Dissecting the Spatial Structure of Overlapping Transcription in Budding Yeast

by

Timothy Danford

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
October 19, 2009

Certified by .... [illegible] ....
David K. Gifford
Professor
Thesis Supervisor

Accepted by .... [illegible] ....
Terry P. Orlando
Chairman, Department Committee on Graduate Theses

Dissecting the Spatial Structure of Overlapping Transcription in Budding Yeast

by

Timothy Danford

Submitted to the Department of Electrical Engineering and Computer Science
on October 19, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis presents a computational and algorithmic method for the analysis of high-resolution transcription data in the budding yeast *Saccharomyces cerevisiae*. We begin by describing a computational system for storing and retrieving spatially-mapped genomic data. This system forms the infrastructure for a novel algorithmic approach to detect and recover instances of same-strand overlapping transcripts in high resolution expression experiments. We then apply these algorithms to a set of transcription experiments in budding yeast, *Saccharomyces cerevisiae*, in order to identify potential sites of same-strand overlapping transcripts that may be involved in novel forms of transcriptional regulation.

Thesis Supervisor: David K. Gifford
Title: Professor

Acknowledgments

This thesis would never have been completed without the help, both personal and academic, that I received from a large number of people. Bruce Donald mentored me as an undergraduate, directed me to graduate school, and remains the smartest man I've ever met. At MIT, Ernest Frankel, Ben Gordon, Rick Young, Gerry Fink, and Tommi Jaakkola provided advice, mentorship, and careful instruction. I was lucky to have wonderful collaborators in Duncan Odom, Divya Mathur, and Sudeep Agarwala. There were also administrators without whom I never would have survived even a single semester: Jeanne Darling, Marilyn Pierce, and Janet Fischer. MIT was where I met a large number of my friends and lab-mates, including John Barnett, Shaun Mahony, Georg Gerber, Reina Riemann, Bob Altshuler, Chris Reeder, Adam Marcus, Jason Rennie, and Ted Benson. Finally, my non-MIT friends were just as instrumental in helping me survive and graduate: John Snavely, Jackson Childs, Jolene Pinder, Michelle Beaulieu, Christa Bosch, and Jason Elliott. Ale Checka and Laura Stuart deserve their own, special, acknowledgement.

Alex Rolfe has been my officemate for five years, and I've thanked my luck for each one. He's been a good friend, a close collaborator, someone to whom I can go for good and trustworthy advice, and a skilled debugger of LaTeX – in short, the best officemate a computer science student could have hoped for. Along with Alex, Robin Dowell has been a friend, collaborator, and mentor throughout the last several years of school. And, of course, David Gifford has been my advisor throughout my graduate career. He has shown endless patience, and unwavering support, over my eight years at M.I.T. The care he takes in his research, and the advice he gives to his students, speaks to his thoughtfulness and vision.

My parents, Steve and Linda, managed to sit through eight years of watching their oldest son stumble through graduate school without ever muttering a critical word within my earshot. They supported me financially when I needed them, provided an endless stream of encouragement and help the rest of the time, and were the best co-bloggers a son could hope for. My father taught me to program and was my original role model as a scientist; and making my mother laugh still makes me incredibly happy. I hope this thesis makes them proud.

Finally, there's Rachel: who supported me unconditionally through an expensive and emotionally draining experience that (at times) must have seemed as if it would never end, without whose constant encouragement and help I never would have finished, and who I love endlessly and dearly. This thesis is dedicated to her.

Contents

List of Figures

List of Tables

Chapter 1

Represention and Analysis of High-Resolution Genomic Data

The experimental design and analysis of genomic data has, in the past, often been based on gene annotations or another discrete set of units. Microarray expression experiments' results are typically summarized by gene-specific intensities or values. ChIP-Chip transcription factor binding experiments are designed to probe well-defined promoter regions corresponding to individual genes, and their analysis is carried in terms of those genes. These discrete analyses were influenced by the coverage limitations of early microarray technologies, which had probe densities roughly equivalent to the number of genes in bacteria or yeast. Later array technologies provided genomic coverage sufficient to cover smaller genomes, but only had enough probes to tile around the start-sites of gene annotations in higher eukaryotes (human and mouse).

Biological analysis of these experiments begins with gene-based genomic data, then combines those results with other structured data sources such as protein interaction networks[28] or transcriptional regulatory networks[25]. Often these analyses look for paths, cliques, or other graph-theoretic patterns that are reflected in the original experimental dataset[9]. Even simpler analyses, based on matching experimental results against large numbers of predefined gene sets or other annotations can be used as a large-scale screening mechanism[57]. The concept of *enrichment*, the over-representation of discrete units (such as genes) defined by a genomic dataset within a pre-defined collection of sets, is a central concept to much of systems-biological analysis[43].

These kinds of analyses, in terms of separate and independent units, still only give us an incomplete picture of the genome's biology. Chromatin structure, or chemical marks applied to histones or the DNA itself, add a second layer of information that cannot be completely inferred from the chromosomal sequence itself[36]. Genes have diverse regulatory structures, often with multiple promoters and enhancers that are located kilobases away from the annotated start of the gene's coding sequence. Gene transcripts may be spliced into different coding messages or may have alternate transcriptional forms. More recently, systematic studies have begun to show that "transcription" is not limited only to the coding regions of the genome, but widely occurs (and is prevalent, in some genomes) in the intergenic regions. These so-called "noncoding RNAs" may be functional in their own right or may carry a function through the act of their production[58].

To take a step beyond simple genes and gene-sets requires spatial analysis methods. Spatial models and algorithms treat the genome as an experimental landscape, and only attempt to tie experimental results and observations back to genes or other annotations during the analysis. A spatial approach to genomic analysis identifies transcription factor binding, chromatin state or histone marks, and transcriptional activity, based on their genomic location which is determined without bias from nearby sequence annotations.

High-density microarrays and high-throughput sequencing are allowing researchers to finally measure events at high spatial resolution and across a genome-wide scale. Microarrays with probe spacings every 50 nucleotides, or even less than the width of the array's probe itself (so called *tiling* microarrays), are able to provide measurements across an entire chromosome from which the fine structure of biological events can be deconvolved. High-throughput sequencing technologies sample DNA fragments from an experimental population that are sequenced and mapped back to the target genome. A deconvolution process can be used to decode the locations of specific events and biological processes along the genome from the mapped results.

Hand in hand with these novel experimental methods, researchers have created computational and statistical frameworks for their spatial analysis. Deconvolution is a popular method for finding the precise locations of biological events which we expect to have a point-like existence along the genome [50, 66]. The binding locations of transcription factors are often modeled as occurring at a single base-pair (or regions of only a few base-pairs in length). Other biological processes are expected to produce regions of activity; the interpretation of these experimental datasets often requires a statistical "segmentation" process, in which the active regions are identified and separated from the inactive or background regions. Examples of experiments which can be interpreted in this way include CGH datasets, histone density and chromatin accessibility, or high-resolution transcription datasets. These analysis methods can be used for either microarray or sequencing-based experimental results.

Segmentation-based spatial analysis methods will overlook *overlapping* events. Some biological phenomena cannot be understood solely as individual, independent, and non-overlapping regions. For example, transcripts may overlap due to the time- and population-averaged nature of microarray experiments and cell populations. If two sub-populations of cells within a biological sample are producing different transcripts which cover genomic regions that are not completely disjoint, then these transcripts will appear on a microarray (or other experimental measurement) as overlapping events. Other dynamic genomic phenomena, which could be present at different locations in different cell populations, will present similar overlapping experimental signatures. Analysis of these experimental datasets should attempt to recover these coherent but overlapping regions when at all possible; otherwise, the experimenter may be led to believe that three transcripts are present instead of (for example) two, or be mistaken in the relative locations and intensities of the transcripts that are present.

In this thesis, we describe how to adapt a novel method for the analysis of tiling microarray transcription data by applying a computational to overlapping genomic regions: additive transcript re-assembly based on genomic segmentation. We also demonstrate the use of the software system and the new tiling microarray segmentation and analysis framework to model a new, systematic phenomenon observable in high-resolution tiling microarray data: the presence of *overlapping transcripts* in budding yeast. We conclude by demonstrating how this new computational understanding of a genomic dataset allows us to extend our understanding of transcription and regulation in this same organism.

1.1 The Spatial Character of Modern Genomics Data

Classical biology understands genomic data in terms of its component pieces – genes and proteins – and standard biological relations between them. Undirected graphs are used to model the physical interactions between proteins. Metabolic reactions that take place within the cell can be modeled as a directed, bipartite graph between reactant nodes and reaction nodes, where the directionality of the edges reflects the input/output relationships of reactants and reactions[14]. Genes are associated with reaction nodes if their corresponding protein product catalyzes that reaction. Genes are also arranged into regulatory networks, directed graph representations connecting two genes if the product of the first gene regulates the transcription of the second gene. Some functional genetics tests, such as experiments that measure genes that are lethal when deleted in pairs or genes whose expression rescues the effects of the other deletions, are represented as networks or sets-of-sets[22]. Quantitative experimental measurements of genes or proteins can be represented as sets which satisfy a pre-determined cutoff: for example, the set of genes whose expression exceeds some statistically-significant threshold level may be indicated as an over-expressed or up-regulated set associated with the experimental condition.

Modern genomics, in contrast to this classical understanding, represents genes in terms of their spatial location along a chromosomal sequence. Associating genes or other sequence features with their spatial coordinates along a linear chromosomal sequence enforces a geometric understanding of the structure of the genome. Genes, promoters, enhancers, transcription factor binding sites, histone locations, accessible or inaccessible regions of the DNA, noncoding RNAs, enhancers, splicing motifs, and every other sequence features are points, regions, or intervals along a 1-dimensional space of coordinates. Consequently, a spatial understanding of genomic data requires new ways of representing, storing, and querying these geometric coordinates. One-dimensional regions are characterized by their spatial distance along the genome, and by their spatial relationships to each other: inclusion, containment, and overlap.

These spatial relationships between sequence features are related to the functional relationships between the features. The regulatory relationship between a transcription factor and a target gene depends on the binding of the factor (that is, the location of its binding site, a sequence feature) within the promoter or enhancer for the target gene. Filtering these binding sites by measurements of the accessibility or inaccessibility of the surrounding DNA has been shown to improve the accuracy of regulatory relationship prediction – if a binding site is contained within an inaccessible region of DNA, then it is less likely to indicate a regulatory relationship between its binding transcription factor and any nearby gene[55, 54]. Genes themselves have a spatial structure reflected in the ordering and spacing of their exons and the relative locations of sequence signals for splicing. Higher-order spatial patterns of genes along a longer chromosomal region, such as the Hox gene clusters in vertebrates whose spatial structure reflects the order of their activation and the ultimate patterning of the organism itself, can produce functional effects and relationships.

The treatment of genomes and chromosomes as *one*-dimensional spaces is itself an over-simplification. Recent experimental work has shown that genomes have higher-order three-dimensional structure. DNA wraps around histones in packed conformations, and chromosomes contact each other in distinct locations and may even form loops or other forms of "secondary structure." The physical location of the chromatin within the nucleus itself (for instance, distance to the nuclear membrane) may even play a role in the regulation of genomic processes such as transcription[6].

Traditional relational databases for storing proteomic or gene-centric data are poorly equipped for representing and storing these sorts of spatial relationships in an efficient manner. In the UCSC relational schema, different sets of gene or sequence annotations are stored in separate table sets[31]. While this may reflect the incremental nature of the UCSC database's growth and the desire to keep separate datasets in separate locations, it also means that a query to the database which seeks *all* annotations within a given region (a "spatial containment query") will be required to touch many different tables. Even those database schema designs which present spatial coordinates in a single table, or a very limited set of tables, are often ill-suited to more complex patterns of spatial query. One prominent use of the spatial containment query is to produce results which will be used to populate a genomic browser or automatic annotation mechanism. However a visual browser will present its results to the user in a manner which depends on the spatial scale of the query – the larger the scale the less detail will be shown, with more detail presented as the user "zooms in." The rules which are used to aggregate detailed high-resolution data into results suitable for visualization are sometimes difficult to encode in a simple relational aggregation or grouping operator. This deficiency has led to the development of special data structures that exist outside a relational database altogether, allowing multi-resolution indexing and novel spatial query types which are unsupported by standard SQL languages.

1.2 Gene Expression and Genomic Transcription

Transcription of the genome, the process by which DNA is turned into mRNA, is an example of a biological process along the genome which requires a high-resolution spatial understanding. Transcription is the process by which DNA is turned into mRNA, the message which is ultimately translated into protein. We say that a gene is expressed if its sequence is transcribed. We measure the expression of a gene as a proxy for the presence and relative abundance of the protein for which the gene codes. This assumption, that expression is a suitable substitute for protein levels, is not completely accurate: the persistence or degradation of the protein itself, the editing, regulation, transport, and translation of the mRNA message, and the presence of gene isoforms and chemical modifications of the protein which affect its functional status, all mediate the relationship between observed expression of a gene and the presence or functional efficacy of its encoded protein.

Experimental methods for measuring gene expression are varied in their ability to detect novel transcripts, produce quantitative information about expression levels, and to operate at high throughput rate. Before the existence of complete genome sequences to guide them, investigators used sequencing to determine the sequence of mRNAs purified from a cell population; these ESTs (or Expressed Sequence Tags) were organized into libraries and used to enumerate sets of *transcripts* which were produced by a given type of cell under particular experimental conditions. From these ESTs, or from complete genomic sequences available later, researchers could design probes which hybridized uniquely to a single putative transcript. These probes allowed the use of blots to measure the presence and to give rough relative abundance estimates for the expression of individual genes. Other techniques, such as qPCR, were developed as more accurate methods for quantitative measurement of gene expression.

Gene expression measurements on a genome wide scale first became commonplace with the advent of microarray technologies. Microarrays are an experimental platform upon which thousands of probes can be simultaneously fixed; samples of genomic transcription are then processed, amplified, and labeled before being hybridized to the array's probes. Probes which show the presence of labeled sample material hybridized to their location provide an indication of the presence of the corresponding transcript. Quantitative estimates of expression are also possible from the amount of label present at a probe location.

A gene is expressed if there is a transcript produced by the cell which contains the coding sequence of that gene (and which is, presumably, subsequently translated into protein). Transcription is itself a spatial process. Transcripts may run beyond the annotated coding boundaries of the gene, containing the $5'$ and $3'$ UTRs (un-translated regions). Gene expression may be present as a function of a transcript spanning multiple genes simultaneously. Genes themselves may be multiply transcribed, with alternate promoters or variable endpoints. Transcription is a strand-specific phenomenon, so the ability to detect the direction in which a gene is transcribed (the presence of *sense* and *antisense* transcripts) can be important for a functional understanding of that gene's expression. Finally, transcripts themselves may occur outside the bounds of known gene annotations. Recent studies in yeast and humans have shown that significantly higher fractions of the genome are transcribed than are occupied by coding sequence annotations. Some of the transcribed regions of the genome are believed to have interactions with the expression of coding regions, or functional roles in their own right.

The first microarrays allowed expression measurements for thousands of genes simultaneously by using probes that were unique to each individual coding region. However, our spatial understanding of gene expression has shown us that only looking at *gene* expression gives an incomplete picture of the transcriptional activity along the genome as a whole. Gene-specific probes provide ambiguous information about the spatial nature of transcripts overlapping the probe and completely miss any transcripts that fall between the probes.

Tiling microarrays are designed to provide a spatial understanding of transcription along the entire genome. Microarray technologies allowing designs with large numbers of probes allow researchers to design arrays that tile an entire genomic sequencing with closely-spaced probes. The spacing of the probes provides a high spatial resolution – few transcripts are small enough to remain undetected by falling between the probes, and detected transcription can be spatially localized to the nearest measured probe. High-throughput sequencing experiments (RNA-Seq) may also be used to provide measurements at a similar genomic resolution. By sequencing and mapping transcripts sampled from the experimental population, RNA-Seq can also determine the locations and extents of transcription.

Both tiling microarrays and RNA-Seq measure populations of transcripts from an experimental sample, either by averaging them (microarrays) or sampling from them (sequencing). These technologies are able to accurately detect the spatial bounds of *transcription*, but they are unable to directly indicate the individual *transcripts* which are being measured. The computational analysis methods which have been applied to these experimental results attempt to classify transcribed regions from the surrounding background or noise, but do not attempt to reconstruct the presence, locations, or abundance of individual transcripts. This analysis is complicated by the fact that transcripts in a population may be derived from overlapping genomic regions, even on the same strand of the DNA. For example, two transcripts of a single gene which share the same end-point but start from alternate promoters will share a common suffix ($3'$ end) but the longer transcript will have a prefix ($5'$ end) that is not contained in the shorter transcript. These overlapping transcripts are observed as mixtures on the experimental apparatus (sequencing or microarrays), and are presented as regions of "complex" segmentation in existing computational analyes.

The goal of the second half of this thesis is to demonstrate a new computational algorithm for the spatial detection and analysis of overlapping transcripts in tiling microarray data. This algorithm will be adapted to use the output of standard segmentation-style analyses of microarray experiments, and will be performed independently of existing gene annotations or other sequence features.

1.3 Thesis Outline

Chapter 2 outlines our core platform, the Genomic Spatial Events (GSE) database, that provides the storage, query, analysis, and visualization framework for the rest of this thesis. We also outline, in the same chapter, two previously-published uses of the GSE system for analyzing genomic datasets in the same genome and between different species. Chapters 3 to 5 describe the application of a system built atop GSE to analyze strand-senstive tiling microarrays for the detection of coding and noncoding transcription in budding yeast. Chapter 3 describes the adaptation of a standard method for normalizing expression microarray datasets (Robust Multichip Averaging, or RMA) for use with new type of tiling microarrays. In Chapter 4, we outline how these tiling microarrays can be used to detect a biological phenomenon, same-strand overlapping transcripts, which has not been previously examined using systematic computational methods at a global genomic level. Finally, in Chapter 5, we show how these computational techniques can be used to expand and refine the existing models of transcription and regulatory control in yeast.

Chapter 2

Genomic Spatial Events Database

Our understanding of experimental data and analysis methods from Chapter 1 has driven the development of an integrated storage and analysis system for genome-mapped sequence, annotation, and quantitative data. This chapter describes GSE, the Genomic Spatial Event database, a system to store, retrieve, and analyze several types of high-throughput microarray data. GSE handles expression datasets, ChIP-Chip data, genomic annotations, functional annotations, the results of our previously published Joint Binding Deconvolution algorithm for ChIP-Chip[50], and precomputed scans for binding events. GSE can manage data associated with multiple species; it can also simultaneously handle data associated with multiple 'builds' of the genome from a single species. The GSE system is built upon a middle software layer for representing streams of biological data; we outline this layer, called GSE-Bricks. We show how it is used to build an interactive visualization application for ChIP-Chip data. This visualizer software is written in Java and communicates with the GSE database system over the network.

Some methods simultaneously collect hundreds-of-thousands, or even millions, of data points. Microarrays contain several orders of magnitude more probes than just a few years ago. Short-read sequencing produces raw reads whose individual experiment size is often measured in gigabytes [30]. For example, the Illumina sequencer produces single-end reads between 30-35 base pairs in length. A single lane of the reads from the sequencer may produce between 8-10 million reads, which (if we assume that the bases are bit-packed but that a unique identifier is stored for each read) means that storing a single lane of reads will require on the order of 120 megabytes of disk space. The Illumina sequencer has eight parallel lanes, which means that every run (around three days) will produce nearly a gigabyte of raw sequence data. Those space requirements are simply for storing the simplest output of the sequencer, without additional information as to sequence quality information or mapped genomic locations. Furthermore, standard data structures for string analysis and pattern matching often require $\mathcal{O}(N^2)$ space, where N is the total length of the strings [1]. Storing these datasets in flat files, or in naive formats on disk, quickly becomes unwieldy as the data collection effort extends over months or years worth of data from multiple investigators and laboratories.

Combining these with massive genome annotation datasets, cross-species sequence alignments mapped on a per-nucleotide level, thousands of publicly-available microarray expression experiments, and growing databases of sequence motif information, and we are left with a wealth of experimental results (and large scale analyses) available to the investigator on a scale unimagined just a few years ago.

Successful analysis of high-throughput genome-wide experimental data requires careful thought on the organization and storage of numerous dataset types. The ability to effectively store and query large datasets has often lagged behind the sophistication of the analysis techniques that are developed for that data. Many publicly available analysis packages were developed to work in smaller systems, such as yeast [53]. Flat files are sufficient for simple organisms, but for large datasets they will not fit into main memory and cannot provide the random access necessary for a browsing visualizer.

Modern relational databases provide storage and query capabilities for these vertebrate-scale datasets. Built to hold hundreds of gigabytes to terabytes of data, they provide access through a well-developed query language (SQL), network accessibility, query optimizations, and facilities for easily backing up or mirroring data across multiple sites.

