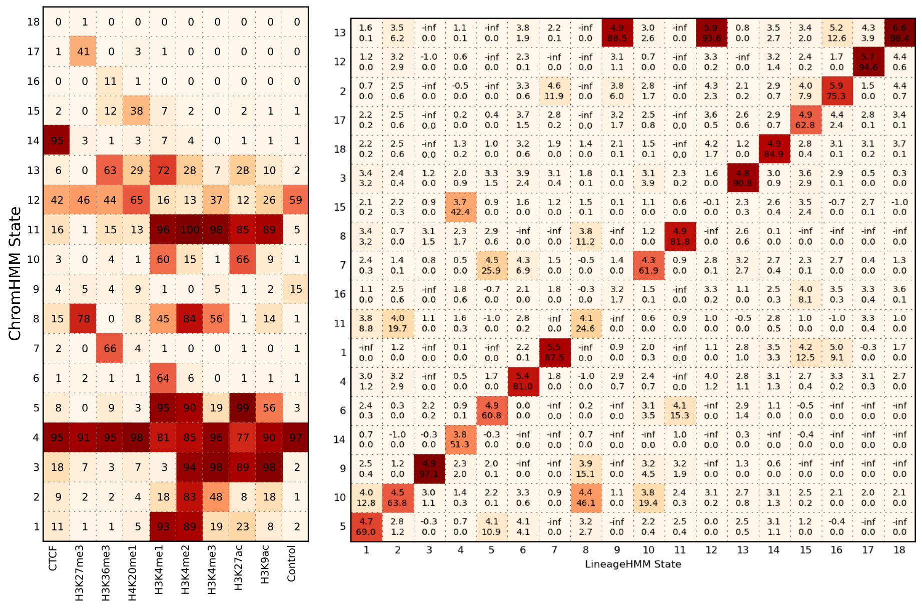


Figure 3.9: **Comparison with ChromHMM.** Left panel: Emission probabilities learned by ChromHMM with $K = 18$. Right panel: Confusion matrix showing the total number of bases identified as belonging to a particular state by the two methods. In each cell, the top number is the log_{10}(number of bins) in the intersection, the value below corresponds to the percentage in the corresponding tree-HMM sites and is used to color the heatmap. (values in each column sum up to 1.)

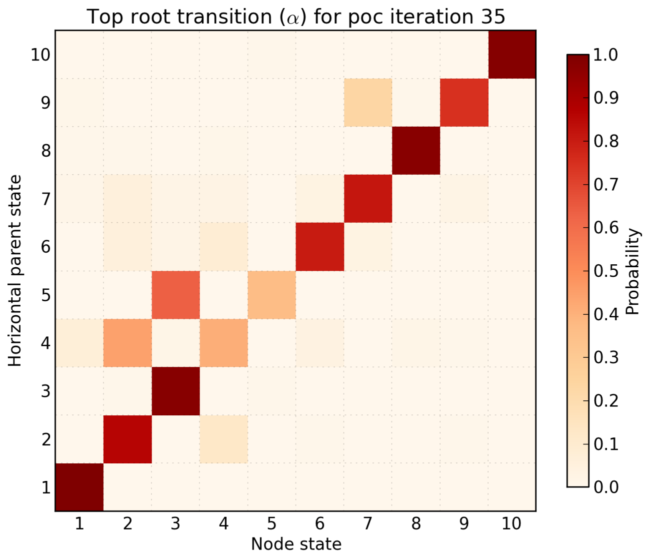


Figure 3.10: **Learned transition matrix α for the root cell type - mouse dataset**

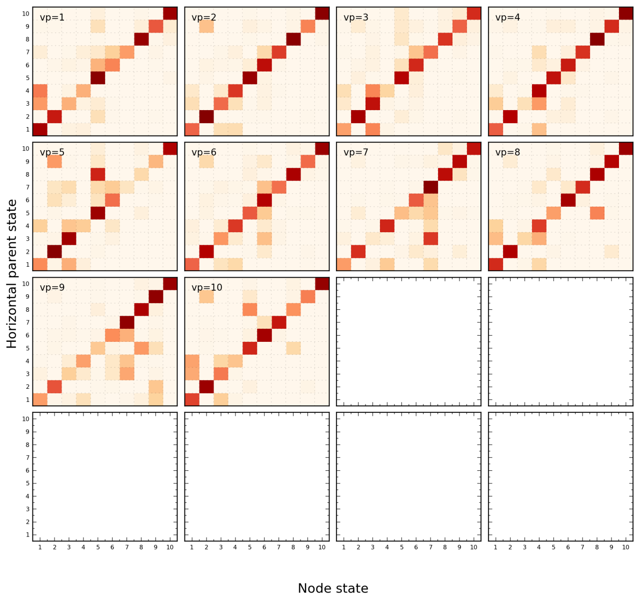


Figure 3.11: **Transition matrix θ shown as specific sub-matrices for each parent state - mouse dataset** Each sub-matrix is associated with vertical parent state 1 to 10 indicated at the top-left.

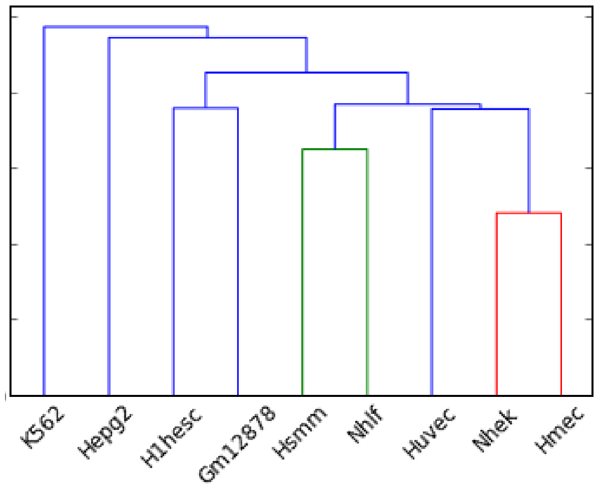


Figure 3.12: Hierarchical clustering of binarized histone marks in human. Branch lengths indicate the manhattan distance between species, considering all histone marks. The two most similar cell types are indicated in red, and the next two most similar are indicated in green.

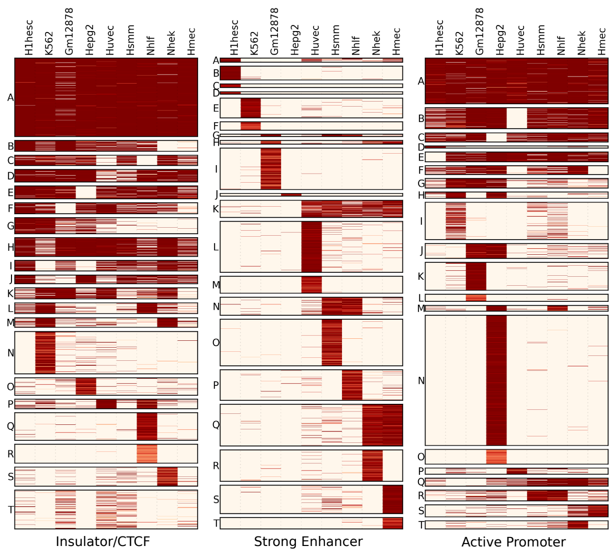


Figure 3.13: **K-means clustering reveals different cell-type specificities for different chromatin states.** K-means clustering with 20 clusters was performed on the posterior probability of all the bins being in each state, posterior probability is shown as heatmap color, cell types are indicated at the top of each column. From left to right: Insulator/CTCF (state 14), Strong Enhancer (state 5), and Active Promoter (state 3).

Chapter 4

Genomix: Scalable *de-novo* genome assembly

4.1 Introduction

Genome assembly is the attempt to reconstruct the original DNA sequence of a genome from short, error-filled, overlapping substrings of the genome. The problem is somewhat akin to attempting to reconstruct a written novel after finding thousands of shredded, error-filled copies of it. Assembly is a classic problem in computational biology with roots that date back to the original substring similarity algorithms of protein alignment, introduced in the early 70's and 80's [82, 108]. Assembly played a critical role in the success of the human genome project, completing in April 2003. Celera's approach to the problem involved so-called "shotgun-sequencing" and required extensive computational efforts to reconstruct the original genome from fragments that were considered short at the time. Their rapid progress was made possible by computational and algorithmic power. The genomes representing many organisms have since been assembled at least to draft quality, but assembly continues to play an important role in biology. Assembly is a first step to when studying a new organism, but also continues to play a role in finding structural variation present in a sample that would be difficult to identify if using reference-based variation discovery (see [115, 120] among many other examples).

Since the inception of the human genome project, the length, quality, and quantity of the read data has completely changed and gone through several phases. In the early days, Sanger sequencing was employed to derive long (>500bp) reads of high quality but the method suffered from low throughput and high costs. The original "next-generation" sequencing came in the flavor of lower quality, extremely short reads (36bp) that were counted by the millions instead of thousands. Further recent advancements have greatly improved the quality, read length (now 100-250bp x 2 pairs), and throughput (with hundreds of millions of reads generated from a single experiment). Some technologies currently offer even longer reads (hundreds to thousands of bases long) at reduced quality and throughput.

