DISSERTATION

FROM RNA-SEQ TO GENE ANNOTATION USING THE SPLICEGRAPHER METHOD

Submitted by

Mark F. Rogers

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2013

Doctoral Committee:

Advisor: Asa Ben-Hur

Christina Boucher
Charles Anderson
Anireddy S.N. Reddy

ABSTRACT

FROM RNA-SEQ TO GENE ANNOTATION USING THE SPLICEGRAPHER METHOD

Messenger RNA (mRNA) plays a central role in carrying out the instructions encoded in a gene. A gene's components may be combined in various ways to generate a diverse range of mRNA molecules, or transcripts, through a process called *alternative splicing* (AS). This allows each gene to produce different products under different conditions, such as different stages of development or in different tissues. Researchers can study the diverse set of transcripts a gene produces by sequencing its mRNA. The latest sequencing technology produces millions of short sequence *reads* (RNA-Seq) from mRNA transcripts, providing researchers with unprecedented opportunities to assess how genetic instructions change under different conditions. It is relatively inexpensive and easy to obtain these reads, but one limitation has been the lack of versatile methods to analyze the data.

Most methods attempt to predict complete mRNA transcripts from patterns of RNA-Seq reads ascribed to a particular gene, but the short length of these reads makes transcript prediction problematic. We present a method, called SpliceGrapherXT, that takes a different approach by predicting *splice graphs* that capture in a single structure all the ways in which a gene's components may be assembled. Whereas other methods make predictions primarily from RNA-Seq evidence, SpliceGrapherXT uses gene annotations describing known transcripts to guide its predictions. We show that this approach allows SpliceGrapherXT to make predictions that encapsulate gene architectures more accurately than other state-of-the-art methods. This accuracy is crucial not only for updating gene annotations, but our splice graph predictions can contribute to more accurate transcript predictions as well. Finally we demonstrate that by using SpliceGrapherXT to assess AS on a genome-wide scale, we can gain new insights into the ways that specific genes and environmental conditions may impact an organism's transcriptome. SpliceGrapherXT is available for download at http://splicegrapher.sourceforge.net.

TABLE OF CONTENTS

1 Introduction **1**

2 Background **6**

2.1 RNA-Seq Data 7

2.2 Transcriptome Analysis Using RNA-Seq 9

2.2.1 Cufflinks 11

2.2.2 SLIDE 12

2.2.3 IsoLasso and IsoLasso/CEM 13

2.2.4 iReckon 14

2.2.5 RNA-Seq Biases 15

2.3 Predicting Transcripts vs. Splice Graphs 16

3 Accurate Splice Graph Prediction with SpliceGrapherXT **19**

3.1 The SpliceGrapherXT Pipeline 19

3.1.1 Splice Graph Initialization 21

3.1.2 Making Predictions from RNA-Seq Data 23

3.1.3 Overview of the Prediction Algorithm 25

3.2 Performance Evaluation 26

3.2.1 Methods for Comparison 27

3.2.2 Simulation experiments 28

3.2.3 Real data 29

3.3 Results 30

3.3.1 Simulation experiments 30

3.3.2 Real data 32

3.4 Conclusion 33

4 Filtering Spliced Alignments **35**

4.1 Related Work 35

4.2 RNA-Seq Alignment 38

4.3 Splice site classifiers . . . 38
4.3.1 SpliceGrapher methods . . . 39
4.4 Performance evaluation . . . 41
4.4.1 Filtering alignments with simulated RNA-Seq data . . . 41
4.4.2 Filtering alignments with real RNA-Seq data . . . 43
4.5 Conclusion . . . 45

5 Resolving Ambiguities Using Realignment . . . **46**
5.1 Realignment Pipeline . . . 46
5.1.1 Creating putative transcripts . . . 46
5.1.2 Realigning reads . . . 48
5.1.3 Resolving exons . . . 49
5.2 Related Work . . . 49
5.3 Evaluating the pipeline . . . 50
5.3.1 Paired-end vs. single-end reads . . . 54
5.4 Discussion . . . 54

6 Using Splice Graphs to Predict Transcripts . . . **55**
6.1 PSGInfer . . . 56
6.2 IsoLasso . . . 57
6.3 Performance Evaluation . . . 57
6.3.1 Simulated data . . . 59
6.3.2 Real data . . . 60
6.3.2.1 Resolving exons . . . 62
6.4 Conclusion . . . 63