Most bioinformatics tools that have taken advantage of database technology, however, are web applications. Often these tools are the front-end interfaces to institutional efforts that gather publicly-available data or are community resources for particular model organisms or experimental protocols. Efforts like UCSC's genome browser and its backing database [31], or the systems of GenBank [3], SGD [15], FlyBase [10] are all examples of web interfaces to sophisticated database systems for the storage, search, and retrieval of species-based or experiment-based data.

2.1 Design outline

2.1.1 Genomic coordinates as core schema

Our system for handling and analyzing genomic datasets is designed around a central observation: the way to integrate different experimental platforms and biological datasets at a genomic level is by establishing a common, comparable mapping of each dataset to genomic coordinates. Data structures assembled with reference to those coordinates – points, regions, and alignments, and per-base or per-bin scoring functions – are then manipulated and compared through a common set of one-dimensional geometric primitives such as overlap, containment, nearest-neighbor detection, and assignment based on distance metrics.

2.1.2 Separating mapping from data

The GSE system consistently separates the concepts of a *dataset* from its associated *mapping to a genome.* This is a necessary separation for several distinct reasons:

1. Publicly-released genomes are subject to consistent updates and revisions. Separating the data itself from its mapping to a particular genome saves space and prevents the data system from storing multiple identical copies of the data for each independent genome revision.

2. Some data might not be able to be mapped directly to the genome itself – examples include control spots on microarrays, or short-reads from sequencers

that are unable to be uniquely mapped to a genome. In many cases, we would still like these data to remain a coherent unit, and not be split into categories based on whether it can be mapped to a particular genome.

3. The mapping of some data (for instance, the short reads from modern sequencers) to a particular genome is itself an experimental process with its own sources of error. Storing the mapping separately from the data allows us to experiment with different methods for mapping or aligning arbitrary data to a genome, and provides the basis for apples-to-apples comparisons between different alignment and mapping methods.

2.2 System architecture

Most analysis methods for genomic data can be decomposed into separate processes for data retrieval and stepwise analysis. Our system should provide a common set of modular software components, that can be composed in different ways for different analysis tasks and pipelines. The entire system should be designed to provide immediate visualization to the end-user, and that visualization should be built around the same components that form the core of any analysis pipeline – this way, the user can be guaranteed that the same data and analysis results which go into his or her static visualizations or dynamic figures should be the same as go into the batch analyses which are reported by the system.

The system that we describe here bridges the gap between the web applications that exist for large datasets and the analysis tools that work on smaller datasets. GSE consists of back-end tools for importing data and running batch analyses as well as visualization software for interactive browsing and analysis of ChIP-Chip data.

The visualization software, distributed as a Java application, communicates over the network with the same database system as the as the middle-layer and analysis tools. Our visualization and analysis software is written in Java and are distributed as desktop applications. This lets us combine much of the flexibility of a web-application interface (lightweight, no flat files to install, and can run on any major operating system) with the power of not being confined to a web browser environment. Our system can also connect to datastreams from multiple databases simultaneously, and can use other system resources normally unavailable to a browser application.

This paper describes the platform that we have developed for the storage of ChIP-Chip and other microarray experiments in a relational database. It then presents our system for intepreting ChIP-Chip data to identify binding events using our previously published "Joint Binding Deconvolution" (JBD) algorithm[50]. Finally, we show how we can build a system for the dynamic and automatic analysis of ChIP-Chip binding calls between different factors and across experimental conditions.

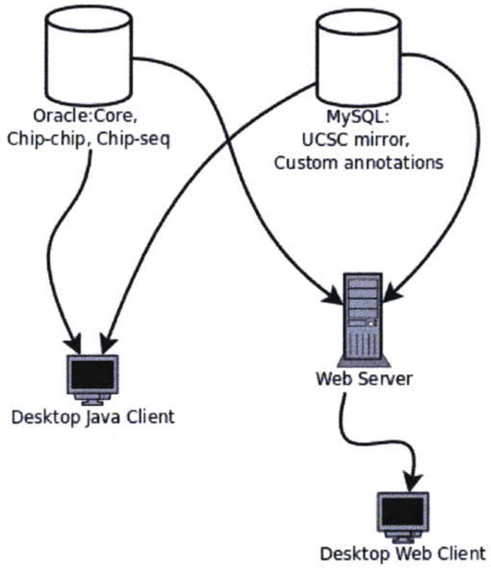


Figure 2-1: GSE is structured as either a "fat" desktop client that connects directly to the underlying database(s), or a "thin" client that can run in a web-browser.

2.2.1 Core schema

The core of our system is a database schema to represent biological data that is associated with genomic loci and associated metadata in a manner independent of specific genomic coordinates. Figure 2-2 shows the common metadata that all subcomponents of GSE share. We define species, genome builds, and experimental metadata that may be shared by ChIP-Chip experiments, expressiod experiments, and ChIP-Seq experiments. We represent factors (e.g. an antibody or RNA extraction protocol), cell-types (tissue identifier or cell line name), and conditions as entries in separate tables.

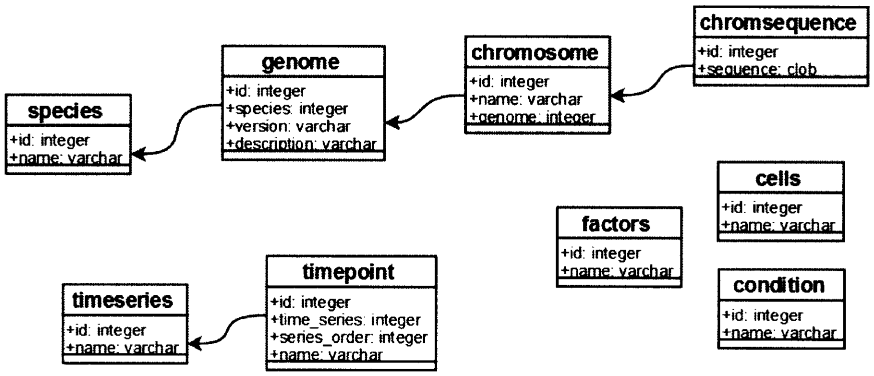


Figure 2-2: CORE relational schema for the GSE System

2.2.2 Microarray schema

GSE's database system also allows multiple runs of the same biological experiment on different array platforms or designs to be so combined. Some of our analysis methods can cope with the uneven data densities that arise from this combination, and we are able to gather more statistical power from our models when they can do so. GSE stores probes separately from their genomic coordinates as shown in Figure 2-3. Microarray observations are indexed by probe identifier and experiment identifier. A key data retrieval query joins the probe observerations and probe genomic coordinates based on probe identifier and filters the results by experiment identifier (or more typically a set of experiment identifiers corresponding to replicates of a biological experiment) and genomic coordinate. To add a new genome assembly to the system, we remap each probe to the new coordinate space once and all of the data is then available against that assembly. Since updating to a new genome assembly is a relative quick operation regardless of how many datasets have been loaded, users can always take advantage of the latest genome annotations.

In our terminology, an *experiment* aggregates datasets which all share the same factor, condition, and cell-type as defined in the common metadata tables. Each

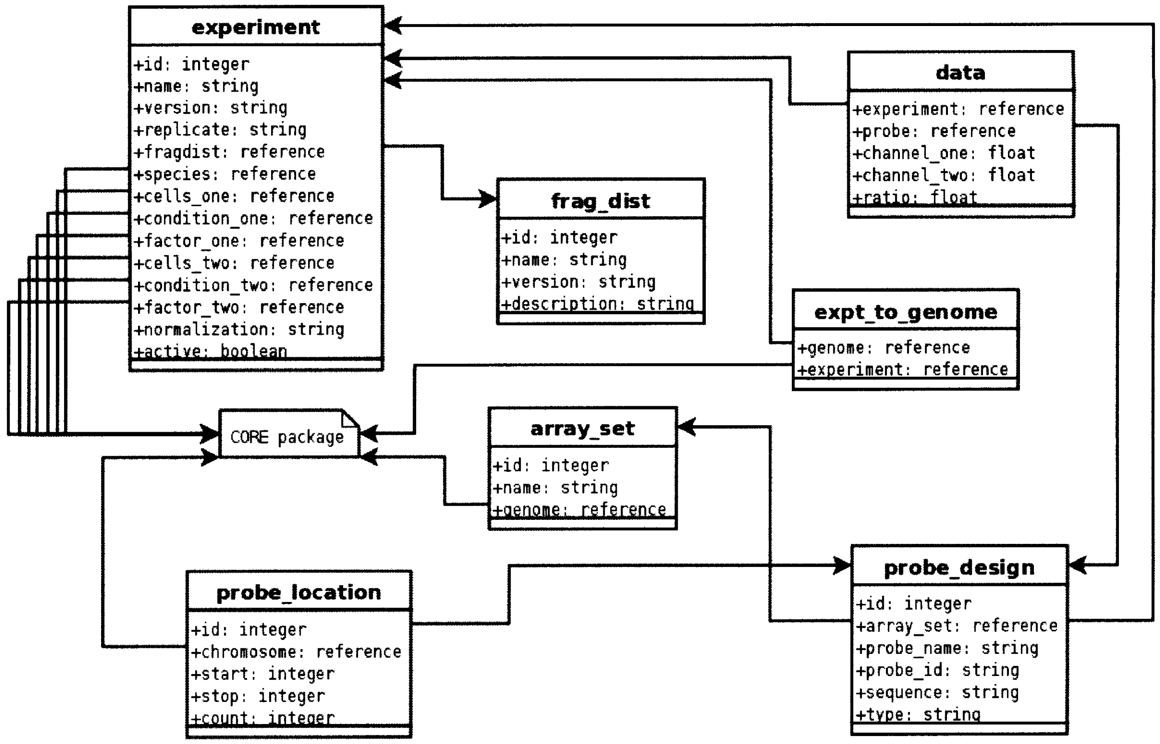


Figure 2-3: `MICROARRAY` relational schema for the GSE System

replicate of an experiment corresponds to a single observation.

2.2.3 Short-read sequencing schema

GSE supports the storage of experimental data from high-throughput short-read sequencing machines in its `CHIPSEQ` schema. The schema for this subsystem is outlined in Figure 2-4. As with array designs in the `MICROARRAY` schema, the logical identity of the reads is stored separately from their alignment and their mapped locations to a particular genome. This separation of alignment from read sequence permits (a) the representation of *unmapped* reads in the database, (b) cleaner handling of reads which map to a genome multiple times, (c) the ability to map a dataset to the same genome multiple times (for instance, if different alignment algorithms or parameters are used), and (d) the ability to maintain mappings for a dataset to multiple genomes simultaneously.

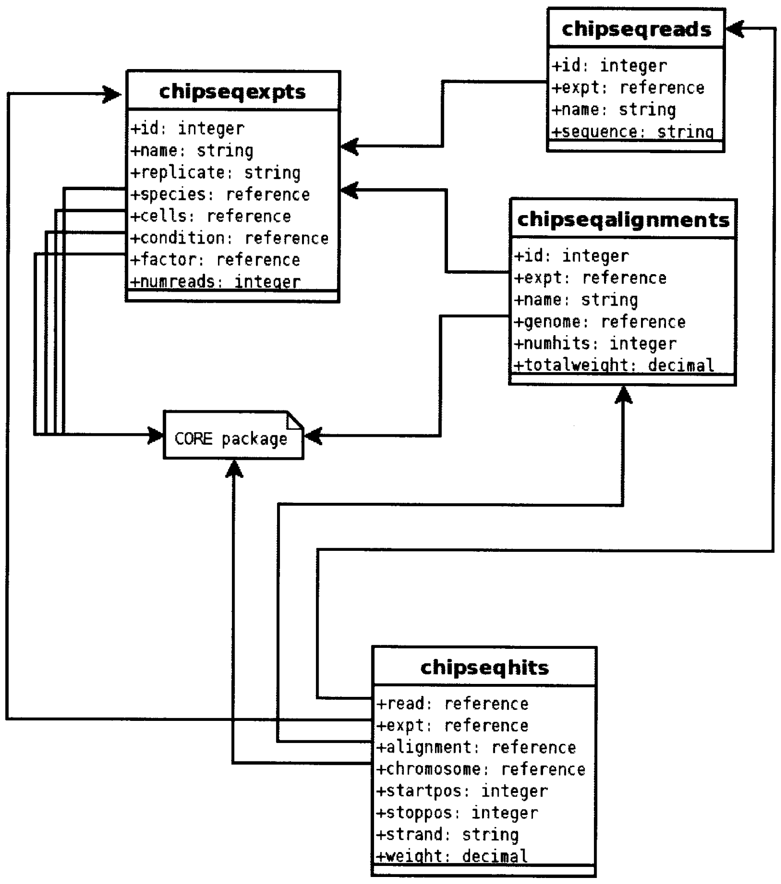


Figure 2-4: `CHIPSEQ` relational schema for the GSE System

2.3 A Dataflow Language for Biological Analysis

Our software system for the manipulation and analysis of biological data is called *GSEBricks.* GSEBricks is a modular software package that allows the user to build software analysis pipelines by composing smaller, reusable components. These components pass data structures representing biological objects between them in well-defined ways, and provide hooks for managing long-lived resources associated with the analysis pipeline as well as distributing the execution of the pipeline across multiple threads or multiple separate machines.

A GSEBricks module is written by extending one of three Java interfaces: `Mapper`, `Filter`, or `Expander`. All of these interfaces have an 'execute' method, with a single Object argument (that is type-parameterized in Java 5). The `Mapper` and `Filter` execute methods have an Object (also parameterized) as a return value. The contract of

`Mapper` is that it produces Objects in a one-to-one relationship with its input, while a `Filter` may occasionally return 'null' (that is, no value). The `Expander` execute method, on the other hand, returns an `Iterator` each time it is called (although the `Iterator` may be empty).

Composition of GSEBricks modules is carried out by combining an instance of one of these three module classes with an existing `Iterator` in order to produce a new `Iterator`. This composition happens using one of three composition classes: `MapperIterator`, `FilterIterator`, and `ExpanderIterator`. A `MapperIterator` combines a `Mapper` with an Iterator, to produce a new Iterator; each element of the new Iterator is the result of calling `Mapper` on the corresponding element of the first Iterator. The `FilterIterator` class works the same way, although the 'null' values returned are dropped from the resulting Iterator. The `Expander` works by concatenating the Iterators that result from each element of the original stream into the single new Iterator.

The second advantage to using the GSEBricks system as the basis for our visualizer is consistency. The GSE Visualizer is itself written as a modular set of custom Java Swing components, which read from GSEBricks and draw the data using the Java 2D graphics library. By building these modular graphical elements as wrappers around GSEBricks components, we are able to guarantee that the pictures from our visualizer will exactly match the data used in all our other analysis tools. This consistency is enforced because the same program code that reads and parses the data for an analysis tool is running in the visualizer itself.

This suggests the third advantage to our GSEBricks system: easy integration of analysis tools into the GSE Visualizer. Because our analysis programs are written from the same components as the visualizer, it becomes easy to simply 'plug' them into that visualizer (as dynamic options from a menu, or through interactive graphics on the visualizer itself).

The fourth advantage of the GSEBricks sysem is the easy extensibility and modification of the GSE Visualizer. Its modular design lends itself to modular extensions. We have been able to quickly extend the visualizer to handle and display data such as dynamically re-scanned motifs (on a base-by-base level within the visualized region), automatic creation of 'meta-genes' (averaged displays of ChIP-Chip data from interactively-selected region sets), and the display of mapped reads from ChIP-PET experiments.

The final advantage of GSEBricks is the extensibility of the GSEBricks system itself. By modifying the code we use to glue the Iterators together, we can replace sequential-style list-processing analysis programs with networks of asynchronously-communicating modules that share data over the network while exploiting the parallel processing capabilities of a pre-defined set of available machines.

2.3.1 Discovering and representing binding events

Modern, high-resolution tiling microarray data allows detailed analyses that can determine binding event locations accurate to tens of bases. Older low-resolution ChIP-Chip microarrays included just one or two probes per gene[18, 19]. Traditional analysis applied a simple error model to each probe to produce a bound/not bound call for each gene rather than measurements associated with genomic coordinates[61]. Our Joint Binding Deconvolution (JBD)[50] exploits the dozens or hundreds of probes that cover each gene an intergenic region on modern microarrays with a complex statistical model that incorporates the results of multiple probes at once and accounts for the possibility of multiple closely-spaced binding events.

JBD produces a probability of binding at any desired resolution (e.g. a per-base probability that a transcription factor bound that location). Figure 2-3 shows the tables that store the JBD output and figure 2-5 shows a genomic segment with ChIP-Chip data and JBD results. Unlike the raw probe observations, JBD output refers to a specific genome assembly since the spatial arrangement of the probe observations is a key input. GSE's schema also records which experiments led to which JBD analysis.

2.3.2 Genomic visualization

The GSE Visualizer is constructed as a software layer that depends on the GSEBricks library. This system provides a uniform interface to disparate kinds of data: not only ChIP-Chip data and JBD analysis, but also genome annotations, microarray expression data, functional annotations, sequence alignment and orthology information, and sequence motif data. The system also handles loading the data from multiple database servers, and smoothly combining the results for any downstream program code.

There are many advantages to using a modular stream-based data system like GSEBricks for our Visualizer and analysis software. The first advantage is that it's easy for new programmers to learn how to use the system. The GSEBricks system is centered around the Java Iterator interface: this is a Java class which returns a stream of objects one at a time. GSE's visualization and GUI analysis tools depend

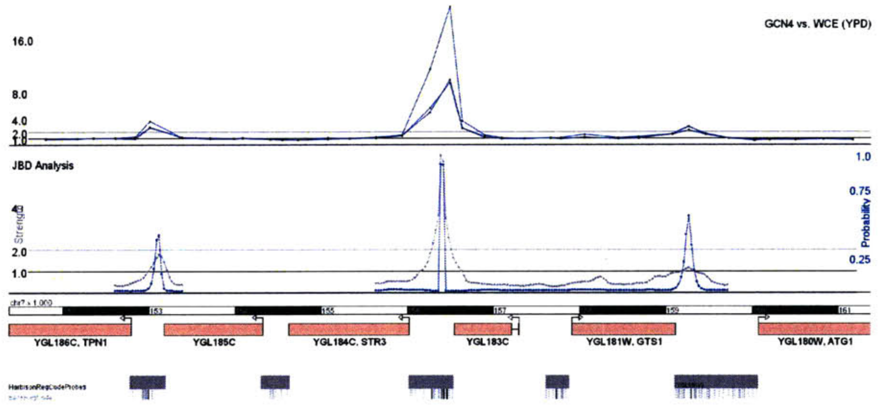


Figure 2-5: A screenshot from the GSE Visualizer. The top track represents 'raw' high-resolution ChIP-Chip data in yeast, and the bottom track shows two lines for the two output variables of the JBD algorithm. At the bottom are a genomic scale, a representation of gene annotations, and a custom painting of the probes and motifs from the Harbison et. al. Regulatory Code dataset.[17]

on a library of modular analysis and data-retrieval components collectively titled 'GSEBricks'. This system provides a uniform interface to disparate kinds of data: ChIP-Chip data, JBD analyses, binding scans, genome annotations, microarray expression data, functional annotations, sequence alignment, orthology information, and sequence motif instances. GSEBricks' components use Java's Iterator interface such that a series of components can be easily connected into analysis pipelines.

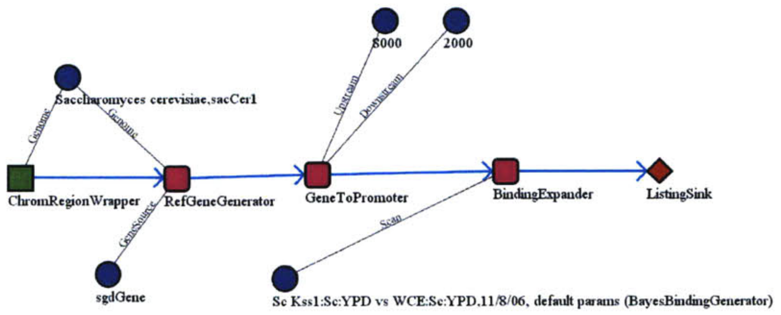
A GSEBricks module is written by extending one of three Java interfaces: `Mapper`, `Filter`, or `Expander`. All of these interfaces have an 'execute' method, with a single `Object` argument which is type-parameterized in Java 5. The `Mapper` and `Filter` execute methods have an `Object` (also parameterized) as a return value. `Mapper` produces `Objects` in a one-to-one relationship with its input, while a `Filter` may occasionally return 'null' (that is, no value). The `Expander` execute method, on the other hand, returns an `Iterator` each time it is called (although the `Iterator` may be empty).

Each GSEBricks datastream is represented by an `Iterator` object and datastreams are composed using modules which 'glue' existing `Iterators` into new streams. Because we extend the Java `Iterator` interface, the learning curve for GSEBricks is gentle even for novice Java programmers. At the same time, its paradigm of building 'Iterators out of Iterators' lends itself to a Lisp-like method of functional composition, which naturally appeals to many programmers familiar with that language.