Assembly methods have varied widely over the same space in time, with algorithms being invented and adapted to better capture the strengths and account for the weaknesses of changing sequencing technologies. Early attempts used greedy methods to connect reads together and assemble the genome [43, 111]. Later methods drew heavily on algorithms from graph theory to reconstruct the genome. Initially, they relied on there being long stretches of overlap between the reads and were well suited to Sanger's long read technology. These methods built some variant of the String Graph, where the reads were represented by nodes with edges connecting overlapping reads (see [79] for an example and review). Reads from early next-generation sequencing experiments were too short and too numerous to be used in such an overlap graph, and, instead, assemblers were built based on the de Bruijn graph. De Bruijn graphs break the reads themselves into smaller kmers, or substrings of length K. De Bruijn graphs have the desirable property that if there are no errors in the reads, the size of the built graph is correlated to the size of the genome rather than the depth of coverage or number of reads in the data. Several attempts have been made to reconstruct the genome using de Bruijn graphs.

4.1.1 Contributions

We introduce a de Bruijn graph assembly method which combines several algorithms from popular assembly programs and includes a new approach to expanding contigs. The implementation is built on Hyracks and Pregelix and scales well with respect to the genome size. Further, the framework is inherently adaptable, able to be run on large clusters of commodity machines (such as Amazon's EC2 cloud), in more traditional grid environments, or even on a single machine with limited memory and computational resources.

4.1.2 Chapter Outline

This chapter is structured as follows: after discussing related work in section 4.1.3, explain our methods in section 4.2, in particular, we explain the mechanics of the de Bruijn graph in section 4.2.1 then proceed to explain the algorithms for cleaning and compressing the graph as well as expanding the contigs in sections 4.2.3 through 4.2.8. In section 4.3, we show our results comparing our method with Ray and Velvet on three different genomes. Finally, we conclude with a discussion about future work in section 4.4.

4.1.3 Related Work

Several previous efforts have been made towards genome assembly using de Bruijn graphs. A popular, early attempt at assembly was made by Velvet [135], which continues to be improved by the authors. Velvet was one of the first methods to identify patterns in the de

Bruijn graph caused by sequencing errors and proposed several serial algorithms for pruning them from the graph. All-Paths [19] and its several derivatives [37] used similar methods, but attempts to expand large contigs by searching for paths that connect them. Among several other notable methods [20, 65, 92], three are expecially pertinent to this work, as they partially address the problem of scalability: 1) ABySS [106], 2) Ray [15], and 3) Contrail [100]. The two former methods are based on the message-passing interface (MPI) framework and effectively distribute the de Bruijn graph across multiple physically distinct computers. While these methods do much to relieve the scalability problem, they are still limited in their ability to handle very large genomes or very deep sequencing coverage with high error rate, being limited to the available main memory of the computers they run on. The latter method, Contrail, is based on the Hadoop framework, and mutates on-disk representations of the graph. Unfortunately, in our testing, Contrail has an extremely long runtime. Hyracks, our underlying framework, is billed as a more efficient and flexible implementation of the "Big Data" tools normally solved by Hadoop.

4.2 Methods

Genome assembly attempts to reconstruct the original sequence of a genome from short, error-prone fragments taken from a non-uniform, random shearing process. The first algorithms for genome assembly used Sanger sequencing to determine the sequence of the fragments, yielding relatively few fragments that were long and highly-accurate. These algorithms Recently, next-generation sequencing While there are several approaches to genome assembly, the de Bruijn graph representation has proven popular and effective, particularly for reads derived from high-throughput sequencing experiments.

4.2.1 De Bruijn Graph Overview

We use the de Bruijn graph to represent the possible assemblies induced by a read dataset. For a given integer $k > 0$ and an alphabet with l unique symbols, Σ^l, a de Bruijn graph is a graph whose vertices V are composed of all possible length-l sequences from the alphabet. Two vertices, A and B are connected by a directed edge $A \rightarrow B$ if and only if the sequence of B is equivalent to the sequence of A left-shifted by a single symbol. Thus, two nodes in the graph are connected if they have an overlap of $k - 1$ symbols. De Bruijn graphs are Eulerian and Hamiltonian and contain l^k total vertices.

In genome assembly, we typically mean the *subset* of the de Bruijn graph induced by a particular dataset. The vertices of the data-induced graph are those kmers found in the reads and two vertices are connected by an edge if and only if a $k + 1$-mer was present in the read data. For example, the de Bruijn graph for $k = 4$ induced by the single read $AACCGGTT$ would consist of the vertices $AAC \rightarrow ACC \rightarrow CCG \rightarrow CGG \rightarrow GGT \rightarrow GTT$. If the reads contain no errors, covers the genome without gaps, and if the genome contains no repeated sequences longer than $k - 1$, the resulting de Bruijn graph will contain the original genome as one of its Eulerian paths (or tours for circular genomes).

Eulerian paths seek the shortest path covering the entire genome... May still be errors that shouldn't be incorporated. As [69] points out, it's not actually the shortest genome we're looking for. It's the one that's most consistent with the reads. The shortest one will collapse repeat regions. In their work, Medvedev et al. attempt to statistically determine the multiplicity of each edge's sequence in the genome and then apply a series of transformations to the graph, simplifying it where possible and consistent with the edge multiplicity. Finally, rather than finding an Eulerian tour of the graph (touching every edge exactly once), they solve the Chinese Postman Problem (finding the minimum-cost tour), minimizing the discrepancy between their tour and the estimated multiplicity of each edge in the genome.

Previous work has attempted to relax each of the above constraints with mixed performance; no single method has come to dominate the genome assembly landscape. Some errors can be resolved by inspecting the graph structure or through pruning particular nodes or edges based on some additional data such as coverage of the kmers. Gaps in genomes can be partially mediated by scaffolding together contigs (large stretches of unambigous sequence) using paired-end read data. Repeated stretches of the genome will be collapsed in the de Bruijn graph, leading to directed loops.

We build on the Hyracks and Pregelix distributed data frameworks to implement our assembly algorithms. Hyracks [16] is an efficient distributed dataflow engine, handling all aspects of data distribution and operator parallelization in exchange for an extensible "operator" interface. The framework and interface is similar to MapReduce/Hadoop [24, 105, 125], but is more flexible, efficient, and comes with several high-performance index structures out of the box. The framework assumes a shared-nothing cluster of computers with limited memory, though it runs very well in more traditional grid environments, where filesystems are often shared amongst machines or large quantities of memory are available. Unlike Hadoop, Hyracks explicitly manages buffer pools to keep as much of the data in-memory as possible and only spills to disk when it is insufficient.

Pregelix [18] is a Hyracks-based implementation of the Pregel API [66], providing users with a flexible, efficient interface for implementing graph algorithms as Bulk-Synchronous Parallel jobs. In this framework, all user-defined work occurs during separate iterations, with nodes communicating through messages that will be delivered *en masse* at the start of the next iteration. All nodes are addressable using their unique key and can save state between iterations using a user-defined vertex value.

4.2.2 Notation

We define notation for our algorithms and several functions. We denote the provided reads as R^i, with $i \in \{1 \ldots I\}$, the number of reads in the input, and each read being a string of length L. We define $subseq(sequence, j, k)$ as the substring of the string *sequence*, starting at position j and including k characters. We define $min(sequence)$ as the smaller of *sequence* and its reverse-compliment. $emit(key, value)$ is the function which passes the $(key, value)$ pair on to the next operator. The user is required to specify the length of the kmers, denoted K.

Each input read represents only one strand of the original DNA fragment and the same fragment may have both strands sequenced. We group both the kmer and its reverse-compliment in the same key/value pair and indicate the relationship between kmers by an edge type, detailed in Figure 4.1. This system is effectively the same as that of [70] but distributes the relationship between nodes to the nodes themselves rather than to a global table. It is also used in [100].

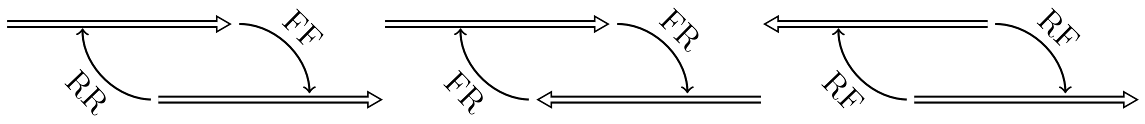


Figure 4.1: The three types of overlap and the four corresponding edge types used to indicate the relationship between nodes. RF and FR edges are symmetric with themselves.

4.2.3 Graph Building using Hyracks

For graph building, we define a Hyracks operator that converts input fastq files into a distributed key/value store. The keys are the full set of kmers seen in the input file and values contain neighbor information and some metadata such as node coverage and, for some nodes, the set of reads that passed through the node. Comparing the kmers in each read with their reverse-compliment, we keep the keys that are lexicographically smaller of the two.

Algorithm 5 Graph Building

```
**Map Operator:**
for all $i \in \{1 \dots I\}$ in parallel do
  for all $j \in \{1 \dots L - K\}$ do
    $emit\,(min(subseq(R^i, j, k)), \{min(subseq(R^i, j-1, k)), min(subseq(R^i, j+1, k))\})$
  end for
end for
Group-by Operator
Reduce Operator:
for all $kmer \in keys$ in parallel do
  $emit\left(kmer, \bigcup_{v \in values(kmer)}\right)$
end for
```

4.2.4 Remove Bad Coverage

Immediately after graph building, we remove any nodes whose coverage is outside of an acceptable range. That is, only nodes with $min \leq coverage \leq max$, where min and max are user-defined parameters, defaulting to 2 and 1000, respectively. Nodes with extremely low coverage are almost certainly caused by sequencing errors. A single sequencing error can create up to $2K - 1$ otherwise absent kmers, one for each overlapping kmer. Nodes with extremely high coverage probably arise from highly-repeated regions of the genome which we don't attempt to resolve. Immediately after graph building, each node inspects its coverage. If the coverage is outside the acceptable range, the node is pruned and a message is sent to all of its neighbors, informing them to remove edges back to the removed node.