7 From Splice Graphs to Biological Insights . . . **64**
7.1 Assessing AS Across Embryonic Developmental Stages . . . 64
7.1.1 AS Activity Peaks Near Phylotypic Stage . . . 65
7.1.2 Old Genes Exhibit More AS Than Young Genes . . . 67
7.2 Assessing AS Between Variants . . . 68

7.2.1 AS Within Novel Genes . . . 71
7.3 Visualizing Chimeras . . . 73
7.3.1 Future Work . . . 75
7.4 Conclusion . . . 76

8 Conclusion . . . **77**
8.1 Future Work . . . 78
8.1.1 Gene Prediction . . . 78
8.1.2 Improving IR Predictions . . . 79
8.1.3 Additional Comparisons . . . 80

A Original SpliceGrapher Inference Rules . . . **98**
A.1 Alternative Splicing Inference from RNA-Seq . . . 98
A.1.1 Intron Retention . . . 98
A.1.2 Alternative 3' and 5' Events . . . 102
A.1.3 Exon Skipping . . . 102

B Counting AS Events . . . **104**
B.1 Intron retention . . . 104
B.2 Alternative 3' and 5' events . . . 105
B.3 Exon skipping . . . 105

C File formats . . . **106**
C.1 Gene Models . . . 106
C.1.1 GFF3 Format . . . 106
C.1.2 GTF Format . . . 107
C.2 EST Alignments . . . 108
C.3 RNA-Seq Alignments . . . 108
C.3.1 SAM Format . . . 110
C.3.2 CIGAR Strings . . . 110
C.4 Splice Graphs . . . 111

D Splice Site Classifier Performance
for 107 Species **112**

E The SpliceGrapherXT Software **114**
E.1 Overview 114
E.1.1 Downloading and Installation 114
E.1.2 Brief Tutorial 115
E.1.2.1 Example 1: Create Classifiers 115
E.1.2.2 Example 2: Filter Alignments 116
E.1.2.3 Example 3: Predict a Splice Graph 116
E.2 The *SpliceGrapher* Environment 117
E.2.1 Environment Variables 117
E.2.2 SpliceGrapher Configuration File 117
E.3 Splice Site Classifiers 118
E.3.1 Building Classifiers 118
E.3.1.1 Identifying Splice-Site Dimers 119
E.3.1.2 Generating Training Data 120
E.3.1.3 Optimizing Classifiers 121
E.3.1.4 Intron and Exon Sequence Lengths 123
E.3.2 Classifier Data 123
E.3.3 Generating ROC Curves 124
E.4 Predicting Splice Graphs 125
E.4.1 Filtering Spliced Alignments 125
E.4.1.1 Collating SAM Files 125
E.4.2 Creating Splice Graphs from Gene Models 126
E.4.2.1 Converting GTF Annotations 127
E.4.3 Creating Splice Graphs from ESTs 127
E.4.4 `predict_graphs.py` 128
E.4.5 `predict_splicegraph.py` 128
E.4.5.1 Finding Splice Forms 129

E.5 Viewing Splice Graphs . . . 131
E.5.1 Simple Viewing Tool . . . 131
E.5.1.1 Shrinking Introns . . . 132
E.5.1.2 Plot Types . . . 132
E.5.1.3 Splice Graphs . . . 133
E.5.1.4 Isoforms . . . 133
E.5.1.5 Read Coverage . . . 133
E.5.1.6 Splice Junction Coverage . . . 134
E.5.1.7 X-Y Plots . . . 134
E.5.2 Multiple Plots . . . 135
E.5.2.1 File Options . . . 135
E.5.2.2 Display Options . . . 135
E.5.3 High-Quality Plotting . . . 136
E.5.3.1 Main Section . . . 138
E.5.3.2 Plot Sections . . . 138
E.5.3.3 Source Files for Plots . . . 139
E.5.3.4 Output Options . . . 139
E.6 File Formats . . . 140
E.6.1 Gene Model Files . . . 140
E.6.1.1 GFF3 Files . . . 140
E.6.1.2 GTF Files . . . 140
E.6.2 Splice Graph Files . . . 141
E.6.3 SAM Files . . . 141
E.6.4 BED and WIG Files . . . 141
E.6.5 PSL Files . . . 141
E.7 Pre-Built Classifiers . . . 142