Because our analysis components implement common interfaces (eg, `Iterator<Gene>` or `Iterator<BindingEvent>`), it is easy to simply plug them into visualization or analysis software. Furthermore, the modular design lends itself to modular extensions. We have been able to quickly extend our visualizer to handle and display data such as dynamically re-scanned motifs (on a base-by-base level within the visualized region), automatic creation of 'meta-genes'[60] (averaged displays of ChIP-Chip data from interactively-selected region sets), and the display of mapped reads from ChIP-PET experiments[37].

The final advantage of GSEBricks is the extensibility of the GSEBricks system itself. By modifying the code we use to glue the Iterators together, we can replace sequential-style list-processing analysis programs with networks of asynchronously-communicating modules that share data over the network while exploiting the parallel processing capabilities of a pre-defined set of available machines.

Figure 2-6 shows a screenshot from our interface to the GSEBricks system. Users can graphically arrange visual components, each corresponding to an underlying GSE-



```
BindingScanLoader loader = new BindingScanLoader();
Genome sacCer1 = Organism.findGenome("sacCer1");

ChromRegionWrapper chroms = new ChromRegionWrapper(sacCer1);
Iterator chromItr = chroms.execute();

RefGeneGenerator rgg = new RefGeneGenerator(sacCer1, "sgdGene");
Iterator geneItr = new ExpanderIterator(rgg, chromItr);

GeneToPromoter g2p = new GeneToPromoter(8000, 2000);
Iterator promItr = new MapperIterator(g2p, geneItr);

BindingScan kss1 = loader.loadScan(sacCer1, kss1_id);
BindingExpander exp = new BindingExpander(loader, kss1);
Iterator bindingItr = new ExpanderIterator(exp, promItr);

while(bindingItr.hasNext()) {
    System.out.println(bindingItr.next());
}
```

Figure 2-6: A GSEBricks pipeline to count the genes in a genome. Each box represents a component that maps objects of some input type to a set of output objects. The circles represent constants that parameterize the behavior of the pipeline. The code on the right replicates the same pipeline using Java components.

Bricks class, into structures that represent the flow of computation. This extension also allows non-sequential computational flows – trees, or other non-simply connected structures – to be assembled and computed. The interface uses a dynamic type system to ensure that the workflow connects components in a typesafe manner.

Workflows which can be laid out and run with the graphical interface can also be programmed directly using their native Java interfaces. The second half of Figure 2-6 gives an example of a code-snippet that performs the same operation using the native GSEBricks components in Java.

Chapter 3

Probabilistic Modeling and Segmentation of Tiling Microarrays

Microarrays have been used to measure the levels of gene expression for over a decade [20]. As their use became widespread, the need to correct for sources of "obscuring variation" and to normalize the results for comparison between experiments has been a key concern for biologists. Without a correction and normalization step, probe-specific and experiment-level technical effects can obscure the interesting variation due to biological causes, and make it difficult for microarrays to inform studies of either comparative or absolute gene expression. Corrected and normalized data, however, allows the use of statistical tests to detect differential and absolute gene expression levels and to illuminate biological mechanisms of regulation and transcription. Many of the techniques which are applied to tiling or other high-density microarrays today will be extended and applied to sequencing or other high throughput techniques tomorrow.

One relatively new development in microarray technology is the advent of the *tiling microarray*. Unlike earlier array designs, which sought to probe known or putative transcripts with one or more individual probes on the array, tiling arrays attempt to probe each region of the entire genome at a nearly-uniform density. Modern biological understanding of transcription has shown that, in many organisms, much higher fractions of the genome are transcribed than were previously believed. Much of that transcription is known to take place *outside* of confirmed gene annotations or putative open reading frames. However, microarrays which were designed to probe known gene annotations will be biased against the measurement of these non-coding transcripts. The goal of these tiling designs is fill in this blind-spot, and to allow for the unbiased measurement of transcripts throughout an entire genome.

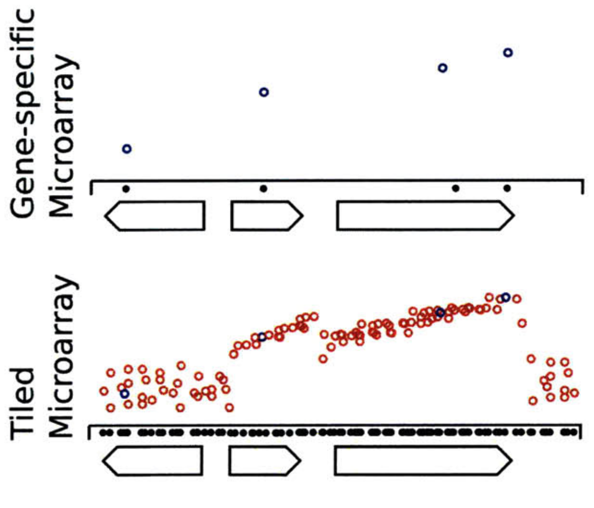


Figure 3-1: Unbiased Assessment of Transcription via Tiling Microarrays
Older array designs measure only the expression of annotated transcripts. Tiling microarrays, by providing complete measurements along the entire genome (red probes), give a view of transcription unbiased by the annotation set.

As the total number of publicly-available microarray datasets has increased, and as the density of the microarrays themselves has deepened, efficient methods for normalization have become more important. However, many of these normalization methods were designed for earlier annotation-based non-tiling microarray designs – their goal is the determination of unbiased estimates for the expression levels of individual genes which are probed by the array design. Because tiling microarrays lack this *a priori* assignment of probes to annotations that make traditional normalization methods possible, their analysis often requires an additional pre-processing step: the *segmentation* of probes into subsets which are then considered exchangeable witnesses to the expression of a single underlying transcript.

Modern sequencing techniques have also allowed, in the last three years, a new experimental technique for measuring transcription and other genome-wide biological events at a high resolution. RNA-Seq has become an established method to discover the location, extent, and intensity of transcription throughout an entire genome.

3.1 Prior Work

Before any multi-experiment microarray dataset can be analyzed, it must first be normalized for inter-array comparison. Computational methods for normalization have been studied for almost as long as microarray expression data has been produced [46]. Many normalization methods are based on manufacturer-specific features of microarray design. Normalization methods for Affymetrix arrays, which contain perfect-matching and partial mis-matching (PM and MM) *probe sets*, can be categorized by whether they use the mismatch probe information in their normalization calculation. The standard method for averaging multiple Affymetrix array experiments, Robust Multichip Averaging (or RMA), was developed by Irizarray, Bolstad, and Speed [5]. Other methods are designed for two-color array platforms, or for platforms which contain spots for "spike-in" or other artificial control values [23]. Some normalization methods attempt to reconstruct relative intensities against a baseline experiment (often the "zero" time point in a time-series experiment). Other methods depend on biological assumptions such as the constant expression of "housekeeping" genes or the constant level of total RNA content in the sample cell population [65].

The literature on the subsequent analysis, after normalization, of tiling microarrays or other whole-genome experimental methods for measuring transcription can be divided into different categories based on the experimental platform for which the analysis method was originally designed and the method by which the locations of gene annotations are incorporated into the analysis. The *supervised* analyses of high-density microarrays depends on the locations of the probes relative to the gene annotations of the measured genome. These methods estimate expression values for particular genes by summarizing the observed intensities of the array probes that map within that gene's annotation (or a portion of the annotation, for splice-form specific estimation). Most methods for the *unsupervised* analysis for tiling microarrays, in which gene annotations do not constrain the set of analyzed probes, were derived from older statistical models for lower-density arrays. The simplest of these methods emphasize the identification of individual "enriched" or "bound" probes [61]. When applied to a tiling or other high-density microarray, these methods were adapted to look for consecutive sequences of enriched probes that would indicate consistently high levels of expression or binding. Different approaches vary in the complexity of the model they deploy to detect bound or expressed probes. For example, a study of noncoding and intergenic expression in *Mycobacterium leprae* determined expression based on consecutive runs of four probes with intensities greater than 60% of the maximum normalized probe intensity score [2].

A second general approach for the analysis of tiling microarrays is derived from methods in use for array-CGH data. These methods utilize a dynamic programming (DP) algorithm to induce a *segmentation* of the corresponding tiling array data. Picard et al. initially described a segmentation algorithm for learning constant-intensity regions in array-CGH data [48]. Their algorithm could be provided with the requirement that segment-level noise be constant across the entire array (homoscedastic) or could vary from segment to segment (heteroscedastic); their output produced a sequence of *breakpoints*, or segment boundaries, which divided the genome into regions of constant experimental intensity in a provably optimal way. The work of David et al. pioneered the use of the Picard segmentation method to interpret dense tiling arrays for transcription data in yeast [11, 26]. However, they had to adapt an additional statistical model for the post hoc classification of segments according to whether they were transcribed or not. Picard and his collaborators later developed a hybrid algorithm which included both of these steps in a single iterative expectation-maximization-style approach [49].

Some algorithmic approaches for identifying transcription in microarray data have relied on statistical methods for identifying "change-points" in time-series data, adapted from signal processing and econometrics literature [35]. These methods attempt to find hinges or "change points" analogous to the breakpoints induced from a segmentation. Change point regression techniques can utilize statistical tests for the existence of a hinge or breakpoint at a certain location, related to the relative accuracy of a single linear model against two spatially separate linear models for explaining the data in a given genomic region [34]. Bayesian extensions to classical regression techniques have also been suggested for the change-point estimation problem [16].

Another algorithmic approach to segmentation uses hidden Markov models (HMMs) for the discovery of hidden genomic "states" to which genomic regions are jointly assigned and which explain the observed array data [56]. Standard HMM inference algorithms utilize a dynamic programming approach which closely resembles that of standard segmentation algorithms – the differences exist in how the models are parameterized, with HMMs allowing an easier specification of the total number of "states" in the model but produce spatial effects through an obscure transition matrix parameter, while segmentation methods are often easier to parameterize in spatial terms but are usually only provided with two separate states (a choice which is generalized in this thesis).

High-throughput sequencing for genome-wide transcription has its own set of standard analysis methods. Typically, a sequencer produces distinct sequence *reads* which are mapped back to unique locations (*hits*) within the target genome; due to the statistical sampling nature of the way in which reads emerge from the sequencer, we expect those target genomic regions with higher numbers of overlapping hits to be more likely part of the biological phenomenon measured by the experiment. Simple analyses mapped sequencing data can be performed by shoehorning the hit locations and counts into a standard microarray-style analysis method. For example, genomic *bins* can be defined *a priori* along the length of the target genome, and those bins which carry a statistically significant number of hits relative to a model of expected counts are reported as bound or transcribed [44]. More sophisticated techniques which rely on the unique nature of sequence data can be utilized as well. In their study of yeast transcription from cDNA sequencing, Miura et al. determined unique sequence sigatures indicating the $5'$ and $3'$ ends of transcription and mapped those ends by scanning for regions which were enriched with hits bearing those unique sequence signals [42].

3.2 Notation

We will adopt the notation shown in Figure 3-2 for tiling array data and segmentation throughout the rest of this thesis. Taking the design of an array to be understood from

$i \in \mathbb{I}$	:	total set of probes
$j \in \mathbb{J}$	:	total set of experiments
x_i	:	location of probe i
y_{ij}	:	intensity of probe i in experiment j
$s \in \mathbb{S}$	:	total set of segments
s_i	:	segment of probe i
Θ_s	:	parameters of segment s
$\mathbf{5}_s$, $\mathbf{3}_s$	:	$5'$, $3'$ ends of segment s

Figure 3-2: Array and Segment Notation

the context, a probe on an array design will be indexed with the letter i, and the total set of probes on the array will be denoted $\mathbb{I}$. We omit the details of multi-chip designs and assume that all probes are present on a single physical microarray. The location of probe i will be indicated with variable x_i; by "location" we mean the chromosome name and base-pair offset of the middle of the probe with respect to an understood global genomic coordinate system. For example, *chromosome* 8*, base* 1,206,312 *in the mouse genome version* ***mm9*** is a complete location, although this level of detail will be rarely given. Probes are assumed to be of a uniform, negligible width — the physical extent of the probe will not play a role in this analysis. Furthermore, we assume that the standard set of one-dimensional geometric operations are available for genomic locations (of probes or other sequence features) on the same chromosome.

Experimental observations of probe intensities will be indexed with the letter j, and the total set of available experiments (where such an understanding is possible) as $\mathbb{J}$. The observed *value* of probe i in experiment j will be denoted s_{ij}, and the normalized *intensity* of the same probe in that experiment as y_{ij}.

The set of genes (which I will take to be equivalent to a set of gene annotations for the purposes of this work) will be indexed with the letter g, and the total set of genes will be given the name $\mathbb{G}$. When necessary, the start and end coordinate of the gene g will be called s_g and e_g and will be assumed to have the same form as the probe location values. The orientation of a gene o_g is a binary variable taking either the value **W** or **C** (for Watson or Crick, the two complementary strands of the chromosomal DNA).

Later, as we develop and examine algorithms for segmenting and understanding genomic data, we will use the notation $\mathbb{S}$ for a segmentation of a genomic dataset. A segmentation consists of a set of *segments* – each segment is a genomic region, and together the segments partition the total genomic coordinate space (the complete sequence). For any genomic location l we will let $s[l]$ denote the segment in which that location falls (where the segmentation $\mathbb{S}$ is understood); when discussing a particular probe i, we will let s_i be a shorthand for $s[x_i]$.

Dual to the notion of a segmentation $\mathbb{S}$ is the set of *breakpoints* $\mathbb{B}$ of that segmentation. The breakpoints are the genomic locations which form the boundaries between consecutive segments – any set of breakpoints determines the corresponding segmentation, and any segmentation gives rise to a set of breakpoints. I will use the notation $\mathsf{b}_l[s]$ and $\mathsf{b}_r[s]$ to indicate the *left* and *right* breakpoints, respectively, of the segment s. If segments s and s' are adjacent along the genome in a given segmentation, then either $\mathsf{b}_r[s] = \mathsf{b}_l[s']$ or $\mathsf{b}_l[s] = \mathsf{b}_r[s']$, that is, adjacent segments have a breakpoint in common.

Breakpoints also allow us to define *sub*-segmentations; a sub-segmentation is a segmentation of a chromosomal region either before or after a breakpoint. Subscript and superscript notation will define either the left or the right sub-segmentation: $\mathbb{S}_b = \{s \in \mathbb{S} : \mathsf{b}_r[s] < b\}$ and $\mathbb{S}^b = \{s \in \mathbb{S} : \mathsf{b}_L[s] \geq b\}$.

3.3 Probabilistic Models of Probe Intensities

Every method for manipulating or analyzing microarray data begins with a model of probe intensities. Normalization methods model observed probe signals as the combination of a biological signal and different technical or experimental noise components. Gene summarization methods, on the other hand, treat the normalized probe intensities as measurements of underlying gene expression values. Both normalization and summarization algorithms attempt to infer these component values from the total set of observed probe signals or intensities.

In this section, I outline a standard model for probe intensities central to a standard normalization and summarization algorithm known as Robust Multichip Averaging (RMA). The notation will be slightly different from the presentation given in the original RMA publications, in order to allow for easy extension, in subsequent sections, as we relax some of the assumptions inherent in the original RMA formulation.

According to the algorithm's original developers [5], "RMA" is the the proper name for the last step of a three-step process of estimation and averaging (although all three steps are available as a packaged standard analysis method for Affymetrix microarrays in the Bioconductor R software package):

1. *Background Correction:* Additive "background" terms that are uniform across a single array are estimated and subtracted from each probe on the array.

2. *Normalization:* Quantile normalization is performed on a combined set of arrays, under the assumption that the experiments have roughly similar distributions of expression values.

3. *Probe-Affinity Correction and Gene Averaging:* The parameters of a linear model, including parameters for both gene-level expression and probe-level affinities, are fit using a robust estimation method.

RMA starts by assuming that, on an array j, each probe i has a measured signal intensity s_{ij}. The first step, *background correction*, attempts to perform a blind separation of this signal into two components,

$$\hat{y}_{ij} = s_{ij} - b_j \tag{3.1}$$

where b_j is the "background" term for experiment j.

These corrected probe signals are then normalized so that the measurements from all arrays in a sample share a common distribution.

$$y_{ij} = \mathcal{Q}_{\mathbb{J}}(\hat{y}_{ij}) \tag{3.2}$$

Standard practice in the use of the RMA package identifies $\mathcal{Q}_{\mathbb{J}}$ with the process of *quantile normalization* across the set of experiments $\mathbb{J}$, although other normalization algorithms could be used in its place where appropriate. Once the y_{ij} values have been determined from background-correction and normalization they modeled as the sum of a probe-specific intensity term α_i and an experiment-specific gene expression term γ_{gj}:

$$y_{ij} = \alpha_i + \gamma_{gj} + \epsilon_{ij} \tag{3.3}$$

The $\epsilon_{ij} \sim \mathcal{N}(\cdot; 0, \sigma_y)$ is a unit-level error term. γ_{gj} is the expression level of the gene g in experiment j; it is the value that we wish to estimate, and it is the same for any probe that is designed to measure the expression of gene g. The α_i is a bias term which captures effects on probe intensity that are specific to the probe (and not the gene for which it is designed or the experiment in which it is measured). This probe bias term is intended to encapsulate probe-level biological effects such as non-specific binding affinity and effects that depend on the sequence of the probe.

In the original formulation of RMA, the terms α_i and γ_{gj} are jointly estimated using the *median polish.* Median polish is an iterative matrix method for estimating two-factor models that subtracts medians in order to be robust to outliers but is not guaranteed to converge [33]. After correcting, normalizing, and estimating the probe-level error terms α_i, the final step is to summarize the complete set of probe intensities into a gene-level expression value: γ_{gj}. This final summarization step depends on the set I_g, the set of probes which are designed to measure the expression of gene g.

$$\gamma_{gj} = \frac{1}{|I_g|} \sum_{i \in I_g} (y_{ij} - \alpha_i) \tag{3.4}$$

I_g, or the *design* of the array, is an input to the RMA algorithm and must be known *a priori* for the gene-level estimation to be carried out.

The nature of the Affymetrix experimental protocol and designs, in which perfect-match probes are designed to uniquely interrogate a particular transcribed region and the experimental protocol priming and reverse transcription (RT) to convert mRNA into DNA for hybridization with the array, leads to a situation in which each probe is assumed to be an identically-distributed measurement of the concentration of gene it is designed to measure. This is the reason that the core RMA model, Equation 3.3, has no additional covariates per probe beyond the factors for the probe-level error and gene's expression. This is easily generalized. In Equation 3.5, we give a generalization to Equation 3.3 that incorporates a vector of probe-level predictors:

$$y_{ij} = \alpha_i + \gamma_{gj} + \beta \mathbf{x}_i + \epsilon_{ij} \tag{3.5}$$

where x_i is the vector of predictors for probe i and β is the predictive coefficients corresponding to those covariates. Here the α_i term from Equation 3.3 has been split into two terms: $\beta \mathbf{x}_i$, the systematic component, and a new random component α_i. An example of this sort of generalization would be to represent the GC content of each probe as a separate predictor, and use the β vector to include a global estimate of GC content on the observed log-intensity of a probe.

When using this extended model it may no longer be possible to simply estimate the probe-level covariates by a process analogous to the median polish process of the original RMA method. Chapter 5 will give an example of when probe-level covariates are useful. In that chapter, the covariates for each probe will include the genomic distance from the probe to the $3'$ end of a given transcript or segment, and the β_i parameter will measure the rate of signal "fall-off" measured in the experiment.

3.4 Segmentation of Tiling Microarrays

RMA, a successful microarray normalization and comparison method, requires the array design I_g as input. Since tiling microarrays explicitly eschew such an *a pri-*

ori design for the purposes of unbiased detection of transcription – *unbiased*, in this case, by the annotation knowledge that would form the core of an array design. Instead of designing the arrays around sets of annotations, probes are taken from the genome at uniform or nearly-uniform spacings.

Yet the ability to apply an RMA-like algorithm to tiling microarrays is an important goal: the uniform nature of tiling array designs does not eliminate the need to correct for probe-specific errors or to average multiple measurements estimation of transcript-level values. Instead, we are left with the problem of computationally determining or inferring the array design parameter I_g so that an RMA-style algorithm can be fruitfully executed.

An algorithm which *learns* the design of the array from the data itself – a process which partitions the probes on an array into separate sets which each interrogate a separate underlying transcript – is called a *segmentation* algorithm. In this chapter, I describe the design and implementation of a segmentation algorithm, `SEG`, for analyzing tiling microarray experiments. The `SEG` algorithm extends previous computational segmentation techniques in two specific ways:

1. the ability to jointly segment multiple experiments simultaneously, and

2. the use of distinct shapes for different types of segments (automatic differentiation between *transcribed* and *background* regions of the genome.

These extensions have been devised to deal with the particular details and requirements of the tiling microarray experiments described in Chapter 5.