4.2.5 Graph Compression

There is overhead associated with storing each node and in performing each iteration. One way to reduce this overhead is to compress the graph where possible. Two nodes that have $degree = 1$ towards each other can be compressed together without loss of information, but providing a significant reduction in the number of nodes in the graph and the number of iterations required for searching for particular patterns in the graph. Our implementation of graph compression is inspired by that in [100], which in turn is based on algorithms from [117].

As shown in Figure 4.2, compressing the graph requires coordination between neighbors; some nodes subsume other nodes while other nodes are subsumed and removed from the graph. Coordination is done between neighbors in the graph through a controlled random number generation: each node flips a coin and “head” nodes subsume “tail”, becoming longer and replacing the edge towards the “tail’ node with the tail node’s next edges. The “tail” node is removed from the graph and informs its neighbors to update their edges to point to the “head” node. To coordinate this process, each node must know the head/tail state of itself and all its neighbors. Neighbor state is made available without communication by seeding the random number generator with the kmer key of the neighbor (or self to determine a node’s own state) and the iteration number. In this way, each node knows their neighbor’s state, even if that neighbor hasn’t yet flipped the coin to determine state.

One further coordination step is necessary before the merge can take place: those nodes being subsumed must update their neighbors with the edges that will be present after the merge. Since those neighbors may themselves be subsumed in the same iteration, we take an extra iteration to coordinate edge updates across the graph. Figure 4.2 shows the state of the graph after each of the two iterations required for one round of merging. Merging continues until there are no more compressible nodes in the graph.

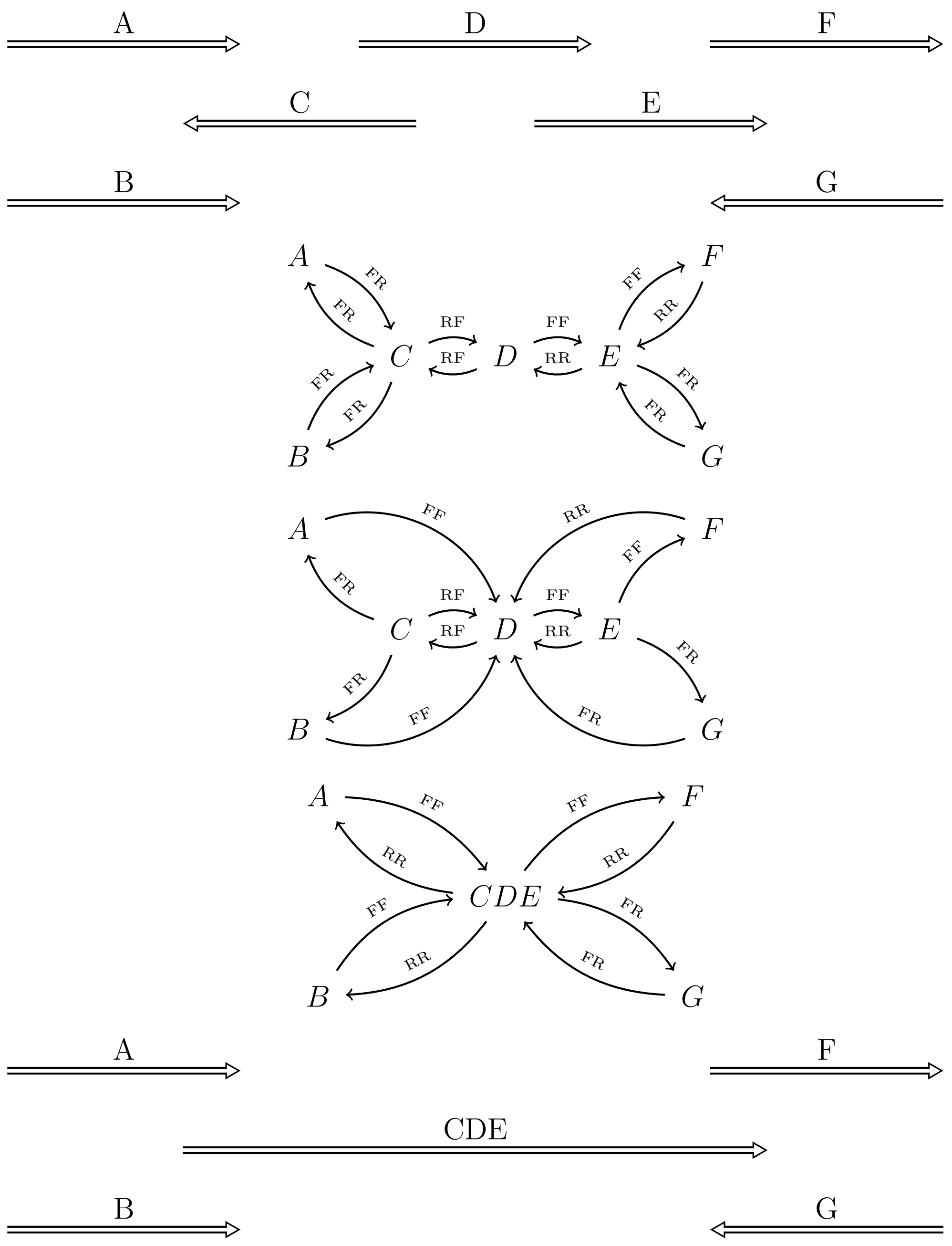


Figure 4.2: ***Compressing the graph***. **Top:** Simplified view of seven overlapping kmers. Nodes C, D, and E can be compressed into a single supernode since the pairs {C, D} and {D, E} have *degree* = 1 towards each other. **Middle-top:** Equivalent bidirected graph detailing edge types induced by kmer orientations. In the merge phase, D chooses "heads" and will subsume adjacent "tail" nodes C and E. **Center:** C and E update their neighbors to point to the subsuming node D. **Middle-bottom:** C and E merge towards D, and D uses their neighbor's edge type information to resolve its new edge types back towards the neighbors of C and E. The new kmer in CDE is now of length $len(\mathrm{C}) + len(\mathrm{D}) + len(\mathrm{E}) - (K - 1) \times 2$. To simplify graph updates, merged nodes maintain the key and orientation of the original "heads" nodes, in this case D. **Bottom:** Simplified view of the merged nodes.

4.2.6 Tip Remove

Sequencing errors that appear near the beginning or end of reads will cause “tips” or dead-ends in the de Bruijn graph. While an error introduced at the last position of a read will cause only one erroneous kmer (and a tip of length 1), an error introduced K bases from the end of the read will cause $2K - 1$ error kmers, with a tip length of $2K - 1$. Velvet originally suggested removing tips from the graph based on their length and coverage [135], and we do the same. Unbroken chains of length $2K - 1$ terminating in a dead-end with relatively low coverage (default: 5) will be excised from the graph, including all elements of the chain. This step usually happens after merging the graph, making tip identification very easy. However, removing tips from the graph can introduce additional tips, since multiple short tips may extend from the same branch. We therefore proceed with search for tips, originating at all dead ends in the graph, proceeding through all simple chain nodes, and stopping the search when all paths have been checked.

4.2.7 Bubble Merge

Sequencing errors and the presence of alternative alleles in the genome will cause “bubbles” in the graph. All $2K - 1$ kmers overlapping the single error will branch off at the last non-overlapping kmer and rejoin the main path after the last overlapping kmer. If multiple reads include adjacent errors, the main path may be littered with many non-overlapping bubbles. If a single read contains multiple sequencing errors within $K - 1$ bases of each other, the length of the bubble will be increased to the size of the union of the two component bubble paths before rejoining the main path. At high coverage, bubbles can be very common on the graph. For example, at 100x coverage and a 1% error rate, we expect every base in the genome to be covered by either a bubble or a tip. Still, the length of the individual bubbles should be relatively short since it is not common many errors to be present within

$K-1$ bases of each other in a single read. Unfortunately, the $2K-1$ erroneous kmers have some chance of matching a kmer that actually exists in the genome, causing a bridge to be formed between the erroneous bubble and some other part of the graph, which may represent sequence from some very distant portion of the genome.

To identify bubbles in the graph, Velvet implements a breadth-first search, starting at nodes with high coverage. The search proceeds until a node is visited twice. The twice-visited node must be the ending node of a bubble; Velvet proceeds to find the starting point of the bubble, extracts the sequence of the two alternative paths, and if the sequences are similar enough, removes the path with lower coverage. Nodes are processed from highest coverage to lowest coverage to give preference to higher-confidence operations, should two operations be mutually exclusive.

We follow a similar strategy, with the following changes: 1) we process all possible bubble paths simultaneously in parallel, 2)we don't give preference to high-coverage nodes, and 3) we only remove the connection of the bubble path to the main path rather than removing it from the graph. These changes are mostly in place to simplify the search process; we feel it is important for each operation to proceed in parallel. In our implementation, every node in the graph is considered a seed and starts a local search, seeking potential bubbles that **must** include the seed node's edges as the beginning of the bubble. If such a bubble is found, the seed nodes will be the last common nodes between two or more bubble paths. Rather than backtracking to get the sequence of traversed nodes, we store that information in the messages of the search itself and mark each node with the traversed paths. The extra information needed to store the paths within the nodes and messages will reduce the runtime of this algorithm if the sought paths become too long, since every path is explicitly stored and the number of paths is exponential in the tree width of the graph. In our testing, however, we found that very few searches become unmanageable since the search range is limited (default: 100bp). It may be necessary to impose an upper limit on the number of branches each seed is allowed to expand into, or to seek an alternate implementation that only stores the paths implicitly, as Velvet does.