At a global level, the goal of a segmentation algorithm is to build a complete, spatial, probabilistic model for the intensity of every probe on the entire array. This global model, however, is stitched together out of smaller models which model the intensities of the probes in a specific, coherent, spatially-localized region: a *local model.* A segmentation algorithm may use a single uniform class of local models to build its global representation, or it may draw its local models from several different classes. For example, it may have one class of local models for transcribed regions and a second class for background or "noise" regions; in this way, the segmentation not only provides a complete probabilistic prediction for the intensity of every probe on the array, but it also provides an implicit *labeling* of those probes depending on which class of local model is chosen for that probe's location.

Regardless of whether the segmentation algorithm uses a single or multiple classes of local model, its ultimate goal is to stitch together a complete set of local models into an optimal global model for probe intensities. In this work, as in earlier segmentation algorithms, the measure of "optimality" will include a likelihood function for the probe intensities. However, it may also include separate terms such as prior probabilities on the total number or length of segments and the total number of different kinds of segments (if we have a prior belief about the total amount of transcription across the genome, for instance).

Local Models of Transcription

The core of any model for continuous spatial measurements along a genome (i.e. either a tiling microarray or read-counts from sequencing) is a *local model* for the experimental observations in a given genomic region. In this work, these local models take the form of probabilistic generative models for the intensities of tiling microarray probes that fall within a given region. A local model is parameterized at least by the bounds of the region it is meant to model; it may also take other pararameters as well.

The simplest local model that we will deal with in this thesis is the *flat* model for transcription. The flat local model is parameterized by both its region boundaries

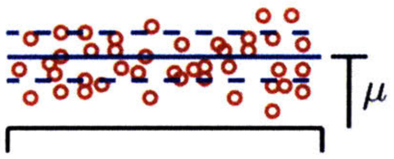


Figure 3-3: Local Flat Model
Probes within a region are modeled as iid replicates from a single distribution that contains no spatial component.

as well as by the mean μ and variance σ^2 of the probes within it. The probabilistic model

$$P(y_{ij}|\mu,\sigma^2,x_L,x_R) = \begin{cases} \mathcal{N}(\cdot;\mu,\sigma^2) & \text{if } x_L \le x_i \le x_R \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

The likelihood function of the flat probability model assumes that the probe values within the region fit by the flat model are independently and identically distributed according to Equation 3.6, and that every experiment j has its own set of parameters $\theta_j = \langle \mu_j, \sigma_j^2 \rangle$ and $\theta_{\mathbb{J}} = \{\theta_j : j \in \mathbb{J}\}$. The boundary parameters x_L and x_R imply an

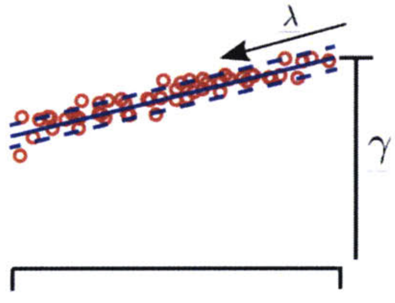


Figure 3-4: Local Linear Model
The local linear model is parameterized by an intensity (the value of the transcription at the 3′ end of the segment) and a falloff parameter. Together, these form a local linear model that is sensitive to the spatial location of probes within the segment.

assignment of probes to a local model – a probe is said to be assigned to a local model if its location x_i is in the range $[x_L, x_R)$. A local model is a probability distribution assigning positive probability to the observed intensities of all the assigned probes.

$$\mathcal{L}_F(x_L, x_R, \theta_{\mathbb{J}}) = \sum_{i:x_i\in[x_L,x_R)} \sum_j \log P(y_{ij}|x_L, x_R, \mu_j, \sigma_j^2) \tag{3.7}$$

Dynamic Programming for Spatial Segmentation

A local model of transcription provides a basic description of the probe intensities in a particular region; however, we are interested in a *global* global model for transcription. A global model for transcription divides an partitions a genomic region (of length L) into local models – every probe within the complete genomic region must fall within exactly one segment.

A global model is therefore defined by two elements: a segmentation $\mathbb{S}$, and an assignment $\Theta : \mathbb{S} \to \theta_{\mathbb{J}}$ that associates a local model likelihood $\mathcal{L}_F(\mathsf{b}_L(s), \mathsf{b}_R(s), \mu_s, \sigma_s^2)$ to every segment s. The segmentation provides the complete set of boundaries for every local model in the global segementing model; the assignment matches a set of local model parameters to each experiment in each segment of the segmentation. Together, the two components provide a complete probabilistic description of the observed intensity of every probe on the tiling microarray, and can be written as a total data log likelihood:

$$\mathcal{L}(\mathbb{S}, \Theta) = \sum_{s \in \mathbb{S}} \mathcal{L}_F(\mathsf{b}_L[s], \mathsf{b}_R[s], \Theta(s)) \tag{3.8}$$

The goal of a segmentation algorithm is to find the "best" set of local models for the observed tiling array data. In this thesis, we interpret "best" to mean that we find the segmentation and assignment function which optimize the total data log likelihood function in Equation 3.8. The first step is to notice that, since the probes are conditionally independent of each other given the segmentation and the local model parameters for their assigned segment, optimizing the assignment function is easy if we have already determined the segmentation: we can simply find the maximum likelihood parameters $\theta_s^* \equiv \langle \mu_s^*, \sigma_s^* \rangle = \arg\max_{\mu,\sigma} \mathcal{L}_F(\mathsf{b}_L[s], \mathsf{b}_R[s], \mu, \sigma^2)$ given the probes which are assigned to that segment. Therefore, our total data log-likelihood can be written as a function of just the segmentation $\mathcal{L}_S(\mathbb{S}) = \max_\Theta \mathcal{L}(\mathbb{S}, \Theta)$, and the segmentation algorithm need only optimize this reduced log likelihood.

Optimizing just a function of the segmentation itself, $\mathcal{L}_S(\mathbb{S})$ may appear to be difficult on its own, until we realize that the optimal segmentation has an optimal substructure: if we choose any breakpoint $b \in \mathbb{B}$ of the *optimal* segmentation $\mathbb{S}$, then the sub-segmentations $\mathbb{S}_b$ and $\mathbb{S}^b$ must themselves be optimal segmentations of the sub-regions $[0, b)$ and $[b, L)$. (By contradiction: if they were not, then we could substitute the better sub-segmentation into the original "optimal" segmentation and thereby improve its total log likelihood.)

This optimal sub-structure means that we may recursively solve the problem of finding the optimal segmentation for a region $[0, L)$ by finding the optimal segmentations of successively smaller pieces of the region and then "piecing them back together." To show how to compute this optimal global log-likelihood function, we write a recursive expression for the global log likelihood of any region:

$$\mathbb{L}(x_1, x_2) = \max \left\{ \begin{array}{l} \pi_F(x_1, x_2, \theta^*) + \mathcal{L}_F(x_1, x_2, \theta^*) \\ \max_b(\mathbb{L}(x_1, b) + \mathbb{L}(b, x_2)) \end{array} \right\} \tag{3.9}$$

Equation 3.9 defines the optimal total data log-likelihood for a region $[x_1, x_2)$ of the genome - but to find the optimal segmentation, and not just the log-likelihood score of that segmentation, we need a corresponding function that records the choice made at each step in the recursion. We use the π_F function to denote a complexity penalty, which constrain the ultimate size and number of segments as explained in Section 3.4.1. To make this explicit, we write a "choice function" $\mathbb{H}$ whose definition mirrors that of Equation 3.9.

$$\mathbb{H}(x_1, x_2) = \max \left\{ \begin{array}{ll} \mathsf{F} & \text{if } \mathbb{L}(x_1, x_2) = \pi(x_1, x_2, \theta^*) + \mathcal{L}_F(x_1, x_2, \theta^*) \\ \mathbb{H}(x_1, b) + \langle b \rangle + \mathbb{H}(b, x_2) & \text{if } \mathbb{L}(x_1, x_2) = \mathbb{L}(x_1, b) + \mathbb{L}(b, x_2) \end{array} \right\} \tag{3.10}$$

(Here we use the notation $\langle \ldots \rangle$ to denote an ordered list.) Notice that in the optimal segmentation, each segment s gets its own optimal set of parameters which maximize the likelihood $\mathcal{L}(\mathsf{b}_L[s], \mathsf{b}_R[s], \theta_j)$ for every experiment j; howeer, the breakpoints $\mathbb{B}$ of the segmentation are *shared* across all experiment $\mathbb{J}$.

Multiple Local Models for Transcription

So far we have described a form of segmentation that fits only one kind of local model to the observed data. Our segmentation algorithm fits the data by finding the optimal arguments (segment boundaries and parameters) to the single local likelihood function $\mathcal{L}_F$. However, we often find it useful to be able to fit *two* or more different kinds of local models. For example, in the Σ1278b data described in the final chapter of this thesis, *transcripts* are distinguished from a "background" or noise level by a characteristic falloff or slope from the $3'$ to the $5'$ ends of the transcript. This is modeled not with a flat segment but with a linear fit. The likelihood function now takes three parameters (an intercept, a slope, and a variance term).

$$P(y_{ij}|\gamma, \lambda, , \sigma^2, x_L, x_R) = \begin{cases} \mathcal{N}(\cdot; \gamma + \lambda\delta_{is}, \sigma^2) & \text{if } x_L \leq x_i \leq x_R \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

We have introduced the notation $\delta_{is} = \mathsf{b}_R[s] - x_i$.

The local model likelihood $\mathcal{L}_T$ has the same form as $\mathcal{L}_F$, and the total likelihood is modified to add *two* separate choices in the optimization.

$$\mathbb{L}(x_1, x_2) = \max \left\{ \begin{array}{l} \pi_T(x_1, x_2, \theta_T^*) + \mathcal{L}_T(x_1, x_2, \theta_T^*) \\ \pi_F(x_1, x_2, \theta_F^*) + \mathcal{L}_F(x_1, x_2, \theta_F^*) \\ \max_b(\mathbb{L}(x_1, b) + \mathbb{L}(b, x_2)) \end{array} \right\} \tag{3.12}$$

The "choice" function for this recursive likelihood function can now return *either* F, T, or a breakpoint b. I will refer to this hybrid segmentation algorithm with two forms of local model by the name SEG.

3.4.1 Segmentation Complexity Penalties

We have introduced extra terms in the recursion equation describing the log-likelihood scoring function which is optimized by the dynamic programming algorithm for segmentation. Both terms, a $\pi_T(x_1, x_2, \theta)$ and a $\pi_F(x_1, x_2, \theta)$ which are added to the local model score for the linear and flat models respectively, are additive and are functions of both the location of the segment and its local parameters. This formulation leaves unspecified the exact functions which describe these penalties.

There are three choices that the segmentation algorithm must make in order to build an optimal global model out of segments and local models. The first choice is whether to split a given region into optimally-explained sub-regions, or to model the region with a single local model. The second choice is the selectoin of which local model to use: linear (transcribed) or flat (no transcription)? After having chosen a class of local model to explain a particular region, the third choice is that of the parameters of the model which are used to explain the data.

Each of these choices will always be made in one particular way by an *unconstrained* segmentation algorithm. The choice of *splitting* a region into sub-regions, of choosing a breakpoint, allows the segmentation algorithm more flexibility by providing it the ability to multiply the number of degrees of freedom it has to explain the data almost ad infinitum (we typically assume a minimum segment size of between three and five, so dictating that breakpoints are only unique up to the location of their neighboring probes is means that regions are not infinitely divisible). Each time a breakpoint is chosen, the segmentation algorithm has twice as many parameters to use to explain the same amount of data; therefore, an unconstrained segmentation algorithm that attempts to optimize the log-likelihood of the data will *always* choose to use as many segments as possible. Furthermore, the local linear model *contains* the local flat model   any choice of parameters for the flat model will have an identical log-likelihood score to a linear model with the same intercept γ as the flat model's mean μ and a zero slope λ. An unconstrained segmentation algorithm will therefore never choose a flat model for any region, since the log-likelihood of the flat model is strictly dominated by that of the linear model.

A complexity penalization will constrain the segmentation algorithm, and prevent it from blindly choosing the most complex model (minimally-sized segments that are all fitted with the linear local model) for the complete set of genomic data. To provide a complexity penalty, we use a scoring function which independently biases each of the three choices the segmentation algorithm can make. These scoring functions are penalties which reduce the log-likelihood score of more complex models, ensuring that those models are chosen only when they improve our understanding of the data. Each of these terms contains free parameters which must be independently set *before* we apply our algorithm to real datasets.

The first penalty term biases our segmentation algorithm against the choice to split a region into sub-regions. This function subtracts a value from the log-likelihood that is linearly related to the size of the local model segment: smaller segments receive a larger penalty. Segments of length greater than 1000 bp receive no penalty, and those of the minimum length (a minimum of 50 bp, five consecutive probes) receive a penalty C_b. Segments of width $50\text{bp} < w < 1000\text{bp}$ receive a linearly-interpolated penalty of $(1 - \frac{w}{1000})C_b$.

The second penalty subtracts a constant bias term C_t from the fitted log-likelihood of the local linear model. This biases the segmentation algorithm against the choice of local linear models for regions of the genome. The final term of the complexity penalty is a term on the intensity μ of flat segments. To incorporate this bias into our model, we choose a term C_f that penalizes *higher* values of the μ parameter. The penalty term for a parameter μ is $C_f\mu$, which resembles an exponential prior distribution on the log-intensity of the flat segment.

Taken together, we can now write out functional forms for the *pi* complexity penalty functions in Equation 3.12:

$$\pi_T(x_1, x_2, \gamma, \lambda, \sigma) = -(C_t + C_b|x_2 - x_1|) \tag{3.13}$$

$$\pi_F(x_1, x_2, \mu, \sigma) = -(C_f\mu + C_b|x_2 - x_1|) \tag{3.14}$$

The parameters C_t, C_f, and C_b are all free parameters that must be chosen prior to running the segmentation algorithm. In practice, we choose these parameters by a gridded search over a bounded subset of the three-dimensional parameter space. For each setting of the parameters, we generate synthetic data with known transcribed segments. The segmentation algorithm is run and the settings of the parameters are scored by the fraction of the transcribed genomic region (which was synthetically generated, so the "true" segments are known) that was recovered by local transcript models produced from the segmentation algorithm. Those parameters are chosen which maximize this fraction recovered. The noise parameters of the synthetic data generation process were estimated from a limited number of hand-annotated genomic regions (both transcribed and background regions).

3.4.2 Evaluating the SPLIT Algorithm

Both the simple and hybrid segmentation algorithms were implemented as Java modules, and run on the data described in 5 as well as synthetically-generated data. Taking the optimal settings of the free parameters from Subsection 3.4.2, we can evaluate the performance of SEG and hRMA on two examples of synthetic data. In these examples, random examples are generated and then the SEG and hRMA algorithms are run on the results. The predicted segment boundaries, and transcript intensities, are then compared against their true values and the distribution of their errors is plotted.

Synthetic Data

Evaluating the segmentation algorithm using synthetic data reveals its overall accuracy, as well as its difficulty in recovering the accurate $5'$ end of a transcript in high-noise and low-transcript-intensity regimes. In Figure 3-5, we show an example of synthetically-generated data (Part A) and an evaluation of the SPLIT algorithm's ability to recover segment identities over the same region at differing levels of noise. Here we have generated identically-spaced artificial transcripts over a 1000-bp long region of sequence, tiled at a density comparable to an authentic yeast microarray (50 base pairs between probes). We generate these transcripts for different levels of noise, where the noise scale is measured by the variance of a Gaussian distribution which is randomly added to the log-intensities of the artificially generated probes. For each level of noise, we discover segments using the SPLIT algorithm, and map their "coverage" across the window in which the data is located. This is then repeated 1000 times for each noise level. Part (B) of Figure 3-5 shows the fraction of the artificial replicates, at each level of noise, in which each region of the window is contained within a called segment. For the levels of noise that we tested, the accuracy of the segmenter for the rightmost two segments is high, $> 99\%$. The variation comes in the ability of the segmentation algorithm to correctly identify the leftmost ($5'$) segment, especially at higher levels of noise. This reflects the trouble that our segmentation algorithm will have with identifying the 5′ end of transcribed regions which descend "into the noise," a tradeoff induced by the experimental protocol of the yeast data we have chosen to analyze.

Comparison with *David et al.*

We can also compare our algorithm by running it on actual microarray data in yeast, and comparing it against a published set of tiling microarray transcription experiments also in yeast [11]. Figure 3-6 shows our segmentation of yeast microarray expression data, side-by-side with the corresponding data from David et al. Three indicated regions are marked for comparison. The first region, A, is a location where the `SPLIT` algorithm calls a segment but the segmentation of David et al. is inconsistent and too short. The second region, B, shows a location where both `SPLIT` and the segmentation of David et al. are able to accurately capture the same, long transcript. The third region, C, shows an example of a 5′ transcript end which is recovered by the David et al. data but is (likely) missed by the `SPLIT` algorithm due to the low intensity and surrounding noise.

In Figure 3-7, we show a scatter plot of region intensities between our yeast microarray dataset and the David et al. dataset. To the left axis of the graph are the marginal David et al. intensities, split by the `SPLIT`-induced label (transcribed or not). The Picard and Huber segmentation method relies on an intensity threshold, either determined post hoc [48, 26, 11] or learned as a parameter of the global segmentation [49], in order to differentiate transcribed from background segments. Such a threshold would be manifest as a boundary on the y-axis of Figure 3-7; the area of the blue histogram *above* that threshold, or the red histogram *below* the threshold, would be those regions at which our algorithm (`SPLIT`) and the Picard/Huber segmenter would disagree on the segment identity.

3.5 Probabilistic Models of Differential Expression from Segmented Data

Once we have segmented the microarray data, the segment labels provide the required *array design* input to an RMA-style normalization and comparison algorithm: $I_t = \{i : s_i = t\}$. However, our segmentation algorithm includes the ability to fit multiple local models to a particular segment, each of which may include additional probe-level covariates. Furthermore it would be useful to have the ability to make fully probabilistic evaluations of certain statements (such as the probability of the differential expression of a particular inferred transcript between two experiments). Neither of these goals is well-supported by the existing implementations of RMA, most of which rely on the median polish algorithm and other ad hoc (but empirically useful) optimizations.

For the purposes of this thesis, I have re-implemented the RMA model as a fully probabilistic hierarchical linear model. The hierarchical model, designated `hRMA`, models transcript-specific (that is to say, segment-specific) parameters as drawn from a common distribution. This formulation allows for partial pooling between estimates of transcript expression levels, allowing information sharing between estimates of the levels for separate probes and transcripts (for robust estimation despite outliers) as well as handling missing values through standard probabilistic methods (integrating over their possible values).

$$\begin{aligned} y_{ij} &= \alpha_i + \gamma_{sj} + \beta \mathbf{x}_i + \epsilon_{ij} \\ \epsilon &\sim \mathcal{N}(\cdot; 0, \sigma_y) \\ \alpha_i &\sim \mathcal{N}(\cdot; 0, \sigma_\alpha) \\ \beta_i &\sim \mathcal{N}(\cdot; \mu_\beta, \Sigma_\beta) \\ \gamma_{sj} &\sim \mathrm{Exp}(\cdot; \lambda_\gamma) \end{aligned} \tag{3.15}$$

3.5.1 Evaluating the `hRMA` Model

A method for evaluating hierarchical linear models such as this is through the use of a Markov-chain Monte Carlo (MCMC) sampling method. Software packages such as `BUGS` and `UMACS` provide the ability to express hierarchical models in a flexible domain-specific language (DSL), and then to estimate values for the variables in the model through the use of iterative sampling algorithms.

Instead of using these pre-packaged third-party software packages, we implemented a custom Java interpreter for evaluating hierarchical probabilistic models. Models are expressed in an embedded Java format, as Java classes whose structure reflects the model structure itself. The interpreter samples from the distribution specified by the model using Gibbs sampling; a symbolic expression for the conditional distribution of each individual variable is calculated when the model is specified and resampled using a Java implementation of slice sampling. The interpreter returns samples as instances of the Java class which specifies the model, and thus is easily integrated into the larger GSE system without requiring data import/export and external (non-Java) software. Figure 3-8 shows two differentially-expressed genes identified by the `hRMA` algorithm in the Σ1278b expression dataset (described in detail in Chapter 5). ERG13 is a component of the ergosterol biosynthesis pathway and PHO84 is a phosphate transporter; both are known to be related to Σ1278b-specific function [13]. In Figure 3-9, we diagram the breakdown of expressed and differentially expressed regions in Σ1278b and S288C, as determined by the `SPLIT` and `hRMA` algorithms. The `SPLIT` and `hRMA` algorithms, used in tandem, are able to discover functionally important regions of transcription and differential transcription between two closely related strains of yeast.

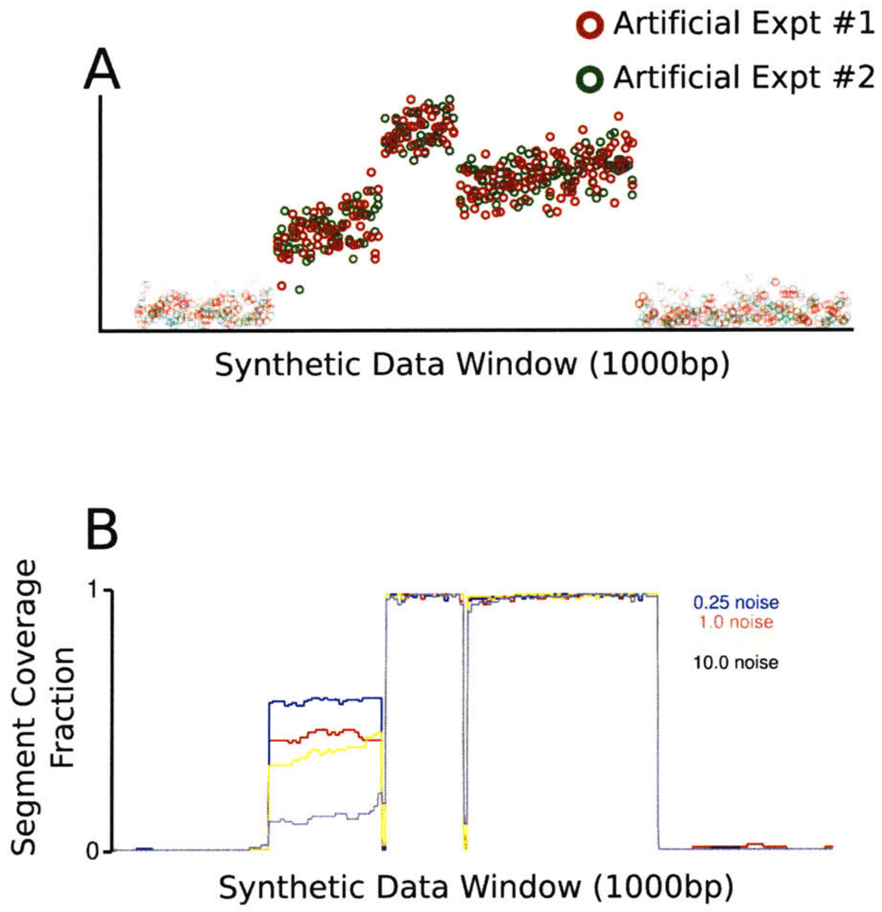


Figure 3-5: Synthetic Segment Recovery
Estimating the probability of segment recovery, per-base, from synthetically generated data with variable levels of noise. Segment recovery varies with the relative level of noise and segment intensity. Furthermore, the 5′ end of segments are least likely to be accurately recovered under low signal-to-noise ratios; this reflects the primary disadvantage of an experimental protocol which induces a 3′-to-5′ falloff in signal intensity over transcripts.

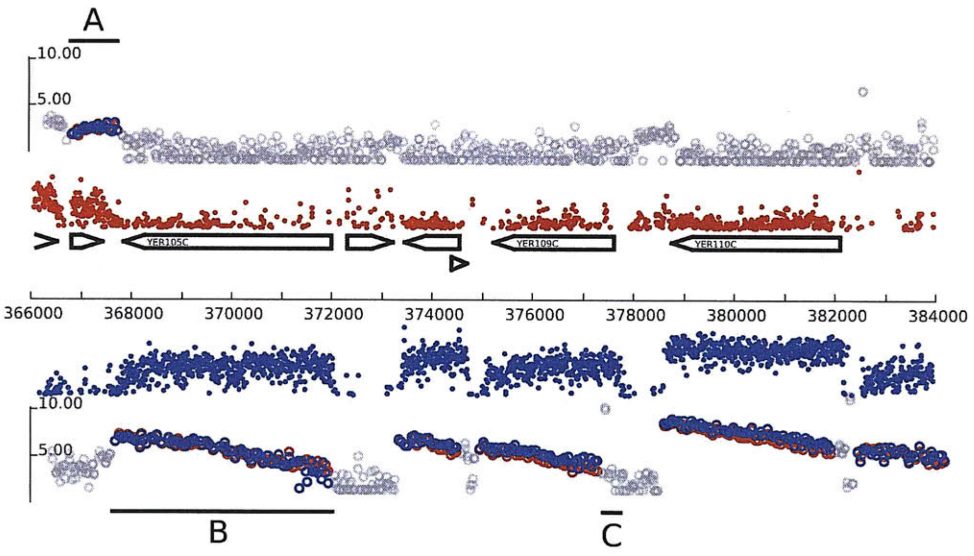


Figure 3-6: Comparison with 3rd Party Dataset: David et al.
Transcribed segments called by the SPLIT algorithm are compared against data from David et al. Data in the top and bottom-most tracks are segmented by SPLIT; solid red and blue points indicate probes in transcribed segments, and gray points are probes in flat (noise) segments. The second and fourth tracks (red dots above the genes and blue dots below) show strand-specific probes from the David et al. dataset. Three regions of comparison are indicated with black bars and letters A-C.

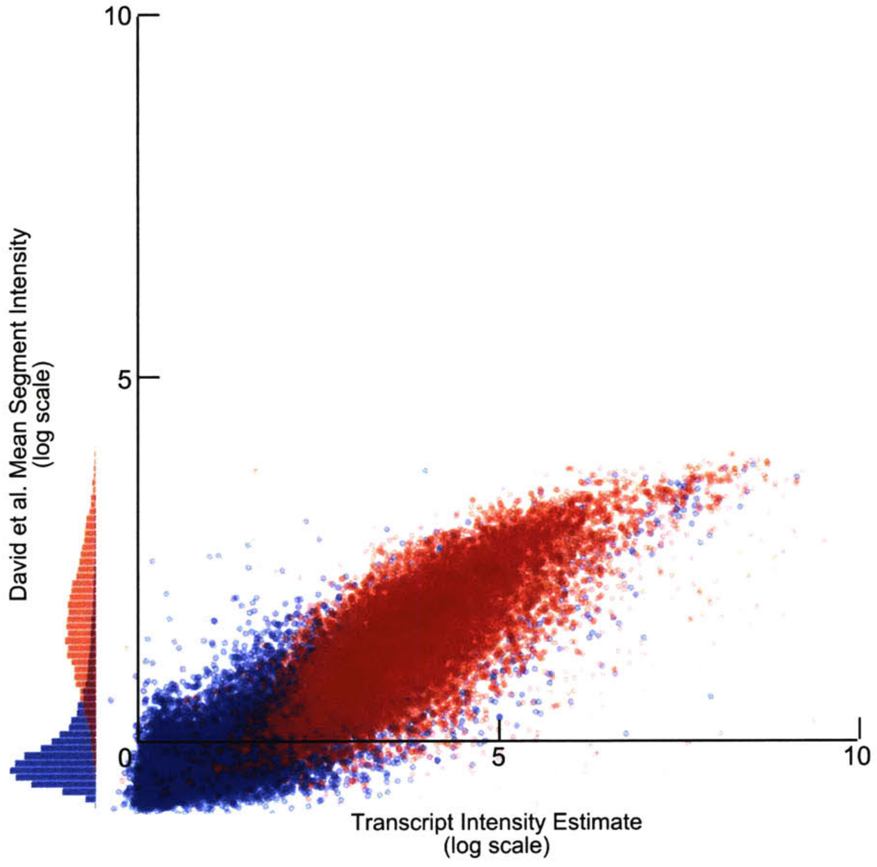


Figure 3-7: Relative Intensity Comparison: David et al.
Each SPLIT-discovered segment is represented by a dot: red for those TRANSCRIBED-labeled segments, and blue for background segments. For each segment, the predicted SPLIT intensity (γ_{sj}, where j indicates the mat-α experimental dataset) and mean intensity from the David et al. microarray are plotted as the x- and y-coordinates. The marginal distribution of the y-coordinates are shown to the left.

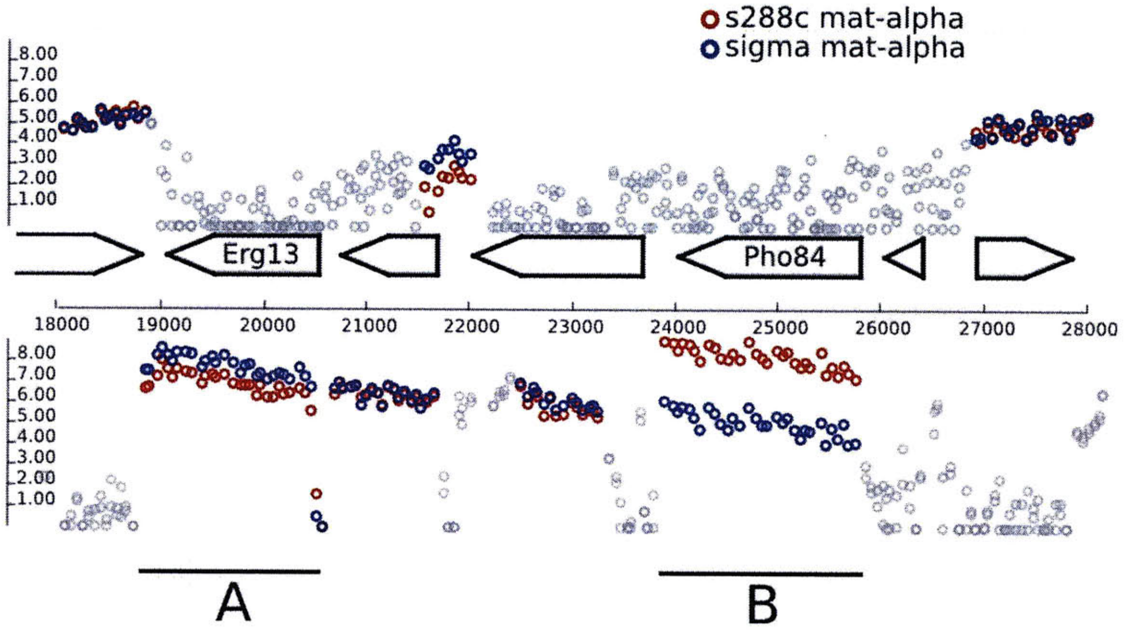


Figure 3-8: Differential Transcription in Σ1278b: ERG13 and PHO84
SPLIT recovers two regions of differential expression, over the genes Erg13 and Pho84. Pho84 is up-regulated in S288C, while Erg13 is up-regulated in Σ1278b.

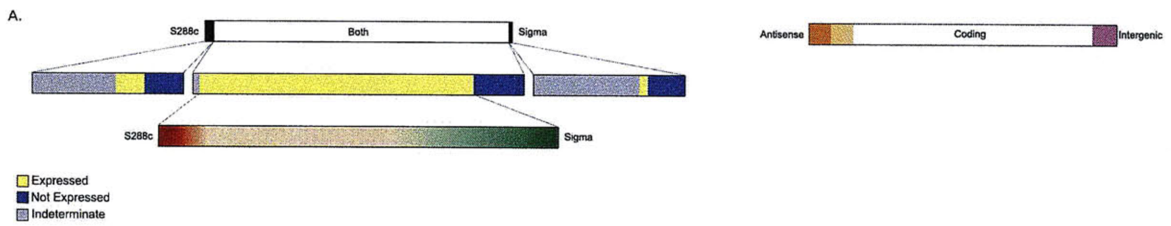


Figure 3-9: Global Differential Transcription in Σ1278b
Bar charts showing the expression of coding and non-coding segments (as determined by overlap with gene annotations in S288C), and the differential expression of those segments that are expressed [13].

Chapter 4

Computational Recovery of Overlapping Transcripts

Segmentation-based analyses of tiling microarray data suffer from the problem of spatial averaging. To measure a the expression of a single gene, a single probe with a sequence unique to that gene would be designed for the array, and hybridization to that probe would be measured during the experiment. If multiple distinct transcripts overlap the probe, however, then the probe's measurement will display an intensity that is a function of the abundance of all the transcripts averaged together. It would be impossible, from a single probe measurement, to recover the relative abundance of these transcripts (or even to resolve the number of contributing transcripts at all). This averaging effect for single probes on a microarray means that gene-centric expression microarrays are completely insensitive to the issue of spatial resolution and overlap, and that tiling microarrays require a more careful analysis than segmentation for the discovery of overlapping transcripts.

The high-resolution measurements of tiling microarrays and high-throughput sequencing (RNA-Seq) offer a potential solution to the problem of spatial resolution. Tiling microarrays use a number of spatially arrayed probes to collect parallel measurements of the transcripts which match or overlap them. High-throughput sequencers gather enough reads, and map enough hits, that the spatial resolution of the changes in the number of hits mapped is often only a few base pairs. Both tiling microarrays and sequencing experiments would be able to resolve the exact boundaries of a single unique transcript, by noting where the probe intensities or read densities appaer to fall.

While these experimental and analytic methods can help resolve the problem of *spatial* averaging, there are two kinds of averaging that they *cannot* correct for: temporal and population averaging. Temporal averaging refers to the fact that the experimental sample is not an instantaneous snapshot of the transcriptional activity in a cell – rather, it is a time-average of that activity. Furthermore, the cell itself is a repository for transcripts which have not themselves been marked and broken down yet and these traces of past transcription will show themselves in the sample as well. Population averaging means that the samples themselves are not composed of a single cell, or even necessarily a completely homogeneous group of cells. Different sub-populations of cells within the sample may give rise to different transcriptional behaviors; but these behaviours are, again, averaged together and measured simultaneously on the array or by the sequencer.

The problem of spatial resolution for multiple, overlapping transcripts is still not solved by standard analysis methods on either tiling microarrays or sequencing experiments. Figure 4-1 shows an artificial example of how two overlapping transcripts (orange and blue arrows) might appear on a tiling microarray dataset (indicated as red probe intensities). While the probes of the array generate sufficient resolution to acccurately determine the start of the leftmost transcript and the end of the rightmost transcript, they are unable to determine *on their own* the actual spatial relationship of the two transcripts to each other. An accurate segmentation analysis of this region could produce the internal breakpoints that indicate the locations of the start or end of a transcript; however the segmentation analysis only provides us with a spatial partition of the genome, and it too fails to produce an analysis of this transcribed regions into overlapping transcripts.

4.0.2 Two Insights into Overlapping Transcript Detection

The work of David et al. [11] identified regions of "complex segmentation," where multiple segments of transcription were found in adjacent (same strand) configurations to each other. However, the analysis of transcription does not end by identifying complex or unanalyzable regions; as suggested in the previous section, we would like to dissect such regions into coherent, overlapping transcripts. But how are we to tell apart, for instance, three tandem non-overlapping transcripts from two tandem overlapping transcripts?

Our algorithm, `STEREO`, is founded on two fundamental insights into way that overlapping transcripts manifest in the probe intensities (or read densities, from a sequencing experiment) along the genome. These two insights, which we respectively term the *additivity* and *differential* insights, will allow us to discover regions of same-strand overlapping transcription.

This first insight we have into the detection of overlapping transcripts from tiling microarrays comes from our assumption that multiple transcripts overlapping a probe will produce an intensity at that probe that is somehow the "sum" of the abundances of the transcripts (although we assume in this work that this relationship is linear, in general the intensity of a probe is a possibly-nonlinear function of the abundance of the transcripts which overlap it: $y_{ij} = f(A_j, B_j)$ where A_j and B_j are the abundances of two overlapping transcripts). Figure 4-1 shows an example of additivity in a cluster with three segments: A, B, and C. We might guess that these three segments are explained by the pattern of two transcripts, Orange and Blue, that are drawn below the cluster. If these two transcripts are the true pattern which produced this cluster, we can infer (from the expression of segments A and C alone) estimates of intensities for each transcript; the estimate at the $3'$ end of segment C is the expression level of the orange transcript, while the estimated slope of segment A will be used to interpolate a $3'$ expression intensity for the blue transcript. If these are the true transcripts, then the probes in segment B should display the *sum* of the falloff-interpolated intensities of the two transcripts. Our first insight into the detection of overlapping transcription is that segments whose estimated intensities seem to be the sum of the intensities of neighboring transcripts are reasonable candidates for explanation by a smaller number of overlapping transcripts.

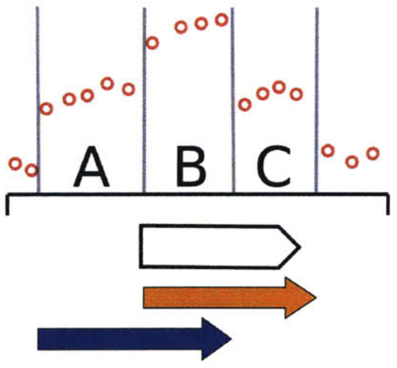


Figure 4-1: Overlapping Insight #1: Additivity
The orange and blue transcripts might explain the cluster of segments A, B, and C if the intensity of segment B is the sum of the intensities implied by segments A and C.

The second insight into overlapping transcription comes from differential expression of segments between two comparable experiments. If we have multiple experiments, measured on the same array platform, for which we expect that some of the transcripts are changed in abundance (but not in location or spatial extent), then we can leverage this second insight into the accurate discovery of overlapping transcripts that are differentially expressed. Figure 4-2 shows an example of this phenomenon on the same three-segment/two-transcript arrangement. If the orange and blue transcripts are the true explanation for the three segments, then transcripts which are themsevles differentially expressed between experiments should produce spatially-coherent patterns of differential expression in the probes and segments they cover. Conversely, if segment A was the only segment to be differentially expressed (or if segments A and B were differentially expressed with *different* changes), then this would argue against the combination of Orange and Blue as a joint explanation of this transcription in this cluster.

In this chapter, we present the `STEREO` algorithm, which stands for "Simple Transcript Enumeration for the Reconstructionof Expressed Overlapping transcription."

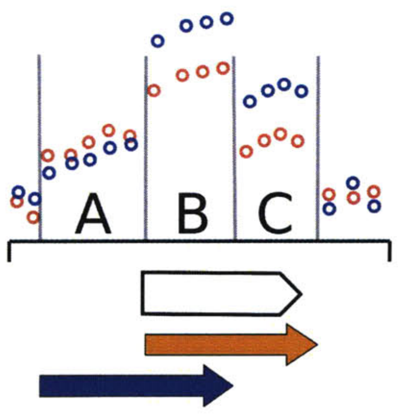


Figure 4-2: Overlapping Insight #2: Differential Transcription

STEREO is an enumeration-based method with a model of transcript additivity at its core, which takes as input a segmentation of tiling microarray data and produces the locations and relative abundance estimates of transcripts which explain the transcribed regions. The algorithm relies on the prior segmentation analysis to provide it (through the breakpoints and segment identities) the locations and boundaries of potential transcripts. Unlike previous computational analyses of tiling microarray data, the algorithm allows itself to enumerate, fit, and report transcript arrangements containing overlapping transcripts as simpler explanations for complex transcribed regions. We believe it is the first computational method for the identification and re-assembly of overlapping transcribed regions on a genome wide scale.

4.1 Prior Work

Analysis of tiling microarray data by segmentation was originally used for the analysis of comparative genomic hybridization [48, 59]. Picard et al. described the first dynamic-programming based segmentation algorithms for discovering regions of copy number variation in array-CGH experiments [48]. They later extended their algorithm to provide automatic labeling of segments using a hybrid dynamic programming/expectation maximization approach [49]. These methods derive their computational efficiency from the fundamental assumption that the segments they identify form a non-overlapping partition of the genome into spatially coherent intervals, an assumption which allows the use of dynamic programming approaches to discover optimal segmentations.

Tiling microarrays are also used to measure the transcription of genomic regions, and segmentation algorithms were similarly adapted to uncover consistently transcribed regions in those datasets [52]. Huber et al. adapted the segmentation algorithm of Picard to identify transcribed regions from tiling arrays [26]. This method was then used in David et al., which published the first tiling microarray study of transcription in yeast [11]. One additional feature of tiling microarrays was their ability to discover strand-specific transcription through the use of strand-specific probes and experimental protocols which preserved the strand-specificity of the sample. The array results of David et al. were strand-specific, and so were able to identify regions of opposite-strand overlapping transcription.

Microarrays are not the only method for analyzing transcription on a genome-wide scale and in an unbiased manner; sequencing of cDNA has been a standard way to identifying unknown transcripts. Miura et al. sequenced expressed cDNA tags to produce a catalog of $5'$ and $3'$ transcript end-points throughput the yeast genome [42]. Sequencing measures single transcripts (and not transcribed regions) directly, and therefore can give information about the structure of transcript overlaps, starting, and ending points assuming that the read length is long enough relative to the transcript lengths.

The use of new, high-throughput short read sequencing machines to investigate transcription has led to the recent adoption of RNA-seq as a measurement of genome wide transcription [39, 64]. RNA-seq experiments sequence fragments of transcripts which are randomly selected from the sample. Nagalakshmi et al. provided the first strand-insensitive view of transcription through RNA-seq in budding yeast [44]. These results have been extended in a strand-specific manner in related strains of yeast by Wilhelm et al. [63]. Unlike traditional sequencing, which produces longer reads, these unpaired-end short read sequencing techniques are unable to give us a full picture of the transcripts from which they were taken and suffer from the same problem of transcript mixture as microarrays. Some sequencing protocols produce reads which are insensitive to the strand of the underlying transcript, requiring that downstream computational analyses include strand-differentiation as one of their goals [45].