4.2.8 Scaffolding

Our final algorithm for cleaning the de Bruijn graph consists of finding its high-confidence regions and attempting to expand them using the original reads. The methodology is similar to Ray [15], which proceeds as follows: First, a high-confidence version of the graph is built by removing nodes with low coverage (e.g., everything below 10x). The resulting graph will have very few sequencing errors but its components will be largely disconnected. Long, unbroken chains of kmers are identified and marked as high-confidence seeds. Next, the edges of the confident regions are extended through a voting process where, for each branch, the branch's kmer is checked for overlap with the reads present in the confident region. If one branch's kmer overlap *dominates* all other branches' kmers, the dominating branch is retained as part of the high-confidence region and the expansion voting continues with the chosen branch's next-neighbors. The process finishes when a dead end is reached or no dominating edge is found, resulting in a long walk through the graph. Finally, walks that overlap are checked to see if they should be merged together. Ray's voted walk performs admirably on the datasets we've tested.

We now present several improvements over the original Ray algorithm that should improve the accuracy of the generated walk. These improvements each come at the cost of a longer runtime, as they focus on searching in the neighborhood of the decision node. We use Figure 4.3 as a frame of reference to describe our improvements. Specifically, the nodes that were part of the original high-confidence region are considered "walk" nodes, the node whose branches are under consideration is termed the "frontier" node, and each of the branches under consideration are considered "candidates".

$$\begin{tikzcd}
 & & & & & & p^1_{13} \arrow[r] & \cdots \\
 & & & & c_1, p^1_{11}, p^1_{21} \arrow[r] & p^1_{12}, p^1_{22} \arrow[ur] \arrow[dr] & & \\
s \arrow[r] & w_1 \arrow[r] & w_2 \arrow[r] & f \arrow[ur] \arrow[dr] & & & p^1_{23} \arrow[r] & \cdots \\
 & & & & c_2, p^2_{11} \arrow[r] & p^2_{12} \arrow[r] & p^2_{13} \arrow[r] & \cdots
\end{tikzcd}$$

Figure 4.3: **Node designations for scaffolding.** The walk starts at a seed node, indicated by s, and has already proceeded through walk nodes w_1 through w_3. The frontier node f makes the decision about which node to include next in the walk from the candidates c_1 and c_2. Ray's algorithm makes the inclusion decision based only on this information. Our improvements include searching beyond the candidates into complete paths which may include additional branches. Over the search distance d, c_1 has split into two paths p^1_{1*} and p^1_{2*}, whereas candidate c_2 still has only one search path, p^2_{1*}. The decision made in f about whether to include c_1 or c_2 can be augmented by the combined kmer from each path as well as read information from the paths. We omit edge types and edges that don't participate in the algorithm (e.g., there could be other edges incoming to one of the path nodes).

Expanded Candidate Branches

The first improvement to scaffolding focuses on the kmer being compared. While Ray looks only at the next kmer in each branch (essentially, the single letters that proceed the frontier node), we expand the candidate kmers to a greater length. Since the graph has been compressed before scaffolding commences, most kmers are substantially longer than K. With only a single extra iteration, we can check not only the next letter but also a potentially large number of letters that proceed the single letter.

In Ray, determining dominance of a particular branch depends on three inequalities and a variable m, determined by the coverage of the frontier node. Given the set of reads from the high-confident walk, and in particular, the kmers from those reads that have the same *offset* as the candidate kmers, the read kmers are compared against each candidate kmer.

The dominant edge must have m-fold more total read kmers that exactly match the dominant edge's kmer, it must have an m-fold higher *offset-weighted* match score than all other branches, and the smallest non-zero walk node's contribution to the number of matching read kmers must be greater than m times the smallest non-zero walk node contribution. This is summarized in Equations 4.1, 4.2, and 4.3 as adapted from [15]. We say that a candidate y "wins" against y' if the following inequalities hold:

$$m \cdot \Sigma_{i=1}^{l}(l-i)\text{offset}_i(w,y') < \Sigma_{i=1}^{l}(l-i)\text{offset}_i(w,y) \tag{4.1}$$

$$m \cdot \Sigma_{i=1}^{l}\text{offset}_i(w,y') < \Sigma_{i=1}^{l}\text{offset}_i(w,y) \tag{4.2}$$

$$m \cdot \min_{i\in\{1...l\}}(\text{offset}_i(w,y') > 0) < \min_{i\in\{1...l\}}(\text{offset}_i(w,y') > 0) \tag{4.3}$$

where

$$\text{offset}_i(w,y) := \Sigma_{x\in w}\delta(\text{subseq}(x, i - \text{start}(x,w), K) = y) \tag{4.4}$$

that is, the total number of reads (x) in walk $w = \langle x_1 \dots x_J\rangle$ that exactly match candidate kmer y. The m factor is determined by the coverage of the frontier node and is smaller for higher coverage frontiers. We say that y *dominates* if and only if the inequality holds for all other candidates $y \in \text{edges}(f) \setminus \{y'\}$.

In a compressed graph, the candidate kmer y is of variable length, at least as long as K. In this case, we modify the offset function as follows:

$$\text{offset}_i^{\text{compressed}}(w, y) := \Sigma_{x \in w} \Sigma_{m \in \{0 \cdots M-K\}}$$
$$\delta(\text{subseq}(x, i - \text{start}(x, w) + m, K) = \text{subseq}(y, m, K)) \quad (4.5)$$

where $M = \min_{y \in Y^f}(\text{length}(y))$ and Y^f is the complete set of candidates for the current frontier f. This counts the total number of length-K subsequences in all reads that exactly match each length-K subsequence of the candidate kmer at the correct offset. Since the candidates may be of different lengths, we limit the the number of kmers compared to the shortest of the candidates.

In the compressed graph, we replace the offset_i function in Equations 4.1 and 4.2 with $\text{offset}_i^{\text{compressed}}$, however Equation 4.3 is a minimization over walk nodes, which have been compressed. We modify this equation to instead measure the number of distinct *reads* that overlap the candidate, that is:

$$\text{count}_i^{\text{compressed}}(w, y) := \Sigma_{x \in w}$$
$$\delta\left(\Sigma_{m \in \{0 \cdots M-K\}} \delta(\text{subseq}(x, i - \text{start}(x, w) + m, K) = \text{subseq}(y, m, K)) > 0\right) \quad (4.6)$$

In section 4.2.8, we show how to relax the minimum length constraint by expanding the candidate search to include multiple candidate paths.

Expanding Further and Scoring Multiple Candidate Paths

Candidate branch nodes contain kmer sequences that are of variable lengths, sometimes as short as K. When short, the important decision of including the branch in the confident walk is based on limited information, sometimes as little as a single base will determine which branch to include.

To overcome this shortcoming, we expand the candidate branches into candidate *paths*, enumerating all possible paths up to a predetermined distance d from the frontier and accumulating their complete sequence. Whereas previously $Y^f = \{y_i, i \in \text{edges}(f)\}$, we now enumerate the complete set of paths originating at the frontier f, passing through each candidate c as Y_c^f, each of which is normally at least d letters long (the exception occurs when a path encounters a dead end, in which case it will be shorter). Now, each candidate has a complete set of possible paths that pass through it. In this case, a candidate c *dominates* if and only if at least one of its paths dominates all paths in all other candidates' paths, that is, if and only if:

$$\exists\, p_c \in Y_c^f : \quad \forall\, c' \in \text{edges}(f) \setminus \{c\}, \;\; \forall\, p'_{c'} \in Y_{c'}^f \quad \text{dominates}(p_c, p'_{c'}) \tag{4.7}$$

Mutual Scoring

Another piece of information available in making the decision about which candidate to include in the path comes from reads that are present in the candidate paths themselves, which may be oriented back towards the walk itself. Up until this point, we have relied on the the reads in the walk nodes to score the candidate branches and/or paths. We can also allow the candidate branches themselves to score the walk. We define the final score for a candidate's path as the simple sum of the walk's offset score of the candidate path and the candidate's offset score of the walk.

By allowing the candidate branches to expand completely and by using mutual scoring from candidate paths, we effectively search the neighborhood surrounding the frontier decision point, and inform the candidate inclusion process in a way that, to the best of our knowledge, has not been attempted previously.

4.3 Results

In this section, we present preliminary results for the Genomix program in two main dimensions: first, how the algorithms scale to large genomes, across multiple machines, and in different available memory regimes. Second, we explore the accuracy of the algorithm in reconstructing several genomes. Our initial results are based on running all algorithms in single-end mode.

Timings

One compelling reason for using the Hyracks/Pregelix framework is its transparent and efficient use of main memory. Traditional assemblers must be able to fit their graph representation in RAM and are therefore only able to assemble large genomes on very large servers including terabytes of RAM. MPI-based methods such as Ray [15] and ABySS [106] have reduced some of the memory press by partitioning the graph across machines, but are still limited in the size of genome that can be assembled. To the best of our knowledge, only Contrail [100], which uses the Hadoop/MapReduce framework, is capable of scaling beyond the current assembly size limitations without breaking the bank for a large-memory system. Unfortunately, Hadoop uses almost no main memory, instead manipulating on-disk representations of the graph and in our testing, Contrail's runtime clearly suffered.