Transcription itself has traditionally been associated with individual genes [20]. Genes as units were identified by the sequencing and mapping of expressed sequence tags (ESTs), or through analysis of sequence elements and conservation patterns unique coding regions. From these distinct annotations of individually transcribed genes, microarray designs to probe either the expression of the gene or (through experimental protocols such as ChIP-Chip) events in the proximal "promoter" regions immediately upstream of the gene's annotated start site [17].

Transcription *between* genes (so called intergenic or non-coding transcription) is a well-known phenomenon [47, 8, 21, 27, 62, 32, 58]. With the advent of modern high-throughput sequencers, the ability to detect these intergenic transcripts on a global scale became feasible. Sequencing of transcription uncovered transcripts which did not correspond to known open reading frames or genes. Localized collections of hits that did not spatially overlap known gene annotations could be consistently identified. Microarrays with probes specific to these transcripts were designed and used to probe their expression and regulation in parallel with the genes themselves. Tiling microarrays have also been used to detect these same noncoding transcripts. Some of the transcripts, the "cryptic unknown transcripts," were determined to be quickly degraded by the cell's machinery itself, while other noncoding transcripts were shown to be more stable [29, 38, 12].

A second extension to the classical measurement of gene expression was the attempt to measure the existence and relative abundance of different splice-forms of single genes. A gene may be multiply transcribed, or its transcript may be edited into several different transcripts to be translated into protein. If these transcripts are separated, purified, and then either hybridized to an array or input to a sequencer, then we might expect that different splice-forms will lead to different intensities or hit densities at different portions of the gene itself. Several computational methods have been deployed to attempt to recover the relative abundance of different splice-forms given the spatially ambiguous data from microarrays or sequencers [67]. These methods were combined with the advent of tiling array technology, which could simultaneously screen for noncoding transcription.

Although the sense transcripts of genes are important for understanding which proteins are produced within the cell, transcription over the antisense gene strand has also been shown to play an important regulatory role in the transcription of genes through transcriptional interference. The presence of an antisense transcript in yeast has been shown to regulate how the cells control their shape and behavior [24]. This presumably occurs through the interaction of the antisense transcript with the cellular machinery which promotes transcription of the sense transcript. Strand-specific tiling microarray data in yeast has revealed a significant number of antisense transcripts.

Transcripts which overlap on opposite strands are not the only means of overlapping regulatory interaction: several recent studies have also shown that overlapping transcripts on the same strand can play mutually-regulatory roles through interference. In yeast, Martens et al.[40] examined the role of two separate transcripts present at the SER3 gene – a coding transcript, and a second transcript (so-called SRG1) which was initiated from a cryptic upstream promoter and which has an inhibitory effect on the expression of the SER3 coding transcript. In humans, the work of Martianov et al.[41] described a similar system in which a "standard" and an alternate upstream promoter produced transcripts which appeared to have inhibitory effects on each other. These results show that a careful, spatial analysis not only of transcription but of its overlapping characteristics is important for the understanding of which transcripts are produced, and how that transcription is regulated, throughput the cell.

4.2 Notation

The modeling and analysis of overlapping transcripts from microarray or other datsets will require extending the notation outlined in Section 3.2 with new terms and

i, j, s, t	probes i, experiments j, segments s, transcripts t
x_i	genomic location of probe i
y_{ij}	intensity of probe i in experiment j
$\mathbf{5'}_{[s,t]}, \mathbf{3'}_{[s,t]}$	$5'$ and $3'$ ends of segment s or transcript t
$\lvert x - x' \rvert$	linear distance along the genome, in bp
$\mathbf{t}_s$	type of segment s
θ_s	parameters of segment s
δ_{it}	the distance $\lvert x_i - \mathbf{3'}_t \rvert$ from probe i to the $3'$ end of transcript t
T_i	set of transcripts that overlap probe i
γ_{tj}	intensity of transcript t in experiment j
λ_t	3' log-linear slope of transcript t

Table 4.1: Array, segmentation, and transcript notation summary.

variables for transcripts and other derived array concepts. We begin by outlining some basic notation for arrays and transcripts, shown in Table 4.1. Figure 3-2 recalls that notation; i, j, x, and y are all assumed to be given as part of the microarray experimental data, while s, Θ, $\mathbf{5}$, and $\mathbf{3}$ are the output of the segmentation algorithm itself.

For geometric descriptions of locations along the genome, we will use two terms: points and intervals. A point will be a location of a single nucleotide in a genome assembly. Probes map to a genome assembly as point locations, based on the center of the interval to which their sequence is uniquely mapped. Intervals are convex subsets of the genome, single coherent loci specified completely by start and end positions.

A breakpoint set $B = (b_1, \ldots, b_N)$ is an ordered list of genomic locations which partition the genome into a set of non-overlapping intervals called segments. A segmentation is a set of segments which partition a complete genome. For a given set of breakpoints B, we use $\mathbb{S}_B$ to indicate the segmentation defined by those breakpoints. If $s \in \mathbb{S}_B$, then s is a genomic interval whose endpoints ($\mathbf{5'}_s$ and $\mathbf{3'}_s$) are consecutive elements of the list B. A segmentation algorithm assigns each segment a type $\mathbf{t}_s$ and a set of parameters θ_s which provide a local description of the probe values within that segment.

A transcript is a genomic interval, characterized by its start and end points $\mathbf{5'}_t$ and $\mathbf{3'}_t$. It is a single, coherent message transcribed from the genome in one or more cells. It may be edited or it may be present in an unedited form, in which case it will appear as an interval when matched to the genome which produced it. Overlapping transcripts will produce complex regions of transcription. Transcribed regions are sections of the genome which may form part or all of a single mapped transcript or multiple adjacent and overlapping transcripts.

4.3 An Additive Model of Transcription

One thing which determines how well overlapping transcripts can be inferred from microarray data is the *additivity* of the probe intensities: if a single probe is covered by two overlapping transcripts, how does the probe observation reflect the concentration of those transcripts? In the Chapter 3, we outlined a model (Equation 3.3) where the expression of each probe was dependent on the expression of a single underlying gene. For our segmentation analysis we extended this model with a falloff term, $\lambda\delta_{is}$, which modeled the log-linear drop in probe intensity from the $3'$ to the $5'$ end of a transcript, but the basic model remained the same: one gene (or transcript) for each probe.

The presence of *overlapping* transcripts means that we must relax this assumption; it is possible for a single proble to reflect the abundance of several transcripts which all overlap it. Furthermore, the falloff term will be a composite as well since the distance δ_{it} from the probe i to each overlapping transcript t will likely be different for each different transcript.

What is required is a model of the "additivity" of transcript abundances for probes on the array. In a particular experiment j, a probe's intensity will be a function of the intensities of the transcripts which overlap it; if probe i is overlapped by transcripts A and B, we would write: $y_{ij} = f(A_j, B_J)$ for some function f. But what is to be the form of function f?

In this thesis, we will assume that the microarray measurements are in the "linear range" – that is, that the (*non*-log) values of the probes are linar functions of the abundances of the transcripts.

$$e^{y_{ij}} = \sum_{t \in T_i} e^{\gamma_{tj} + \lambda \delta_{it}} \tag{4.1}$$

This is an assumption, and it could be relaxed to produce a more realistic representation of the function f between probes and abundances of the overlapping transcripts T_i. Equation 4.1 can be rewritten as a log likelihood function for a probe, $\mathcal{L}_{ij}(\gamma_{tj}, \lambda, \sigma) = \log P(y_{ij} | \gamma_{tj}, \sigma^2)$, and we can use that to define a log likelihood function for the complete set of probe observations in an arrangement:

$$\mathcal{L}_A(\Gamma) = \sum_{j \in \mathbb{J}} \sum_{i \in C(A)} \mathcal{L}_{ij}(\gamma_{tj}, \lambda, \sigma^2) \tag{4.2}$$

The variable $\Gamma = \{\lambda, \sigma\} \cup \{\gamma_{tj} : t \in A\}$ is the complete set of parameters for an arrangement A.

4.4 STEREO: Reassembly of Overlapping Transcripts

We present a new algorithm, STEREO, for the computational analysis of overlapping transcription. STEREO is organized into two phases. The first phase implements segmentation and discovers genomic intervals which are transcribed in one of the input experiments. The identified genomic intervals are classified into transcript or background classes using both observed probe intensities and the $3'$ to $5'$ transcript intensity fall-off caused by reverse transcriptase processivity. The second phase, transcript reconstruction, resolves this labeled segmentation into consistent arrangements of explanatory RNA transcripts. STEREO performs a combinatorial search of all possible transcripts given the constraints of transcript additivity and differential expression to yield a segmentation.

The identification and quantification of overlapping transcripts proceeds in three steps.

1. Segmentation, and Identification fo Clusters: a segmentation algorithm (such

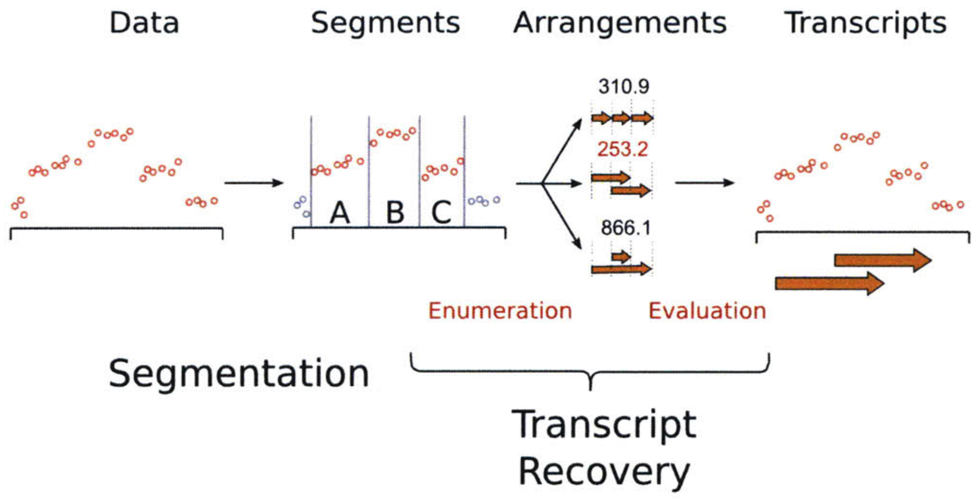


Figure 4-3: Workflow description for segmentation and transcript recovery phases of the STEREO algorithm. The phases operate on sequence in a genomic region tiled by a microarray. The first phase partitions the genome into intervals and assigns each interval a local transcription label. The second phase identifies clusters of transcription. For each cluster, transcript arrangements are enumerated and evaluated, and the optimal arrangement is chosen as the explanation for that cluster.

as SEG) is run on the data, and the breakpoints and segment labels are gathered from that segmentation. These are then used to identify the separate clusters.

2. Enumeration of Arrangements: for each cluster, all possible arrangements are enumerated.

3. Evaluation of Arrangements: for each cluster and for each arrangement that has been enumerated for that cluster, the parameters Γ are estimated for that arrangement in a maximum-likelihood approach. The "fit" of the arrangement to the data within the cluster is evaluated and stored.

Ultimately, for any cluster, the arrangement with the best fit to the cluster's data is returned by the STEREO algorithm as the "explanation" for the transcription within that cluster.

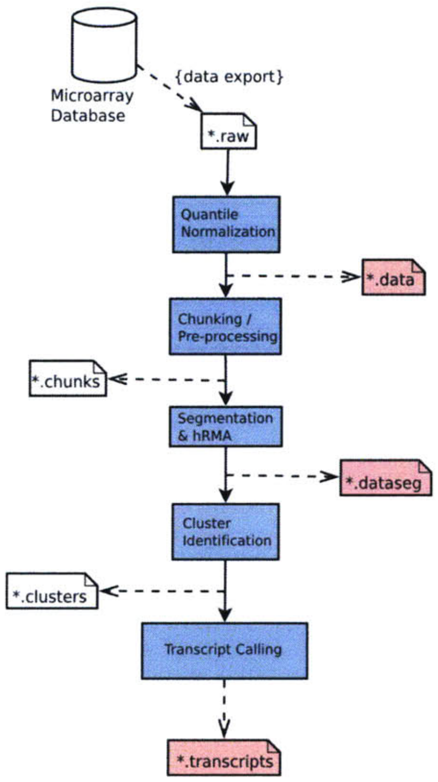


Figure 4-4: STEREO Workflow Diagram
The STEREO algorithm works in four steps: (1) Identify Clusters, (2) Enumerate Arrangements, (3) Evaluate Arrangements, and (4) Choose the Optimally-Scored Arragement.

4.4.1 Evaluating an Arrangement

Starting from the inside and working out, I first describe how we find the parameters Γ for an arrangement and a cluster. The next section will describe how arrangements are enumerated for a cluster. These two steps comprise the core of the STEREO reassembly algorithm.

Equation 4.2 gives the total log-likelihood function for the data within a cluster C depending on the arrangement A. We estimate the parameters for A by maximizing the function $\mathcal{L}_A(\Gamma)$. The complete set of partial derivatives of the $\mathcal{L}_A$ function with respect to each component of the parameter vector is calculated, and then optimization is performed using gradient ascent.

If we let $p_{ij} \equiv \sum_{t \in T_i} e^{\gamma_{tj} + \lambda \delta_{it}}$ and $z_{ij} \equiv \log p_{ij}$, then we can write

$$\frac{\partial \mathcal{L}}{\partial \gamma_{tj}} = \frac{1}{\sigma^2} \sum_i (y_{ij} - z_{ij}) \left(\frac{e^{\lambda \delta_{it}}}{p_{ij}} \right) \tag{4.3}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \frac{1}{\sigma^2} \sum_i (y_{ij} - z_{ij}) \left(\frac{\delta_{it} e^{\lambda \delta_{it}}}{\sum_t \delta_{it} e^{\gamma_{tj} + \lambda \delta_{it}}} \right) \tag{4.4}$$

4.4.2 Arrangement Complexity Penalties

As with segmentation, we do not wish to allow our transcript enumeration and fitting algorithm complete freedom in determining the number and location of transcripts that best fit a cluster's data. The transcript fitting contains a similar model selection problem, as any arrangement with more transcripts necessarily contains more parameters and more degrees of freedom with which to explain the data in a cluster. A completely unconstrained maximum-likelihood solution, as we have outlined so far, will always choose the arrangement with the largest allowable number of transcripts. In the absence of prior beliefs about the expected intensities of those transcripts, the algorithm will always choose transcripts which correspond one-to-one with the segments in a cluster - each transcript will exactly recapitulate the observed data within a segment, and the transcript algorithm will uncover no additional structure above and beyond the segmentation.

We therefore impose three forms of complexity penalty on our `STEREO` algorithm: a prior distribution over the expected intensities of the segments, and a penalty on the number of transcripts and on the number of overlapping segments within an arrangement. Both the penalty on the number of segments and the penalty on the number of segments which contain overlapping transcripts, π_M and pi_V, are affine penalties an arrangement A which contains m segments that overlap in p locations will be assessed the penalty $m\pi_M + p\pi_V$. The values of π_M and π_V were chosen from artificially-generated datasets containing segments whose intensity distribution mirrored that of the S288C mat-α experiment. The same artificial data was used to fit a Gamma distribution to the intensities of the segments, and that distribution was used to penalize the log-likelihood functions maximized above. Assuming that the likelihood of the data is the product of the likelihood *given the intensities* of the transcripts multiplied by the prior probability of those intensities, we can rewrite the equations above as sums of a prior and a likelihood term. Our maximum-likelihood (ML) formulation becomes a maximum a posteriori (MAP) estimation, but all later downstream optimizations are left unchanged.

4.4.3 Assumptions for Overlapping Transcript Detection

To make the detection of overlapping transcripts more precise, we can restrict ourselves in a few significant ways. In addition to the two insights described above, necessary for inferring where likely regions of overlapping transcription may occur, our algorithm also makes several assumptions in order to reduce the size of the problem and the search space we are required to examine.

1. **Segmentation Reliability:** all transcripts begin and end at the boundary of a segment.
2. **Transcription Coverage:** every transcribed segment (local linear model) must be explained by at least one transcript.
3. **Background Removal:** no transcript may explain any background segment (local flat model).
4. **Nonredundancy:** no duplicate transcripts will be examined.

Table 4.2: Transcript Enumeration Assumptions
The `STEREO` algorithm enumerates transcripts for segmented tiling microarray data. We reduce the total number of transcript arrangements that must be examined using a few basic assumptions about how putative transcript calls should match the underlying segmentation and each other.

`STEREO` is a novel algorithm for the identification of regions of overlapping transcription, and for the recovery and quantification of separate transcripts at regions where this overlap is detected. This method relies on the output of a segmentation algorithm (such as `SEG`) as input, and produces a set of possibly-overlapping transcripts along with observed transcript values as output. The method does not rely on the presence of genes, sequence features, or any other markers; however, it does rely on a model of transcript additivity in order to dissect the relative contributions of different transcripts to a region of overlap.

4.4.4 Enumerating Arrangements

The `STEREO` algorithm is an enumerative method for evaluating transcript arrangements and finding the arrangement which best fits a particular cluster. When presented with a transcript arrangement A on a cluster C, optimizing the model log-likelihood given in Equation 4.2 gives us a means of estimating the transcript intensities γ_{tj} for each experiment j and each $t \in A$. The log-likelihood does not specify, however, the method by which (for a given cluster C) we should determine the arrangement A whose parameters we should optimize. The `STEREO` algorithm systematically enumerates a complete set of arrangements over a cluster, evaluates each by its ability to explain (probabilistically generate) the observed data using its optimal parameters, and chooses the arrangement whose optimal parameters best describe the cluster data.

The strategy for identifying the optimal A on a cluster C is one of enumeration with constraints. We systematically generate every arrangement which matches the given cluster, constrained by the assumptions given in Table 4.2. In addition to these basic constraints, we also impose a K *maximum overlap* constraint. For each segment, we can count the number of transcript calls in an arrangement which overlap that segment: we call this the overlap count k_s of the segment s. When enumerating arrangements under a K maximum overlap constraint, we require that no segment's overlap count exceed the bound K.

For a given cluster C with N segments, the `STEREO` algorithm proceeds through the steps outlined in Figure 4-5. The pseudocode for the enumeration itself is given in Figure 4-6.

1. For every possible number $k \in [1, N]$ of transcripts, we enumerate all possible k transcripts over N segments.
2. We remove any arrangement produced by Step 1 which does not satisfy the *coverage* requirement of a valid arrangement – that is, it leaves one or more segments of the cluster "uncovered" by any transcript. For example, the arrangement $A = \{(1, 2)\ (2, 3)\}$ is not a valid arrangement for a cluster with four segments.
3. Next, we apply any additional filters and remove any arrangements which fail those filters (see below for a list of additional filters).
4. Finally, we fit (using the `STEREO` model) a set of transcript intensities to each arrangement A that has passed through all the filters, and we return the arrangement which has the highest likelihood (weighted by a factor which penalizes the number of transcripts used in the arrangement).

Figure 4-5: `STEREO` Algorithm
An outline of the steps involved in the `STEREO` algorithm for fitting transcripts to a cluster of transcribed segments.

4.5 Performance and Testing of `STEREO`

Unfortunately, the performance of the `STEREO` algorithm depends on the number of arrangements it must evaluate, and the number of complete arrangements for a given cluster is exponential in the size of the cluster itself. We can show an upper-bound on this number by listing the transcripts in any given arrangement in lexicographic order, and using that ordering to bound the number of choices possible for all the transcripts. Assume that a cluster of size N segments is given; this implies that it has $N + 1$ breakpoints, which we will number in sequence from $0 \ldots N$. We also assume that we are attempting to arrange M transcripts over the N segments, in such a way that no two transcripts are identical and that the total arrangement is complete (that is, every segment is covered by at least one transcript). Naively, we can choose each transcript's endpoints in $\frac{1}{2}N^2$ possible ways, leading to an $\mathcal{O}(N^{2M})$ bound on the number of transcript choices.