To test how our algorithm scales, we ran Genomix in three different configurations: first, on a single machine with limited memory (16GB), on a small cluster with limited memory (4 machines with 10 CPU's, 10GB RAM each), and on a small cluster with a substantial increase in available memory (4 machines with 10 CPU's, 100GB RAM each). We also explored the effect of varying the input data size by generating synthetic reads from human genome hg19 chromosome 14. Keeping the coverage fixed at 100x, we tested the runtimes of our algorithm's various stages. Figure 4.4 shows a near-linear increase in runtime for a linear increase in input data size, regardless of the computational configuration.

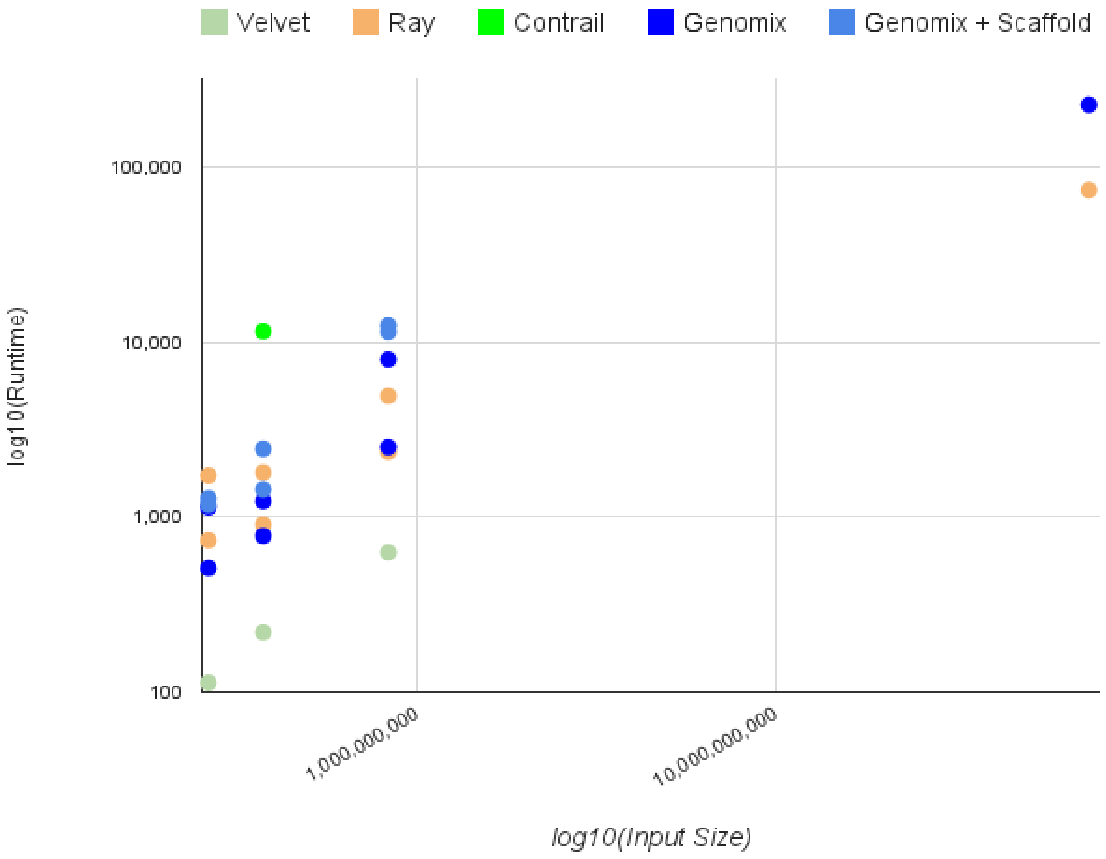


Figure 4.4: Runtime comparison as the size of the genome is increased while maintaining 100x coverage (synthetic data). These first three steps are take a substantial portion of the total runtime since all later steps operate on a greatly simplified graph. The scaffolding step may take much longer than these steps, depending on the search options used.

Assembly Accuracy

Assemblers seek to produce the most complete reconstruction of the genome, with the minimal number of gaps, the largest unbroken sequences, and the least number of errors. Per convention, we use the N_{50} metric to compare lengths and use the scripts released as part of GAGE to measure errors in the assembled contigs. The N_{50} is defined as the length of the shortest contig such that 50% of the original genome is in longer contigs.

	N_{50}	SNPs	Indels <5bp / $\geq$ 5bp	Inv./Rel/Trans.	Time 1/8/40 CPU
Rhodo (4.6MB genome, 101bp SE reads, 180x coverage)					
Velvet	4312	899	324 / 6	1 / 2 / 2	628 / F / NA
Ray	3800	295	118 /4	0 / 2 / 3	? / 2,347 / ?
Genomix	3878	573	138 / 4	2 / 3 / 9	? / 11,402 / ?
E. coli (4.6MB genome, 36bp SE reads, 80x coverage)					
Velvet	8735	24	0 / 0	0 / 2 / 0	220 / F / NA
Ray	12425	37	1 / 1	0 / 1 / 0	1,792 / 904 / ?
Genomix	10756	9	0 / 0	8 / 6 / 0	? / 2,452 / ?
Staph (2.9MB genome, 101bp SE reads, 90x coverage)					
Velvet	22361	100	9 / 4	0 / 2 / 0	113 / F / NA
Ray	3718	49	4 / 2	0 / 4 / 0	1,728 / 735 /?
Genomix	7881	77	3 / 1	3 / 3 / 0	1,276 / 1,176 /?

Table 4.1: Assembly results for three bacterial genomes using Velvet, Ray, and Genomix. In all cases, the algorithms were run with $K = 31$ in single-end mode and all metrics are as reported by GAGE. Abbreviations are Inv. = Inversions, Rel. = Relocations, Trans. = Translocations.

	N_{50}	SNPs	Indels <5bp / $\geq$ 5bp	Inv./Rel/Trans.	Time
Rhodo (4.6MB genome, 101bp SE reads, 180x coverage)					
Velvet	4312	899	324 / 6	1 / 2 / 2	628 (1 CPU)
Ray	3800	295	118 /4	0 / 2 / 3	2,347 (8 CPU)
Genomix	3878	573	138 / 4	2 / 3 / 9	11,402 (8 CPU)
E. coli (4.6MB genome, 36bp SE reads, 80x coverage)					
Velvet	8735	24	0 / 0	0 / 2 / 0	220 (1 CPU)
Ray	12425	37	1 / 1	0 / 1 / 0	904 (8 CPU)
Genomix	10756	9	0 / 0	8 / 6 / 0	2,452 (8 CPU)
Staph (2.9MB genome, 101bp SE reads, 90x coverage)					
Velvet	22361	100	9 / 4	0 / 2 / 0	113 (1 CPU)
Ray	3718	49	4 / 2	0 / 4 / 0	735 (8 CPU)
Genomix	7881	77	3 / 1	3 / 3 / 0	1,176 (8 CPU)

Table 4.2: Assembly results for three bacterial genomes using Velvet, Ray, and Genomix. In all cases, the algorithms were run with $K = 31$ in single-end mode and all metrics are as reported by GAGE. Abbreviations are Inv. = Inversions, Rel. = Relocations, Trans. = Translocations.

Though our initial results in each case are based on single-end performance, Genomix shows promise as a scalable platform for genome assembly and compares favorably to the other methods in terms of N_{50}, though its accuracy could be improved.

4.4 Discussion

We have implemented and tested a new, distributed algorithm for *de novo* genome assembly using approaches from several successful assemblers and have introduced several improvements to the existing methods. We have shown that although there is overhead associated with the distributed framework, the method scales well and should be able to handle much larger genomes than previously feasible.

Ray's voted walk through the graph is a very different approach than previous algorithms. Rather than seeking the walk through the graph that minimizes the total walk length while touching each edge (as in the Eulerian tour of Velvet and many other methods), Ray solves a minimum cost network flow problem locally, using the overlap information from the reads to guide the expanding candidates. This is similar to the minimum-cost network flow problem solved by [69], except that Medvedev sought to match global kmer coverage rather than using read/overlap information. There may be room for a combined approach which would solve the global flow problem while also incorporating local read overlap information., where the read information can guide local connections through the graph.

Our scaffolding implementation currently prunes the graph along the walk rather than incorporating the candidate nodes implicitly. It seems this premature edge pruning is leading to an increased number of large-scale errors (inversions, translocations, and relocations). We explored several strategies to delay graph pruning, but didn't fully explore an implicit inclusion and secondary merge phase as seen in Ray. Perhaps following their example more closely in the scaffolding stage would lead to reduced errors in the graph.

Our implementation of split-repeat handles only one of four possible repeat types enumerated in [69]. We started handling this repeat type fairly late in our implementation, but it had immediate positive effect on our accuracy and N50 metrics so it might be worthwhile to implement the other split methods.

Chapter 5

Conclusion

Through an astronomical increase in the quantity of available data, high-throughput sequencing has revolutionized basic and translational biology. As a result, biology has become even more empirically driven, moving from experimental procedures of limited scope about single genes to comprehensive, simultaneous sweeps elucidating function in genes across multiple cell types. Some basic assumptions are being challenged by the mountains of data, while novel hypotheses are being made, guided by and leading to new experimental procedures, many of which are firmly rooted in sequencing's dropping cost, increasing accuracy, and expanding throughput.