A tighter bound can be achieved by considering the transcripts ordered lexicographically – that is, for each pair of transcripts t_1 and t_2, we induce an ordering on the pair by ordering first on the start coordinate and then on the end coordinate: $t_1 < t_2 \equiv (\mathbf{5'}_{t_1} < \mathbf{5'}_{t_2})$ or $(\mathbf{5'}_{t_1} = \mathbf{5'}_{t_2}$ and $(\mathbf{3'}_{t_1} < \mathbf{3'}_{t_2})$. If we assume that we have written out all transcripts in lexicographic order $t_1 < t_2 < \dots t_M$, then we can arrange their starting and ending points in a grid:

$$\begin{array}{cc} x_1 = 0 & y_1 \in [x_1 + 1, N] \\ x_2 \in [0, N-1] & y_2 \in [x_2 + 1, N] \\ x_3 \in [0, N-1] & y_3 \in [x_3 + 1, N] \\ \vdots & \\ x_P \in [0, N-1] & y_P = N \\ \vdots & \\ x_M \in [0, N-1] & y_M \in [x_M + 1, N] \end{array} \tag{4.5}$$

The first coordinate x_1 *must* equal 0, and some end coordinate y_P must equal N, otherwise we have an arrangement that is not complete. However, without additional restrictions, we are left with $M-1$ start points, each of which has N possible values, and $M-1$ end points again with N possible values. As before, this is a bound of $\mathcal{O}(N^{2M-2})$ for the total number of arrangements.

However, we can restrict our analysis and add an additional constraint: maximum overlap. A maximum overlap constraint is a number K such that no segment is allowed *more* than K overlapping transcripts.

Multiplying the number of options together for an upper-bound, under the K-overlap constraint we again have descending sequence which are multipled together factorially. There are N options for x_1, and $N-1$ options for x_K, and $N-2$ options for x_{2K} – this is the upper half of a factorial sequence, and it continues for $\frac{M}{K}$ factors. Furthermore it is repeated K times, leading to an upper bound on the number of x choices equal to $(\prod_{m=N-\frac{M}{K}}^{N} m)$. Rewriting in terms of factorials, the total upper bound for the number of options on both sides of the table is $(\frac{N!}{(N-\frac{M}{K})!})^2$.

4.5.1 Synthetic Data

We tested the transcript re-assembler on synthetic data generated with an inside-outside pattern of overlapping transcription, as shown in Figure Figure 4-8B. In 1000 randomly-generated regions of synthetic data, the segmenter and transcript re-assembler were able to reconstruct the correct number of transcripts 96% of the time.

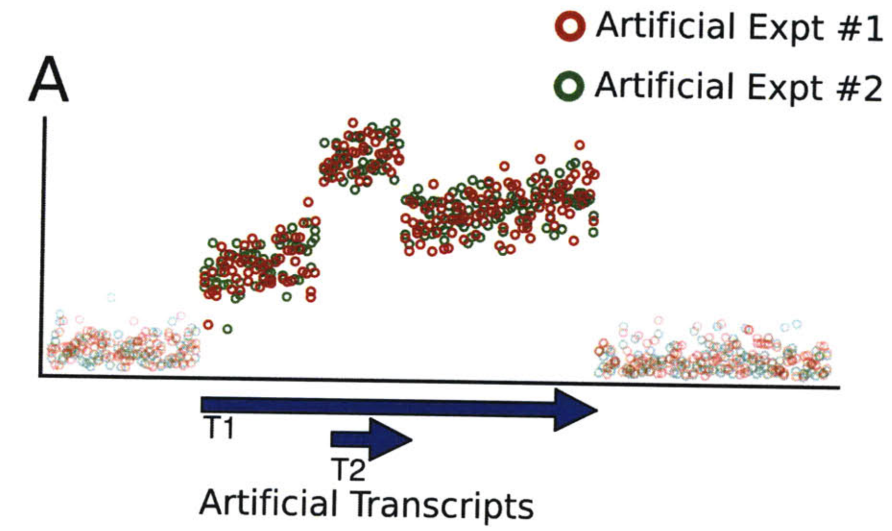
For those instances in which the correct number (2) of transcripts were inferred in the synthetic data, our method was able to able to identify the correct *boundaries* of transcription to within 500 bp 98% of the time, and for these correctly-localized transcripts our intensity prediction was able to recover the original transcript "intensities" with high accuracy.

```
TotalA <- {}
for i = 1:len(C) {
    A <- firstArrangement(i)
    while A != nil {
        TotalA <- TotalA + A
        A <- nextArrangement(A)
    }
}

nextArrangement(A) {
    j <- len(A)
    while lastcall(j) {
        j <- j - 1
    }

    if j ≥ 0 {
        A[j] <- nextcall(A[j])
        for k = j:N {
            A[k] <- A[j]
        }
        return A
    } else {
        return nil
    }
}

lastcall(t) { return start(t) == len(C) - 1 and end(t) == len(C) }

nextcall(t) {
    s <- start(t)
    e <- end(t) + 1
    if e > len(C) {
        s <- start(t) + 1
        e <- s + 1
    }
    return (s, e)
}
```

Figure 4-6: Pseudcode for Arrangement Enumeration

$$
\begin{array}{rc}
x_1 = 0 & y_1 \in [x_1 + 1, N] \\
x_2 \in [0, N-1] & y_2 \in [x_2 + 1, N] \\
x_3 \in [0, N-1] & y_3 \in [x_3 + 1, N] \\
\vdots & \\
x_K \in [0, N-1] & y_K \in [x_K + 1, N] \\
x_{K+1} \in [1, N-1] & y_{K+1} \in [x_{K+1} + 1, N] \\
x_{K+2} \in [1, N-1] & y_{K+2} \in [x_{K+2} + 1, N] \\
x_{K+3} \in [1, N-1] & y_{K+3} \in [x_{K+3} + 1, N] \\
\vdots & \\
x_{2K} \in [1, N-1] & y_{2K} \in [x_{2K} + 1, N] \\
\vdots & \\
\vdots & y_P = N \\
\vdots & \\
x_{M-K+1} \in [\frac{M}{K}, N-1] & y_{M-K+1} \in [x_{M-K+1} + 1, N] \\
x_{M-K+1} \in [\frac{M}{K}, N-1] & y_{M-K+2} \in [x_{M-K+2} + 1, N] \\
x_{M-K+1} \in [\frac{M}{K}, N-1] & y_{M-K+3} \in [x_{M-K+3} + 1, N] \\
\vdots & \\
x_M \in [\frac{M}{K}, N-1] & y_M \in [x_M + 1, N]
\end{array}
\tag{4.6}
$$

Figure 4-7: Counting arrangements under a K-maximum-overlap constraint.



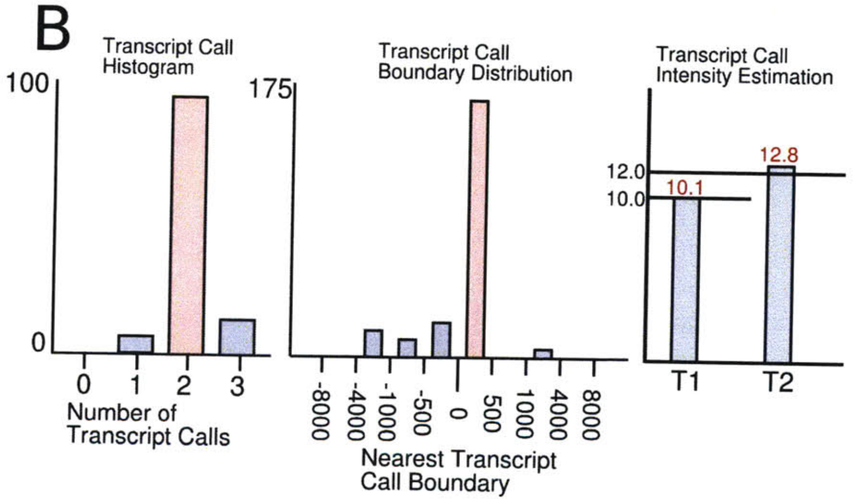


Figure 4-8: Synthetic Transcript Reconstruction
Reconstruction of artificial transcripts is, given the correct bounds as determined by segmentation, able to discover the correct number, overlap, and intensities of the transcripts.

Chapter 5

Computational Detection of Overlapping Transcripts in *Saccharomyces cerevisiae*

In Chapter 4 we presented STEREO, a novel algorithm that discovers cis-regulatory RNA interactions by assembling complete and potentially overlapping same-strand RNA transcripts from tiling expression data. STEREO first identifies coherent segments of transcription and then discovers individual transcripts that are consistent with the observed segments given intensity and shape constraints. Here we show how the use of STEREO allowed us to identify 1446 regions of overlapping transcription in two strains of yeast, including the transcripts that comprise a molecular toggle switch controlling gene variegation. We also discovered 564 transcripts which participated in a same-strand overlapping region with at least one other transcript. These constitute a novel collection of overlapping transcripts which may plausibly exert a regulatory influence on each other.

5.1 Introduction

Evidence has recently emerged from high-throughput expression datasets that overlapping RNA transcripts can play an important role in gene regulation. For example, an antisense transcript can be used to regulate its corresponding sense gene [27, 8]. In budding yeast, a sense/antisense toggle has been shown to regulate the mating type of the cell [24]. The interference of a transcript on the same strand as a coding transcript is also sufficient to play a repressive role in the regulation of downstream genes [40, 41].

Discovering RNA transcript based cis-regulation requires the precise spatial localization of transcripts and the identification of their overlap with other, nearby transcripts. Contemporary algorithms for analyzing tiling microarray identify non-overlapping segments of coherent transcription [48], but they do not attempt to identify the transcripts that generated and potentially span the observed segments. A genomic locus which is multiply transcribed may produce a region of complex segmentation, but no additional resolution of such regions into separate, overlapping transcripts can be provided by dynamic programming or the probabilistic models used by segmentation algorithms.

The STEREO algorithm is the first computational attempt to discover regions of complex transcription from segmentation and to resolve those regions into overlapping transcripts. Overlapping transcripts fall into two mutually-exclusive categories, opposite and same-strand overlap, both of which may be expected to exhibit mutual interference or other regulatory properties. Opposite-strand overlap involves two transcripts are transcribed from complementary DNA strands and whose spatial extents overlap despite their different directions. Sense/antisense pairs of transcripts over the same gene are an example of opposite-strand overlap. Same-strand overlap occurs when two or more transcripts transcribed from overlapping portions of a DNA strand.

We tested STEREO on tiling expression data from two strains of yeast and discovered 1,446 instances of transcriptional overlap. Of these, 564 (39.0%) were overlapping in the same strand, a percentage consistent with previous estimates of alternate promoter usage in known yeast coding regions [42]. Northern blot analysis confirmed a same-strand interaction predicted by STEREO, and STEREO also identified opposite-strand transcripts that are organized into a novel form of molecular toggle switch [7] that controls the state of gene variegaton.

5.2 RNA *cis*-regulation in S288C and Σ1278b

Using an array designed to probe the S288C genome at approximately 50 base-pair resolution, we designed a set of experiments intended to reveal differences in transcription regulation between two closely related strains of *Saccharomyces cerevisiae*: S288C and Σ1278b [13]. Each array had two channels, Cy3 and Cy5, which were used to simultaneously measure the expression in the two strains. In addition to the haploid (mat-α) dataset of [13], we generated diploid expression in rich media with a technical replicate of each experiment. Treating each channel of each array as a separate logical experiment, this design provided us with eight total experiments on which to perform our segmentation and analysis. Data was normalized across experiments using quantile normalization [4].

5.3 `SPLIT` identifies expression using multiple constraints

We used the segmentation and labeling phase of our algorithm to analyze the tiling microarray experiments described in Section 5.2. The `SPLIT` segmentation algorithm was run on each strand of the tiling microarray dataset separately; we identified $14,076$ segments on the Watson strand and $13,792$ on the Crick strand. From the segmentation on the Watson strand, we identified 37.0% of the tiled genome as transcribed, and from the Crick strand we identified 40.3%; taken together, accounting for overlap, we identified 65.2% of the complete genome sequence as transcribed.

The segmentation recovered two noncoding transcripts whose regulatory function is related to their spatial overlap and interference with the production of a downstream coding transcript. In Figure 5-1, we show the locations of two noncoding transcripts, PWR1 and ICR1, that we showed implement a new type of RNA molecular toggle [7]. We also ran an implementation of the Picard segmentation algorithm

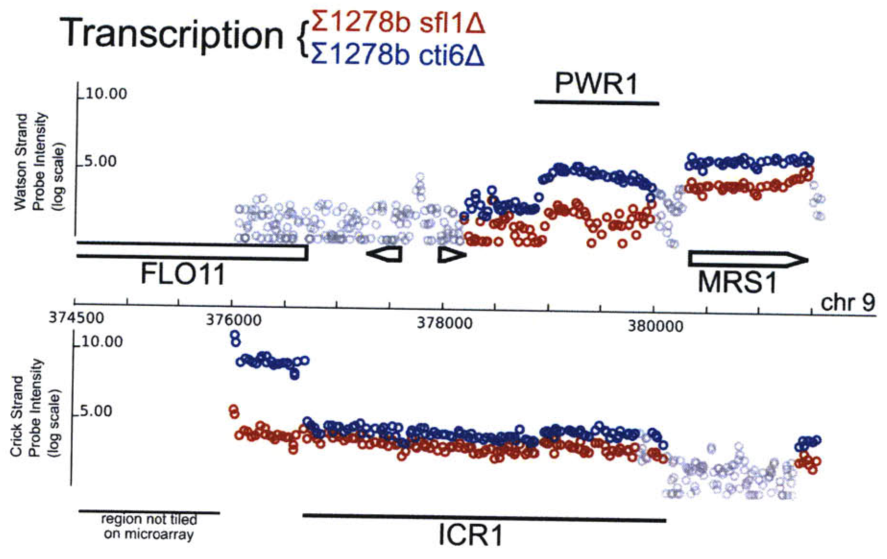


Figure 5-1: We are able to discover noncoding transcription which is known to play a role in the regulation both of a downstream coding transcript (FLO11) and of each other. ICR1 and PRW1 are noncoding RNAs, reported in [7], whose regulatory function is related to their spatial overlap. The segmentation phase of our `STEREO` algorithm is able to find the complete PRW1 transcript and the 3′ end of the ICR1 transcript in the Bumgarner dataset. Regions identified as `BACKGROUND` are shown in grey, `TRANSCRIBED` regions are shown in color. Genes are identified as arrowed boxes. The x-axis is genomic coordinates and the y-axis is log intensity.

on the S288C and Σ1278b dataset. Although this method can be easily adapted to handle multiple experiments simultaneously, it lacks the ability to identify regions with a shape other than a flat regions of transcription; instead, it separates the sloped regions of transcription into "steps" of multiple flat segments. Therefore, the Picard algorithm is unable to handle a key feature of our experimental protocol (the 3′ falloff) and unnecessarily splits single units of transcription into artificially complex sets of segments.

5.4 `STEREO` assembles transcripts from expressed segments

`STEREO` uses an enumeration-based search method to choose the transcript arrangement T which best explains the transcribed regions that are provided as input by the segmentation and labeling phase, as described in Chapter 4. The penalized log-likelihood $\mathcal{L}(\Gamma_T, \lambda_T, \sigma_T) - C(T)$ provides a score by which to evaluate the fit of the transcript arrangement T to the cluster. A complexity penalty $C(T) = \alpha|T| + \beta c_{\text{over}}(T)$ assesses a constant penalty for the total number of transcripts in T and for the number of overlaps $c_{\text{over}}(T)$ in the arrangement. The complexity penalty parameters (α and β) are chosen to optimize transcript discovery against synthetically-generated data.

5.4.1 STEREO Transcript discovery recovers appropriate SER3 and SRG1 transcripts

An example of overlapping transcripts with regulatory interactions in yeast is the SER3 gene and its upstream intergenic transcript, SRG1. The SER3 gene is involved in serine biosynthesis and under repressing conditions its promoter is bound by significant levels of both TATA binding protein (TBP) and RNA polymerase II (Pol II). The expression of a short transcript that runs through the SER3-proximal TATA element is associated with decreased expression of the SER3 transcript itself [40]. Furthermore, a nearly 2 kb read-through transcript starting from the SRG1 TATA element and extending through the entire SER3 gene itself was observed by northern analysis in the same study.

The SER3 and SRG1 genes, and their observed architecture of overlapping transcription, provide a convenient test of our ability to estimate relative intensities of overlapping transcripts. In Figure 5-2, we show that our tiling array data in S288C (red) and Σ1278b (blue) around the SER3 and SRG1 locus. The figure depicts the locations of three overlapping transcripts, shown as orange arrows: one from the upstream SRG1 TATA element extending to the annotated start of the SER3 gene, the second from the SER3 TATA element extending to the end of the SER3 gene, and one 2 kb-long transcript starting from the SRG1 TATA element and extending through the SER3 gene itself.

Using our transcript intensity estimation method we reconstructed relative log-

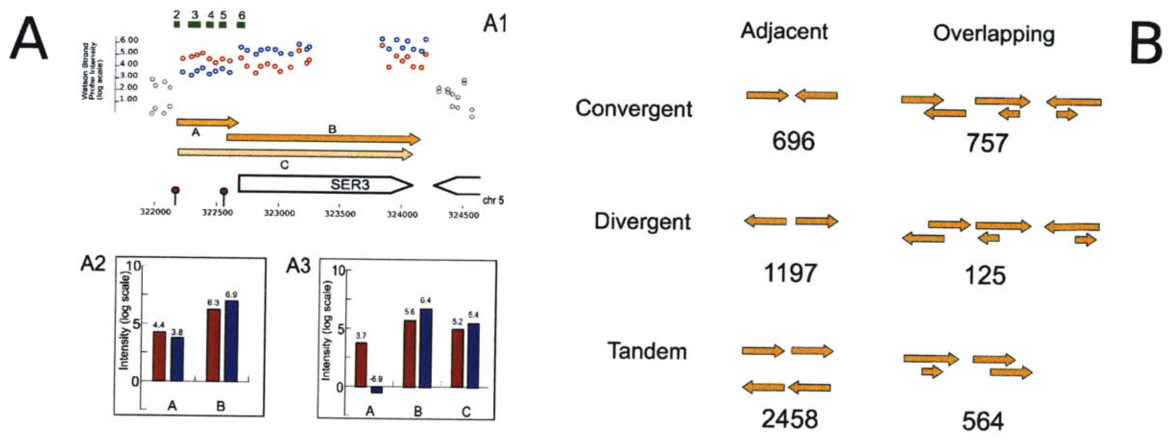


Figure 5-2: **A** Re-construction of transcript intensities at the SER3/SRG1 locus. **A1**. Probes which are included in either the SER3 or SRG1 region and are included in this analysis are displayed in either red (S288C) or blue (Σ1278b). Original probes from Martens et al. enriched for the SRG1 transcript are green. Putative transcripts A, B, and C are shown in orange and TATA elements with red dots. Transcript A corresponds to Martens SRG1 transcript while Transcript B corresponds to SER3 transcript. Transcript C is the "readthrough" transcript Martens detected, extending exactly 2 kb. Transcript intensity analyses were carried out for two arrangements, (**A2**) just the A and B transcripts and (**A3**) all three transcripts. Each transcript has reconstructed intensities for both S288C (red) and Σ1278b (blue) experimental data. **B** Schematics for each category along with the total number of `STEREO` transcripts identified within each category are shown.

intensities of 4.4 and 6.3 for the A and B transcripts respectively; these values are consistent with previously reported concentrations for SRG1 and SER3 respectively [40]. Moreover, the fitted intensities are anti-correlated across cell types, between the two measured strains of yeast. When the SRG1 transcript drops the SER3 transcript rises, consistent with the claim that SRG1's transcription represses that of SER3.

5.4.2 Identification of 1446 overlapping transcripts

`STEREO` resolved the collected S288C and Σ1278b expression datasets into 6609 transcripts. Most transcripts (5233) inferred by our method were strand singletons, covering a single region without a second overlapping transcript. However, our algorithm identified 1446 regions of overlapping transcription, of which 564 were same-strand overlapping transcripts. Figure 5-2 Part B shows a classification of transcript pairs into six categories depending on their relative orientation and overlap, and gives the number of transcript pairs that fell into each category from our dataset.

The segmentation and labeling phase has also been able to uncover overlapping transcript pair predictions which show differential expression between different cell types and strains, and whose variation is consistent with potential repressive regulatory interactions between the overlapping transcripts.

5.4.3 Northern analysis of overlapping predictions

In order to confirm one of our predictions we chose three of the predictions made by our algorithm to test with northern blot analysis. To facilitate northern blot analysis we chose examples to test that had a larger outer transcript with a smaller inner contained transcript that would readily be immediately apparent in the experimental result. In one of the three locations tested northen blot analysis showed same-strand overlapping transcription with transcript lengths matching those produced by STEREO. This validated locus, YCR082W, provides a new example of tandem overlapping transcripts previously unknown in the literature. Instead of reporting overlapping transcripts in this location, an alternate explanation would have been two tandem transcripts aligned head-to-tail; in this case, the transcript discovery algorithm reconstructs the more complex overlap based on our prior distribution over transcript intensities and our belief that higher-intensity transcripts are less likely than lower ones. Zheng et al. have previously attempted to quantify the intensities of multiple overlapping transcripts using a hierarchical Bayesian model [67]. Their approach is limited, however, to the quantification of transcript intensities whose locations have already been specificed from gene annotations or an external datasource. Rochette et al. have reported a set of overlapping transcripts at a genome-wide level in the parasite *Leishmania* [51]. These transcripts were identified by experimental means (5′-RACE) in a genome significantly smaller than yeast, however, and do not represent a comprehensive computational approach to transcript discovery.

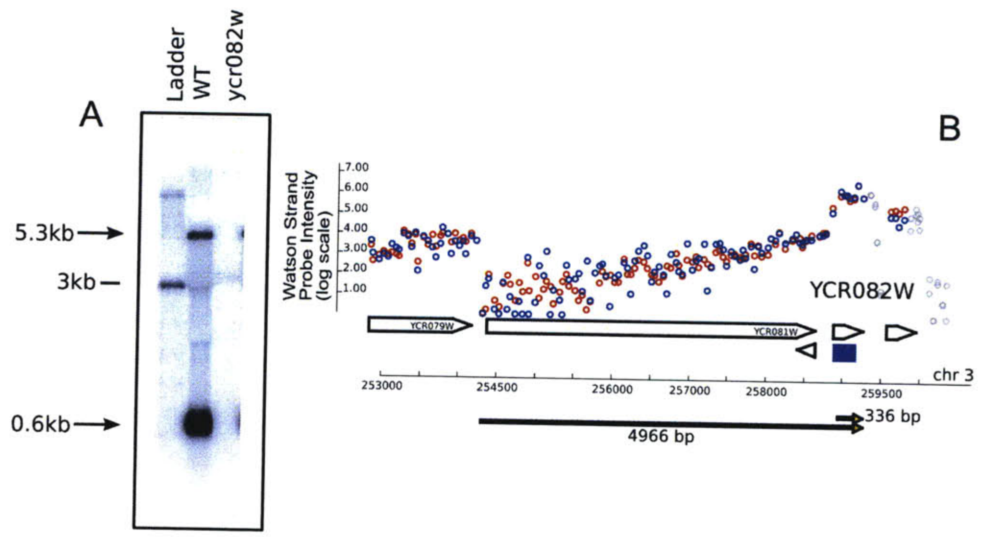


Figure 5-3: Northern analysis was performed at YCR082W to test for the presence of multiple overlapping transcripts. Probes were chosen to cover the first 400 bp of the gene, shown as a blue square. The blot (**A**) shows two transcripts with lengths approximately 5 kb and 600 bp. These transcripts correspond (**B**) to two overlapping transcripts called by the `STEREO` algorithm with lengths of approximately 5 kb and 300 bp. For clarity, only the Watson strand is shown. Probes in `TRANSCRIBED` regions are shown in color for S288C (red) and Σ1278b (blue) data.

5.5 Discussion

We introduced several unique features of our `STEREO` algorithm. In the segmentation phase, we simultaneously incorporated multiple experiments and utilized the slope of the transcription data to identify transcribed segments. In the transcript discovery phase we employed both additive intensity and differential expression to evaluate likely configurations of transcripts.

`STEREO` also has certain limitations. While a 3′ to the 5′ intensity fall off provides a useful constraint, it also makes it more difficult to accurately locate the 5′ ends of long, low-abundance transcripts. In addition, `STEREO` is sometimes unable to separate same-strand overlapping transcripts without differential expression between conditions or strains. In these cases, overlapping transcript calling depends on our prior distributions on transcript intensities. A better understanding of the distribution of transcript abundances will improve the accuracy of our transcript reassembly algorithm. The combinatorial architecture of gene regulation is in part implemented by

RNA based cis-regulation. We are making our set of 1446 candidate interactions available for other investigators.

Bibliography

[1] *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[2] Takeshi Akama, Koichi Suzuki, Kazunari Tanigawa, Akira Kawashima, Huhehasi Wu, Noburu Nakata, Yasunori Osana, Yasubumi Sakakibara, and Norihisa Ishii. Whole-Genome Tiling Array Analysis of *Mycobacterium leprae* RNA Reveals High Expression of Pseudogenes and Noncoding Regions. *Journal of Bacteriology*, 191(10):3321–3327, May 2009.

[3] DA Benson, I Karsch-Mizrachi, DJ Lipman, J Ostell, and DL Wheeler. Genbank. *Nucleic Acids Research*, 35:21–25, January 2007.

[4] Benjamin Bolstad. Probe Level Quantile Normalization of High Density Oligonucleotide Array Data. Technical report, Division of Biostatistics, University of California, Berkeley, December 2001.

[5] Benjamin Bolstad. *Low-level analysis of High-density Oligonucleotide Array Data: Background, Normalization, and Summarization*. PhD thesis, University of California Berkeley, 2004.

[6] Christopher R. Brown and Pamela Silver. Transcriptional regulation at the nuclear pore complex. *Current Opinions in Genetics & Development*, 17(2):100–106, April 2007.

[7] Stacie Bumgarner, Robin Dowell, Paula Grisafi, David Gifford, and Gerald Fink. A Toggle Involving *Cis*-Interfering Noncoding RNAs Controls Variegated Gene Expression in Yeast. *PNAS*, 2009.

[8] Jurgi Camblong, Nahid Iglesias, Celine Fickentscher, Guennaelle Dieppois, and Francoise Stutz. Antisense RNA Stabilization Induces Transcriptional Gene Silencing via Histone Deacetylation in *S. cerevisiae*. *Cell*, 131:706–717, November 2007.

[9] Gal Chechik, Eugene Oh, Oliver Rando, Jonathan Weissman, Aviv Regev, and Daphne Koller. Activity motifs reveal principles of timing in transcriptional control of the yeast metabolic network. *Nature Biotechnology*, 26(11), October 2008.

[10] MA Crosby, JL Goodman, VB Strelets, P Zhang, WM Gelbart, and Flybase Consortium. Flybase: genomes by the dozen. *Nucleic Acids Research*, 35:486–491, 2007.

[11] Lior David, Wolfgang Huber, Marina Granovskaia, Joern Toedling, Curtis J. Palm, Lee Bofkin, Ted Jones, Ronald W. Davis, and Lars M. Steinmetz. A high-resolution map of transcription in the yeast genome. *PNAS*, 103(14):5320–5325, 4 April 2006.

[12] Carrie Anne Davis and Manuel Ares Jr. Accumulation of unstable promoter-associated transcripts upon loss of the nuclear exosome subunit rrp6p in *saccharomyces cerevisiae*. *PNAS*, (9):3262–3267, February 2006.

[13] Robin D. Dowell, Owen Ryan, An Jansen, D. Cheung, Sudeep Agarwala, Timothy W. Danford, Doug Bernstein, P. Alex Rolfe, Gerry R. Fink, David K. Gifford, and Charlie Boone. Genotype to Phenotype: A Comparison of Two Interbreeding Yeast Strains Reveals Complex Genetics of Conditional Essential Genes. in submission.

[14] Natalie C. Duarte, Markus J. Herrgård, and Bernhard Ø. Palsson. Reconstruction and validation of saccharomyces cerevisiae ind750, a fully compartmentalized genome-scale metabolic model. *Genome Research*, 14(7), July 2004.

[15] SS et. al. Dwight. Saccharomyces genome database: underlying principles and organisation. *Brief Bioinformatics*, 5(1):9–22, Mar 2004.

[16] Chandra Erdman and John Emerson. A fast Bayesian change point analysis for the segmentation of microarray data. *Bioinformatics*, 24(19):2143–2148, July 2008.

[17] Harbison et al. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431:99–104, September 2004.

[18] Lee et al. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298:799–804, October 2002.

[19] Ren et al. Genome-wide location and function of dna binding proteins. *Science*, 290:2306–2309, December 2000.

[20] Spellman et al. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, December 1998.

[21] Mohammad Ali Faghihi and Claes Wahlestedt. Regulatory roles of natural antisense transcripts. *Nature Reviews: Molecular Cell Biology*, 10:637–643, September 2009.

[22] Giaever G, Chu AM, Ni L, Connelly C, Riles L, Véronneau S, Dow S, Lucau-Danila A, Anderson K, André B, Arkin AP, Astromoff A, El-Bakkoury M, Bangham R, Benito R, Brachat S, Campanaro S, Curtiss M, Davis K, Deutschbauer A, Entian KD, Flaherty P, Foury F, Garfinkel DJ, Gerstein M, Gotte D, Güldener U, Hegemann JH, Hempel S, Herman Z, Jaramillo DF, Kelly DE, Kelly SL, Kötter P, LaBonte D, Lamb DC, Lan N, Liang H, Liao H, Liu L, Luo C, Lussier M, Mao R, Menard P, Ooi SL, Revuelta JL, Roberts CJ, Rose M, Ross-Macdonald P, Scherens B, Schimmack G, Shafer B, Shoemaker DD, Sookhai-Mahadeo S, Storms RK, Strathern JN, Valle G, Voet M, Volckaert G, Wang CY, Ward TR, Wilhelmy J, Winzeler EA, Yang Y, Yen G, Youngman E, Yu K, Bussey H, Boeke JD, Snyder M, Philippsen P, Davis RW, and Johnston M. Functional profiling of the saccharomyces cerevisiae genome. *Nature*, 418(6896):387–391, July 2002.

[23] Alex Hartemink, David Gifford, Tommi Jaakkola, and Richard Young. Maximum Likelihood Estimation of Optimal Scaling Factors for Expression Array Normalization. In M. Bittner, Y. Chen, A. Dorsel, and E. Dougherty, editors, *Microarrays: Optimcal Technologies and Informatics*, volume 4266, pages 132–140. SPIE International Symposium on Biomedical Optics 2001 (BiOS01), 2001.

[24] Cintia Hongay, Paula Grisafi, Tony Galitski, and Gerry Fink. Antisense transcription controls cell fate in Saccharomyces cerevisiae. *Cell*, 127(4):735–745, 2006.

[25] Zhanzhi Hu, Patrick J Killion, and Vishwanath R Iyer. Genetic reconstruction of a functional transcriptional regulatory network. *Nature Genetics*, April 2007.

[26] Wolfgang Huber, Joern Toedling, and Lars Steinmetz. Transcript mapping with high-density oligonucleotide tiling arrays. *Bioinformatics*, 22(16):1963–1970, 2006.

[27] Thomas A. Hughes. Regulation of gene expression by alternative untranslated regions. *Trends in Genetics*, 22(3):119–122, March 2006.

[28] Trey Ideker, Vesteinn Thorsson, Jeffrey A. Ranish, Rowan Christmas, Jeremy Buhler, Jimmy K. Eng, Roger Bumgarner, David R. Goodlett, Ruedi Aebersold, and Leroy Hood. Integrated genomic and proteomic analyses of a systematically perturbed metabolic network. *Science*, 292(5518), May 2001.

[29] Takashi Ito, Fumihito Miura, and Miyuki Onda. Unexpected complexity of the budding yeast transcriptome. *IUBMB Life*, 60(12):775–781, 2008.

[30] David S. Johnson, Ali Mortazavi, Richard M. Myers, and Barbara Wold. Genome-wide mapping of in vivo protein-dna interactions. *Science*, 316(5830):1497–1502, 2007.

[31] D Karolchik, R Baertsch, M Diekhans, TS Furey, A Hinrichs, YT Lu, KM Roskin, M Schwartz, CW Sugnet, DJ Thomas, RJ Weber, D Haussler, and WJ Kent. The ucsc genome browser database. *Nucleic Acids Research*, 31(1):51–54, 2003.

[32] Mitsuoki Kawano, April Reynolds, Juan Miranda-Rios, and Gisela Storz. Detection of $5'$ and $3'$-utr-derived small rnas and *cis*-encoded antisense rnas in *escherichia coli*. *Nucleic Acids Research*, 33(3):1040–1050, February 2005.

[33] J.H.B. Kemperman. Least Absolute Value and Median Polish. In *Inequalities in Statistics and Probability*, volume 5, pages 84–103. Institute of Mathematical Statistics, 1984.

[34] Marc Lavielle. Using penalized contrasts for the change-point problem. *Signal Processing*, 85:1501–1510, August 2005.

[35] Emilie Lebarbier. Detecting multiple change-points in the mean of gaussian process by model selection. *Signal Processing*, 85:717–736, April 2005.

[36] Erez Lieberman-Aiden, Nynke L. van Berkum, Louise Williams, Maxim Imakaev, Tobias Ragoczy, Agnes Telling, Ido Amit, Bryan R. Lajoie, Peter J. Sabo, Michael O. Dorschner, Richard Sandstrom, Bradley Bernstein, M. A. Bender, Mark Groudine, Andreas Gnirke, John Stamatoyannopoulos, Leonid A. Mirny, Eric S. Lander, and Job Dekker. Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science*, 326(5950), October 2009.

[37] YH et. al Loh. The Oct4 and Nanog transcription network regulates pluripotency in mouse embryonic stem cells. *Nature Genetics*, 38:431–440, March 2006.

[38] Jun Ma, Craig Dobry, Damian Krysan, and Anuj Kumar. Unconventional genomic architecture in the budding yeast *saccharomyces cerevisiae* masks the nested antisense gene nag1. *Eurkaryotic Cell*, 7(8):1289–1298, August 2008.

[39] John Marioni, Christopher Mason, Shrikant Mane, Matthew Stephens, and Yoav Gilad. RNA-Seq: An assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, 18:1509–1517, June 2008.

[40] Joseph A. Martens, Lisa Laprade, and Fred Winston. Intergenic transcription is required to repress the *S*accharomyces cerevisiae SER3 gene. *Nature*, 429:571–574, May 2004.

[41] Igor Martianov, Aroul Ramadass, Ana Serra Barros, Natalie Chow, and Alexandre Akoulitchev. Repression of the human dihydrofolate reductase gene by a non-coding interfering transcript. *Nature*, 445:666–670, February 2007.

[42] Fumihito Miura, Noriko Kawaguchi, Jun Sese, Atsushi Toyoda, Masahira Hattori, Shinichi Morishita, and Takashi Ito. A large-scale full-length cDNA analysis to explore the budding yeast transcriptome. *PNAS*, 103(47):17486–17851, 21 November 2006.

[43] Vamsi K Mootha, Cecilia M Lindgren, Karl-Fredrik Eriksson, Aravind Subramanian, Smita Sihag, Joseph Lehar, Pere Puigserver, Emma Carlsson, Martin Ridderstr—[aring]—le, Esa Laurila, Nicholas Houstis, Mark J Daly, Nick Patterson, Jill P Mesirov, Todd R Golub, Pablo Tamayo, Bruce Spiegelman, Eric S Lander, Joel N Hirschhorn, David Altshuler, and Leif C Groop. Pgc-1—[alpha]—-responsive genes in-

volved in oxidative phosphorylation are coordinately downregulated in human diabetes. *Nature Genetics*, 34(3), June 2003.

[44] Ugrappa Nagalakshmi, Zhong Wang, Karl Waern, Chong Shou, Debasish Raha, Mark Gerstein, and Michael Snyder. The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, 320:1158441–1349, May 2008. 10.1126/science.1158441.

[45] Dmitri Parkhomchuk, Tatiana Borodina, Vyacheslav Amstislavskiy, Maria Banaru, Linda Hallen, Sylvia Krobitsch, Hans Lehrach, and Alexey Soldatov. Transcriptome analysis by strand-specific sequencing of complementary dna. *Nucleic Acids Research*, July 2009.

[46] A. C. Pease, D. Solas, E. J. Sullivan, M. T. Cronin, C. P. Holmes, and S. A. Fodor. Light-generated oligonucleotide arrays for rapid dna sequence analysis. *Proceedings of the National Academy of Sciences*, 91:5022–5026, 1994.

[47] Fabiana Perocchi, Zhenyu Xu, Sandra Clauder-Münster, and Lars M. Steinmetz. Antisense artifacts in transcriptome microarray experiments are resolved by actinomycin D. *Nucleic Acids Research*, 35(19), 26 September 2007.

[48] Franck Picard, Stephane Robin, Marc Lavielle, Christian Vaisse, and Jean-Jacques Daudin. A Statistical Approach for Array CGH Data Analysis. *BMC Bioinformatics*, 6(27), February 2005.

[49] Franck Picard, Stephane Robin, Emilie Lebarbier, and Jean-Jacques Daudin. A segmentation/clustering model for the analysis of array CGH data. *Biometrics*, 63:758–766, 2007.

[50] Yuan Qi, Alex Rolfe, Kenzie MacIsaac, Georg Gerber, Dmitry Pokholok, Julia Zeitlinger, Timothy Danford, Robin Dowell, Ernest Fraenkel, Tommi Jaakkola, Richard Young, and David Gifford. High-resolution computational models of genome binding events. *Nature Biotechnology*, 24(8):963–970, August 2006.

[51] Annie Rochette, Frederic Raymond, Jean-Michel Ubeda, Martin Smith, Nadine Messier, Sebastien Boisvert, Philippe Rigault, Jacques Corbeil, Marc Ouellette, and Barbara Papadopoulou. Genome-wide gene expression profiling analysis of *leishmania*

major and *leishmania infantum* developmental stages reveals substantial differences between the two species. *BMC Genomics*, 9(255), May 2008.

[52] Thomas Royce, Joel Rozowsky, Paul Bertone, Manoj Samanta, Viktor Stolc, Sherman Weissman, Michael Snyder, and Mark Gerstein. Issues in the analysis of oligonucleotide tiling microarrays for transcript mapping. *Trends in Genetics*, 21(8):466–475, August 2005.

[53] E. Segal, R. Yelensky, A. Kaushal, T. Pham, A. Regev, D. Koller, and N. Friedman. Genexpress: A visualization and statistical analysis tool for gene expression and sequence data. In *Proceedings of the 11th International Conference on Intelligent Systems for Molecular Biology (ISMB)*, 2004.

[54] Eran Segal, Yvonne Fondufe-Mittendorf, Lingyi Chen, AnnChristine Tharingstroum, Yair Field, Irene K. Moore, Ji-Ping Wang, and Jonathan Widom. A genomic code for nucleosome positioning. *Nature*, 442(7104), July 2006.

[55] Edward A. Sekinger, Zarmik Moqtaderi, and Kevin Struhl. Intrinsic histone-dna interactions and low nucleosome density are important for preferential accessibility of promoter regions in yeast. *Molecular Cell*, 18(6):735–748, June 2005.

[56] Susann Stjernqvist, Tobias Ryden, Martin Skold, and Johan Staaf. Continuous-index hidden markov modelling of array cgh copy number data. *Bioinformatics*, 23(8):1006–1014, February 2007.

[57] Aravind Subramanian, Pablo Tamayo, Vamsi K. Mootha, Sayan Mukherjee, Benjamin L. Ebert, Michael A. Gillette, Amanda Paulovich, Scott L. Pomeroy, Todd R. Golub, Eric S. Lander, and Jill P. Mesirov. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences of the United States of America*, 102(43), October 2005.

[58] Jay P. Uhler, Christina Hertel, and Jesper Q. Svejstrup. A role for noncoding transcription in activation of the yeast pho5 gene. *Proceedings of the National Academy of Sciences*, 104(19), 2007.

[59] Alexander E. Urban, Jan O. Korbel, Rebecca Selzer, Todd Richmond, April Hacker, George Popescu, Joseph F. Cubells, Roland Green, Beverly S. Emanuel, Mark B. Ger-

stein, Sherman M. Weissman, and Michael Snyder. High-resolution mapping of DNA copy alterations in human chromosome 22 using high-density tiling oligonucleotide arrays. *PNAS*, 103(12):4534–4539, March 2006.

[60] F.C. Wardle and D.T. et al Odom. Zebrafish promoter microarrays identify actively transcribed embryonic genes. *Genome Biology*, 7(R71), August 2006.

[61] L Weng, H Dai, Y Zhan, Y He, S Stepaniants, and D Bassett. Rosetta error model for gene expression analysis. *Bioinformatics*, 22(9):1111–1121, 2006.

[62] Andreas Werner, Gabriele Schmutzler, Mark Carlile, Colin Miles, and Heiko Peters. Expression profiling of antisense transcripts on dna arrays. *Physiological Genomics*, 28:294–300, November 2007.

[63] Brian Wilhelm and Josette-Renee Landry. Rna-seq: quantitative measurement of expression through massively parallel rna-sequencing. *Methods*, 48(3):249–257, July 2009.

[64] Brian Wilhelm, Samuel Marguerat, Stephen Watt, Falk Schubert, Valerie Wood, Ian Goodhead, Christopher Penkett, Jane Rogers, and Jurg Bahler. Dynamic repertoire of a eukaryotic transcriptome surveyed at a single-nucleotide resolution. *Nature*, 453:1239–1243, June 2008.

[65] Yee Hwa Yang, Sandrine Dudoit, Percy Luu, David Lin, Vivian Peng, John Ngai, and Terence Speed. Normalization for cdna microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research*, 30(4), 2002.

[66] Ming Zheng, Leah O. Barrera, Bing Ren, and Ying Nian Wu. Chip-chip: Data, model, and analysis. *Biometrics*, 63(3):787–796, March 2007.

[67] Sika Zheng and Liang Chen. A hierarchical Bayesian model for comparing transcriptomes at the individual transcript isoform level. *Nucleic Acids Research*, pages 1–16, May 2009.