In this thesis, we have presented three enhancements to the state of the art in bioinformatic methods for genomic annotation, guided by high-throughput sequencing. First, we have described improvements in functional annotation of ChIP-seq data through an improved peak caller and a probabilistic model for the ChIP enrichment process, allowing ChIP peaks to be identified in previously-censored repeat regions of the genome. We demonstrated in two datasets that our method increases the total number of peaks called with transcription factor binding motifs without a major reduction in accuracy. We also showed that the peaks called tended to be near repeat regions of the genome and had been overlooked by previous peak calling methods. Second, we showed how classic probabilistic models can be adapted to specific experimental structures to better leverage mutual information between experiments. Specifically, we modified a hidden Markov model for learning histone mark-derived epigenetic state, creating a natural extension to the HMM that captured specific biological realities, in this case shared inheritance of epigenetic state across cell types. We also leveraged the optimization tools of variational inference to efficiently approximate the intractable model. Finally, we built a scalable genome assembler based on but improving the previous methods of Velvet and Ray, while able to efficiently assemble large genomes. The framework's capacity is limited by disk space distributed across many machines rather than available memory in a single computer. We show similar N_{50} to previous methods and promising scalability, while suffering somewhat in accuracy. Our implementations of these improvements are all available as open-source software.

Bibliography

[1] M. J. Aardema and J. T. MacGregor. Toxicology and genetic toxicology in the new era of “toxicogenomics”: impact of “-omics” technologies. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, 499(1):13–25, 2002.

[2] S. Altschul, B. Demchak, R. Durbin, R. Gentleman, M. Krzywinski, H. Li, A. Nekrutenko, J. Robinson, W. Rasband, J. Taylor, et al. The anatomy of successful computational biology software. *Nature biotechnology*, 31(10):894–897, 2013.

[3] T. Bailey and C. Elkan. The value of prior knowledge in discovering motifs with MEME. In *Proc Int Conf Intell Syst Mol Biol*, volume 3, pages 21–9, 1995.

[4] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble. Meme suite: tools for motif discovery and searching. *Nucleic acids research*, 37(suppl 2):W202–W208, 2009.

[5] M. J. Bamshad, S. B. Ng, A. W. Bigham, H. K. Tabor, M. J. Emond, D. A. Nickerson, and J. Shendure. Exome sequencing as a tool for mendelian disease gene discovery. *Nature Reviews Genetics*, 12(11):745–755, 2011.

[6] A. Bannister and T. Kouzarides. Regulation of chromatin by histone modifications. *Cell research*, 21(3):381–395, 2011.

[7] Y. Bergman and H. Cedar. Epigenetic control of recombination in the immune system. *Seminars in immunology*, 22(6):323–9, Dec. 2010.

[8] B. E. Bernstein, T. S. Mikkelsen, X. Xie, M. Kamal, D. J. Huebert, J. Cuff, B. Fry, A. Meissner, M. Wernig, K. Plath, R. Jaenisch, A. Wagschal, R. Feil, S. L. Schreiber, and E. S. Lander. A bivalent chromatin structure marks key developmental genes in embryonic stem cells. *Cell*, 125(2):315–26, Apr. 2006.

[9] M. Bibikova, E. Chudin, B. Wu, L. Zhou, E. Garcia, Y. Liu, S. Shin, T. Plaia, J. Auerbach, D. Arking, et al. Human embryonic stem cells have a unique epigenetic signature. *Genome research*, 16(9):1075–1083, 2006.

[10] J. Biesinger, Y. Wang, and X. Xie. Discovering and mapping chromatin states using a tree hidden markov model. *BMC bioinformatics*, 14(Suppl 5):S4, 2013.

[11] J. Bilmes and C. Bartels. On triangulating dynamic graphical models. In *Proceedings of the Nineteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 47–56, San Francisco, CA, 2003. Morgan Kaufmann.

[12] K. Blahnik, L. Dou, H. O'Geen, T. McPhillips, X. Xu, A. Cao, S. Iyengar, C. Nicolet, B. Ludascher, I. Korf, et al. Sole-Search: an integrated analysis program for peak detection and functional annotation using ChIP-seq data. *Nucleic Acids Research*, 38(3):e13, 2010.

[13] M. Blow, D. McCulley, Z. Li, T. Zhang, J. Akiyama, A. Holt, I. Plajzer-Frick, M. Shoukry, C. Wright, F. Chen, et al. ChIP-Seq identification of weakly conserved heart enhancers. *Nature genetics*, 42(9):806–810, 2010.

[14] V. Boeva, D. Surdez, N. Guillon, F. Tirode, A. Fejes, O. Delattre, and E. Barillot. De novo motif identification improves the accuracy of predicting transcription factor binding sites in ChIP-Seq data analysis. *Nucleic Acids Research*, 2010.

[15] S. Boisvert, F. Laviolette, and J. Corbeil. Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies. *Journal of Computational Biology*, 17(11):1519–1533, 2010.

[16] V. Borkar, M. Carey, R. Grover, N. Onose, and R. Vernica. Hyracks: A flexible and extensible foundation for data-intensive computing. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 1151–1162. IEEE, 2011.

[17] S. D. Boyd. Diagnostic applications of high-throughput dna sequencing. *Annual Review of Pathology: Mechanisms of Disease*, 8:381–410, 2013.

[18] Y. Bu. Pregelix: dataflow-based big graph analytics. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 54. ACM, 2013.

[19] J. Butler, I. MacCallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum, and D. B. Jaffe. Allpaths: de novo assembly of whole-genome shotgun microreads. *Genome research*, 18(5):810–820, 2008.

[20] M. J. Chaisson, D. Brinza, and P. A. Pevzner. De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome research*, 19(2):336–346, 2009.

[21] N. Chuzhanova, S. Abeysinghe, M. Krawczak, and D. Cooper. Translocation and gross deletion breakpoints in human inherited disease and cancer II: Potential involvement of repetitive sequence elements in secondary structure formation between DNA ends. *Human mutation*, 22(3):245–251, 2003.

[22] A. J. Cox. Efficient Large-Scale Alignment of Nucleotide Databases. Whole genome alignments to a reference genome. *http://bioinfo.cgrb.oregonstate.edu/docs/solexa*, 2007.

[23] A. Darwiche. *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009.

[24] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[25] T. Dean and K. Kanazawa. Probabilistic temporal reasoning. AAAI, 1988.

[26] X. Dong, M. Greven, A. Kundaje, S. Djebali, J. Brown, C. Cheng, T. Gingeras, M. Gerstein, R. Guigó, E. Birney, et al. Modeling gene expression using chromatin features in various cellular contexts. *Genome biology*, 13(9):R53, 2012.

[27] R. Durbin. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge Univ Pr, 1998.

[28] J. Ernst and M. Kellis. Discovery and characterization of chromatin states for systematic annotation of the human genome. *Nature biotechnology*, 28(8):817–25, Aug. 2010.

[29] J. Ernst, P. Kheradpour, T. S. Mikkelsen, N. Shoresh, L. D. Ward, C. B. Epstein, X. Zhang, L. Wang, R. Issner, M. Coyne, M. Ku, T. Durham, M. Kellis, and B. E. Bernstein. Mapping and analysis of chromatin state dynamics in nine human cell types. *Nature*, 473(7345):43–9, May 2011.

[30] A. Fejes, G. Robertson, M. Bilenky, R. Varhol, M. Bainbridge, and S. Jones. FindPeaks 3. 1: a tool for identifying areas of enrichment from massively parallel short-read sequencing technology. *Bioinformatics*, 24(15):1729, 2008.

[31] G. Felsenfeld and M. Groudine. Controlling the double helix. *Nature*, 421(6921):448–453, 2003.

[32] N. C. for Biotechnology Information. Ncbi to discontinue sequence read archive and peptidome [online]. 2011. URL: `http://www.ncbi.nlm.nih.gov/About/news/16feb2011`.

[33] N. C. for Biotechnology Information. Status of the ncbi sequence read archive (sra) [online]. 2011. URL: `http://www.ncbi.nlm.nih.gov/About/news/13Oct2011.html`.

[34] L. A. Garraway and E. S. Lander. Lessons from the cancer genome. *Cell*, 153(1):17 – 37, 2013.

[35] J. B. German, B. D. Hammock, and S. M. Watkins. Metabolomics: building on a century of biochemistry to guide human health. *Metabolomics*, 1(1):3–9, 2005.

[36] P. Glaus, A. Honkela, and M. Rattray. Identifying differentially expressed transcripts from rna-seq data with biological variation. *Bioinformatics*, 28(13):1721–1728, 2012.

[37] S. Gnerre, I. MacCallum, D. Przybylski, F. J. Ribeiro, J. N. Burton, B. J. Walker, T. Sharpe, G. Hall, T. P. Shea, S. Sykes, et al. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, 108(4):1513–1518, 2011.

[38] J. Gomez-Ramirez and R. Sanz. On the limitations of standard statistical modeling in biological systems: a full bayesian approach for biology. *Progress in biophysics and molecular biology*, 113(1):80–91, 2013.

[39] R. Hagen, S. Rodriguez-Cuenca, and A. Vidal-Puig. An allostatic control of membrane lipid composition by SREBP1. *FEBS letters*, 2010.

[40] N. Heintzman, R. Stuart, G. Hon, Y. Fu, C. Ching, R. Hawkins, L. Barrera, S. Van Calcar, C. Qu, K. Ching, et al. Distinct and predictive chromatin signatures of transcriptional promoters and enhancers in the human genome. *Nature genetics*, 39(3):311–318, 2007.

[41] M. Hoffman, O. Buske, J. Wang, Z. Weng, J. Bilmes, and W. Noble. Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nature methods*, 9(5):473–476, 2012.

[42] G. Hon, W. Wang, and B. Ren. Discovery and annotation of functional chromatin signatures in the human genome. *PLoS computational biology*, 5(11):e1000566, 2009.

[43] X. Huang and A. Madan. Cap3: A dna sequence assembly program. *Genome research*, 9(9):868–877, 1999.

[44] A. Huda and I. Jordan. Epigenetic regulation of Mammalian genomes by transposable elements. *Annals of the New York Academy of Sciences*, 1178(Natural Genetic Engineering and Natural Genome Editing):276–284, 2009.

[45] E. Jablonka and G. Raz. Transgenerational epigenetic inheritance: prevalence, mechanisms, and implications for the study of heredity and evolution. *The Quarterly review of biology*, 84(2):131–176, 2009.

[46] R. Jaschek and A. Tanay. Spatial clustering of multivariate genomic and epigenomic information. In *Research in Computational Molecular Biology*, pages 170–183. Springer, 2009.

[47] H. Ji, H. Jiang, W. Ma, D. Johnson, R. Myers, and W. Wong. An integrated software system for analyzing ChIP-chip and ChIP-seq data. *Nature biotechnology*, 26(11):1293–1300, 2008.

[48] H.-X. Ju, B. An, Y. Okamoto, K. Shinjo, Y. Kanemitsu, K. Komori, T. Hirai, Y. Shimizu, T. Sano, A. Sawaki, M. Tajika, K. Yamao, M. Fujii, H. Murakami, H. Osada, H. Ito, I. Takeuchi, Y. Sekido, and Y. Kondo. Distinct profiles of epigenetic evolution between colorectal cancers with and without metastasis. *The American journal of pathology*, 178(4):1835–46, Apr. 2011.

[49] W. KA. DNA Sequencing Costs data from the nhgri genome sequencing program (gsp) [online]. 2014. URL: `http://www.genome.gov/sequencingcosts/`.

[50] M. Kagey, J. Newman, S. Bilodeau, Y. Zhan, D. Orlando, N. van Berkum, C. Ebmeier, J. Goossens, P. Rahl, S. Levine, et al. Mediator and cohesin connect gene expression and chromatin architecture. *Nature*, 2010.

[51] M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kino****a, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in kegg. *Nucleic acids research*, 34(suppl 1):D354–D357, 2006.

[52] P. Kharchenko, M. Tolstorukov, and P. Park. Design and analysis of ChIP-seq experiments for DNA-binding proteins. *Nature biotechnology*, 26(12):1351–1359, 2008.

[53] J. Khatun. An integrated encyclopedia of dna elements in the human genome. *Nature*, 489:57–74, 2012.

[54] T. Kim, Z. Abdullaev, A. Smith, K. Ching, D. Loukinov, R. Green, M. Zhang, V. Lobanenkov, and B. Ren. Analysis of the vertebrate insulator protein ctcf-binding sites in the human genome. *Cell*, 128(6):1231–1245, 2007.

[55] M. Kircher and J. Kelso. High-throughput dna sequencing–concepts and limitations. *Bioessays*, 32(6):524–536, 2010.

[56] P. Kolasinska-Zwierz, T. Down, I. Latorre, T. Liu, X. Liu, and J. Ahringer. Differential chromatin marking of introns and expressed exons by h3k36me3. *Nature genetics*, 41(3):376–381, 2009.

[57] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, K. Dewar, M. Doyle, W. FitzHugh, et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001.

[58] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[59] H. Li and R. Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[60] H. Li, J. Ruan, and R. Durbin. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome research*, 18(11):1851, 2008.

[61] R. Li, Y. Li, K. Kristiansen, and J. Wang. SOAP: short oligonucleotide alignment program. *Bioinformatics*, 24(5):713, 2008.

[62] R. Lister, M. Pelizzola, R. Dowen, R. Hawkins, G. Hon, J. Tonti-Filippini, J. Nery, L. Lee, Z. Ye, Q. Ngo, et al. Human dna methylomes at base resolution show widespread epigenomic differences. *Nature*, 462(7271):315–322, 2009.

[63] J. Liu, Z. Zhang, M. Bando, T. Itoh, M. Deardorff, D. Clark, M. Kaur, S. Tandy, T. Kondoh, E. Rappaport, et al. Transcriptional dysregulation in NIPBL and cohesin mutant human cells. *PLoS Biol*, 7(5):e1000119, 2009.

[64] N. J. Loman, R. V. Misra, T. J. Dallman, C. Constantinidou, S. E. Gharbia, J. Wain, and M. J. Pallen. Performance comparison of benchtop high-throughput sequencing platforms. *Nature biotechnology*, 30(5):434–439, 2012.

[65] R. Luo, B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu, et al. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience*, 1(1):18, 2012.

[66] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

[67] F. Markowetz and R. Spang. Inferring cellular networks–a review. *BMC bioinformatics*, 8(Suppl 6):S5, 2007.

[68] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, et al. The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303, 2010.

[69] P. Medvedev and M. Brudno. Maximum likelihood genome assembly. *Journal of computational Biology*, 16(8):1101–1116, 2009.

[70] P. Medvedev, K. Georgiou, G. Myers, and M. Brudno. Computability of models for sequence assembly. In *Algorithms in Bioinformatics*, pages 289–301. Springer, 2007.

[71] M. Might. Hunting down my son's killer [online]. 2012. URL: `http://matt.might.net/articles/my-sons-killer/`.

[72] T. Mikkelsen, M. Ku, D. Jaffe, B. Issac, E. Lieberman, G. Giannoukos, P. Alvarez, W. Brockman, T. Kim, R. Koche, et al. Genome-wide maps of chromatin state in pluripotent and lineage-committed cells. *Nature*, 448(7153):553–560, 2007.

[73] I. Mitchell Guttman, M. Garber, C. French, M. Lin, D. Feldser, M. Huarte, O. Zuk, B. Carey, J. Cassady, M. Cabili, et al. Chromatin signature reveals over a thousand highly conserved large non-coding rnas in mammals. *Nature*, 458(7235):223–227, 2009.

[74] G. E. Moore et al. Cramming more components onto integrated circuits [online]. 1965.

[75] O. Morozova and M. A. Marra. Applications of next-generation sequencing technologies in functional genomics. *Genomics*, 92(5):255–264, 2008.

[76] A. Mortazavi, B. Williams, K. McCue, L. Schaeffer, and B. Wold. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature methods*, 5(7):621–628, 2008.

[77] K. Murphy et al. The bayes net toolbox for matlab. *Computing science and statistics*, 33(2):1024–1034, 2001.

[78] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

[79] E. W. Myers. The fragment assembly string graph. *Bioinformatics*, 21(suppl 2):ii79–ii85, 2005.

[80] R. Nativio, K. Wendt, Y. Ito, J. Huddleston, S. Uribe-Lewis, K. Woodfine, C. Krueger, W. Reik, J. Peters, and A. Murrell. Cohesin is required for higher-order chromatin conformation at the imprinted IGF2-H19 locus. 2009.

[81] A. C. Need, V. Shashi, Y. Hitomi, K. Schoch, K. V. Shianna, M. T. McDonald, M. H. Meisler, and D. B. Goldstein. Clinical application of exome sequencing in undiagnosed genetic conditions. *Journal of medical genetics*, 49(6):353–361, 2012.

[82] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[83] D. Newkirk, J. Biesinger, A. Chon, K. Yokomori, and X. Xie. Arem: aligning short reads from chip-sequencing by expectation maximization. *Journal of Computational Biology*, 18(11):1495–1505, 2011.

[84] E. of DNA Elements (ENCODE). Encyclopedia of dna elements (encode) [online]. 2012. URL: `http://genome.ucsc.edu/ENCODE/`.

[85] Z. Ouyang, Q. Zhou, and W. Wong. ChIP-Seq of transcription factors predicts absolute and differential gene expression in embryonic stem cells. *Proceedings of the National Academy of Sciences*, 106(51):21521, 2009.

[86] C. A. Ouzounis. Rise and demise of bioinformatics? promise and progress. *PLoS computational biology*, 8(4):e1002487, 2012.

[87] D. P. Outpaced by innovation: Canceling an xprize [online]. 2013. URL: `http://www.huffingtonpost.com/peter-diamandis/outpaced-by-innovation-ca_b_3795710.html`.

[88] P. Park. ChIP–seq: advantages and challenges of a maturing technology. *Nature Reviews Genetics*, 10(10):669–680, 2009.

[89] V. R. Patel, K. Eckel-Mahan, P. Sassone-Corsi, and P. Baldi. Circadiomics: integrating circadian genomics, transcriptomics, proteomics and metabolomics. *Nature methods*, 9(8):772–773, 2012.

[90] S. Pepke, B. Wold, and A. Mortazavi. Computation for ChIP-seq and RNA-seq studies. *Nature Methods*, 6:S22–S32, 2009.

[91] S. Pepke, B. Wold, and A. Mortazavi. Computation for ChIP-seq and RNA-seq studies. *Nature methods*, 6(11 Suppl):S22–32, Nov. 2009.

[92] P. A. Pevzner, H. Tang, and M. S. Waterman. An eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

[93] Z. Qin, J. Yu, J. Shen, C. Maher, M. Hu, S. Kalyana-Sundaram, J. Yu, and A. Chinnaiyan. HPeak: an HMM-based algorithm for defining read-enriched regions in ChIP-Seq data. *BMC bioinformatics*, 11(1):369, 2010.

[94] C. Redon, D. Pilch, E. Rogakou, O. Sedelnikova, K. Newrock, and W. Bonner. Histone h2a variants h2ax and h2az. *Current opinion in genetics & development*, 12(2):162–169, 2002.

[95] B. Rhead, D. Karolchik, R. Kuhn, A. Hinrichs, A. Zweig, P. Fujita, M. Diekhans, K. Smith, K. Rosenbloom, B. Raney, et al. The UCSC genome browser database: update 2010. *Nucleic acids research*, 2009.

[96] M. D. Robinson, D. J. McCarthy, and G. K. Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2010.

[97] S. Roychowdhury, M. K. Iyer, D. R. Robinson, R. J. Lonigro, Y.-M. Wu, X. Cao, S. Kalyana-Sundaram, L. Sam, O. A. Balbin, M. J. Quist, et al. Personalized oncology through integrative high-throughput sequencing: a pilot study. *Science translational medicine*, 3(111):111–121, 2011.

[98] E. Rubio, D. Reiss, P. Welcsh, C. Disteche, G. Filippova, N. Baliga, R. Aebersold, J. Ranish, and A. Krumm. CTCF physically links cohesin to chromatin. *Proceedings of the National Academy of Sciences*, 105(24):8309, 2008.

[99] M. Salmon-Divon, H. Dvinge, K. Tammoja, and P. Bertone. PeakAnalyzer: Genome-wide annotation of chromatin binding and modification loci. *BMC bioinformatics*, 11(1):415, 2010.

[100] M. Schatz, D. Sommer, D. Kelley, and M. Pop. Contrail: Assembly of large genomes using cloud computing. In *CSHL Biology of Genomes Conference*, 2010.

[101] C. Schmid and P. Bucher. MER41 Repeat Sequences Contain Inducible STAT1 Binding Sites. *PloS one*, 5(7):e11425, 2010.

[102] S. L. Schreiber and B. E. Bernstein. Signaling network model of chromatin. *Cell*, 111(6):771–8, Dec. 2002.

[103] Y. Seo, H. Chong, A. Infante, S. Im, X. Xie, and T. Osborne. Genome-wide analysis of SREBP-1 binding in mouse liver chromatin reveals a preference for promoter proximal binding to a new motif. *Proceedings of the National Academy of Sciences*, 106(33):13765, 2009.

[104] J. Shendure and H. Ji. Next-generation dna sequencing. *Nature biotechnology*, 26(10):1135–1145, 2008.

[105] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[106] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, and İ. Birol. Abyss: a parallel assembler for short read sequence data. *Genome research*, 19(6):1117–1123, 2009.

[107] S. S. Sindi, S. Onal, L. Peng, H.-T. Wu, and B. J. Raphael. An integrative probabilistic model for identification of structural variation in sequencing data. *Genome Biol*, 13(3):R22, 2012.

[108] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[109] C. Spyrou, R. Stark, A. Lynch, and S. Tavaré. BayesPeak: Bayesian analysis of ChIP-seq data. *BMC bioinformatics*, 10(1):299, 2009.

[110] B. D. Strahl and C. D. Allis. The language of covalent histone modifications. *Nature*, 403(6765):41–5, Jan. 2000.

[111] G. G. Sutton, O. White, M. D. Adams, and A. R. Kerlavage. Tigr assembler: A new tool for assembling large shotgun sequencing projects. *Genome Science and Technology*, 1(1):9–19, 1995.

[112] G. Thijs, M. Lescot, K. Marchal, S. Rombauts, B. De Moor, P. Rouze, and Y. Moreau. A higher-order background model improves the detection of promoter regulatory elements by gibbs sampling. *Bioinformatics*, 17(12):1113–1122, 2001.

[113] B. Tran, A. M. Brown, P. L. Bedard, E. Winquist, G. D. Goss, S. J. Hotte, S. A. Welch, H. W. Hirte, T. Zhang, L. D. Stein, et al. Feasibility of real time next generation sequencing of cancer genes linked to drug response: results from a clinical trial. *International Journal of Cancer*, 132(7):1547–1555, 2013.

[114] C. Trapnell, A. Roberts, L. Goff, G. Pertea, D. Kim, D. R. Kelley, H. Pimentel, S. L. Salzberg, J. L. Rinn, and L. Pachter. Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature protocols*, 7(3):562–578, 2012.

[115] E. Tuzun, A. J. Sharp, J. A. Bailey, R. Kaul, V. A. Morrison, L. M. Pertz, E. Haugen, H. Hayden, D. Albertson, D. Pinkel, et al. Fine-scale structural variation of the human genome. *Nature genetics*, 37(7):727–732, 2005.

[116] D. Ucar, Q. Hu, and K. Tan. Combinatorial chromatin modification patterns in the human genome revealed by subspace clustering. *Nucleic acids research*, 39(10):4063–75, May 2011.

[117] U. Vishkin. Randomized speed-ups in parallel computation. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 230–239. ACM, 1984.

[118] N. Wagle, M. F. Berger, M. J. Davis, B. Blumenstiel, M. DeFelice, P. Pochanard, M. Ducar, P. Van Hummelen, L. E. MacConaill, W. C. Hahn, et al. High-throughput detection of actionable genomic alterations in clinical tumor samples by targeted, massively parallel sequencing. *Cancer discovery*, 2(1):82–93, 2012.

[119] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[120] J. Wang, W. Wang, R. Li, Y. Li, G. Tian, L. Goodman, W. Fan, J. Zhang, J. Li, J. Zhang, et al. The diploid genome sequence of an asian individual. *Nature*, 456(7218):60–65, 2008.

[121] K. Wang, M. Li, D. Hadley, R. Liu, J. Glessner, S. F. Grant, H. Hakonarson, and M. Bucan. Penncnv: an integrated hidden markov model designed for high-resolution copy number variation detection in whole-genome snp genotyping data. *Genome research*, 17(11):1665–1674, 2007.

[122] K. Wang, M. Li, and H. Hakonarson. Annovar: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic acids research*, 38(16):e164–e164, 2010.

[123] Z. Wang, M. Gerstein, and M. Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, 2009.

[124] K. Wendt, K. Yoshida, T. Itoh, M. Bando, B. Koch, E. Schirghuber, S. Tsutsumi, G. Nagae, K. Ishihara, T. Mishiro, et al. Cohesin mediates transcriptional insulation by CCCTC-binding factor. *Nature*, 451(7180):796–801, 2008.

[125] T. White. *Hadoop: The Definitive Guide: The Definitive Guide*. O'Reilly Media, 2009.

[126] E. Wilbanks and M. Facciotti. Evaluation of Algorithm Performance in ChIP-Seq Peak Detection. *PloS one*, 5(7):e11471, 2010.

[127] M. R. Wilkins, R. D. Appel, J. E. Van Eyk, M. Chung, A. Görg, M. Hecker, L. A. Huber, H. Langen, A. J. Link, Y.-K. Paik, et al. Guidelines for the next 10 years of proteomics. *Proteomics*, 6(1):4–8, 2006.

[128] H. Xu, C. Wei, F. Lin, and W. Sung. An hmm approach to genome-wide identification of differential histone modification sites from chip-seq data. *Bioinformatics*, 24(20):2344–2349, 2008.

[129] X. Xu, S. Hoang, M. W. Mayo, and S. Bekiranov. Application of machine learning methods to histone methylation ChIP-Seq data reveals H4R3me2 globally represses gene expression. *BMC bioinformatics*, 11:396, Jan. 2010.

[130] E. Yaffe and A. Tanay. Probabilistic modeling of hi-c contact maps eliminates systematic biases to characterize global chromosomal architecture. *Nature genetics*, 43(11):1059–1065, 2011.

[131] J. Yedidia, W. Freeman, and Y. Weiss. Understanding belief propagation and its generalizations. In *Exploring artificial intelligence in the new millennium*, volume 8, pages 236–239. 2003.

[132] C. Yokoyama, X. Wang, M. Briggs, A. Admon, J. Wu, X. Hua, J. Goldstein, and M. Brown. SREBP-1, a basic-helix-loop-helix-leucine zipper protein that controls transcription of the low density lipoprotein receptor gene. *Cell*, 75(1):187–197, 1993.

[133] C. Zang, D. Schones, C. Zeng, K. Cui, K. Zhao, and W. Peng. A clustering approach for identification of enriched domains from histone modification ChIP-Seq data. *Bioinformatics*, 25(15):1952, 2009.

[134] W. Zeng, J. C. de Greef, Y.-Y. Chen, R. Chien, X. Kong, H. C. Gregson, S. T. Winokur, A. Pyle, K. D. Robertson, J. A. Schmiesing, V. E. Kimonis, J. Balog, R. R. Frants, A. R. Ball, Jr., L. F. Lock, P. J. Donovan, S. M. van der Maarel, and K. Yokomori. Specific loss of histone h3 lysine 9 trimethylation and hp1γ/cohesin binding at d4z4 repeats is associated with facioscapulohumeral dystrophy (fshd). *PLoS Genet*, 5(7):e1000559, 07 2009.

[135] D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[136] X. Zhang, G. Robertson, M. Krzywinski, K. Ning, A. Droit, S. Jones, and R. Gottardo. Pics: Probabilistic inference for chip-seq. *Biometrics*, 67(1):151–163, 2011.

[137] Y. Zhang, T. Liu, C. Meyer, J. Eeckhoute, D. Johnson, B. Bernstein, C. Nussbaum, R. Myers, M. Brown, W. Li, et al. Model-based analysis of ChIP-Seq (MACS). *Genome biology*, 9(9):R137, 2008.

[138] X. Zhou and A. Rokas. Prevention, diagnosis and treatment of high-throughput sequencing data pathologies. *Molecular Ecology*, 23(7):1679–1700, 2014.