Chapter 1

Introduction

The central dogma of molecular biology states that DNA is transcribed into messenger RNA (mRNA), which is then translated into protein [36]. However, this model does not tell the whole story. Transcription produces a strand of precursor mRNA (pre-mRNA) that is then spliced to create the mature messenger RNA [20]. The pre-mRNA transcript consists of two components: *exons* that contain the protein-encoding message, and *introns* that occupy the regions between every pair of exons (Figure 1.1). The splicing process excises the introns from the pre-mRNA and concatenates the exons together to form the final mRNA transcript. Splicing does not occur in the same way under all conditions; for example, exons may be omitted, or introns retained in the final transcript, a phenomenon called *alternative splicing* (AS).

AS plays a critical role in cell activities that influence protein production, allowing an organism to produce a more diverse set of proteins than the number of genes might suggest; for example, the most recent ENSEMBL human genome database reports 20,442 protein-coding genes [56], yet human cells are estimated to produce over 100,000 different proteins [5]. Similarly, AS can help to explain vast phenotypic differences between organisms that have similar numbers of genes. For example, the nematode *C. elegans* has 19,735 protein-coding genes—nearly as many as humans—yet is simple enough that the developmental path of every somatic cell has been mapped out [203]. In addition, AS provides a means for regulating gene expression, allowing cells to respond rapidly to changing conditions. For example, a gene's expression may be suppressed when AS yields transcripts that are degraded rather than translated into protein [104, 166, 28]. AS may shorten dramatically the protein-encoding sequence in a strand of mRNA, initiating a process known as *nonsense-mediated decay* (NMD). In this process, protein complexes rapidly degrade these shortened mRNA transcripts and prevent them from being translated into proteins. This pathway is important because if these shortened transcripts were translated they would produce truncated proteins that could have deleterious effects on a cell [28].

AS has also been implicated in a number of diseases such as various forms of cancer [226], spinal muscular atrophy [197], heart disease [252], dementia [7] and other diseases [212]. For example, *Syk* is a cell-signaling protein that suppresses the spread (metastasis) of cancer cells in humans. Researchers have

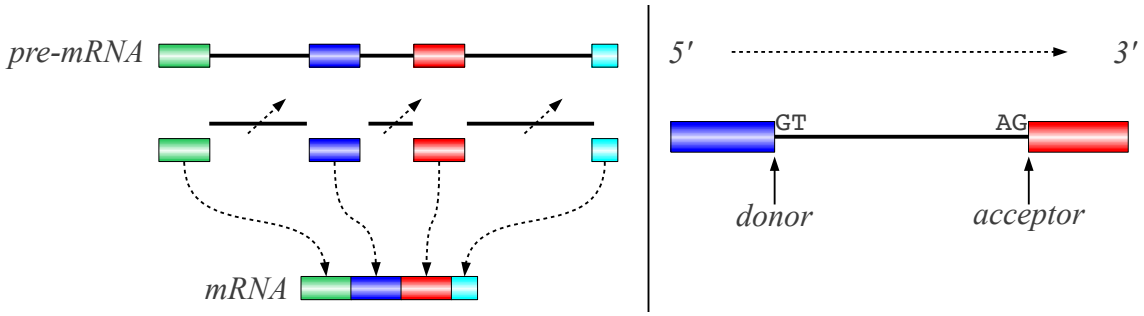

Figure 1.1: Diagram of the splicing process. On the left, introns (solid lines) are excised from pre-mRNA while exons (colored bars) are merged to form mature mRNA. By convention, mRNA sequences are oriented left-to-right from their 5' end to their 3' end (right). An intron has a donor splice site at its 5' end (usually associated with a GT dimer) and an acceptor splice site at its 3' end (typically an AG dimer).

shown that an alternatively-spliced form of this protein found predominantly in breast tumors lacks an exon required for correct localization and hence fails to prevent metastasis [231]. As another example, the gene BRCA1 has a well-documented association with breast cancer in humans (see, e.g., [118, 159, 229]). A common mutation in BRCA1 creates a novel 3' splice site that truncates the resulting protein, inhibiting its ability to suppress tumor growth [77]. Researchers have also shown that AS in the human gene *APOE* may be associated with neurodegeneration in Alzheimer's disease [219]. Accurate gene models and good methods for assessing AS are therefore critical to understanding the underlying causes and progression of these and other diseases.

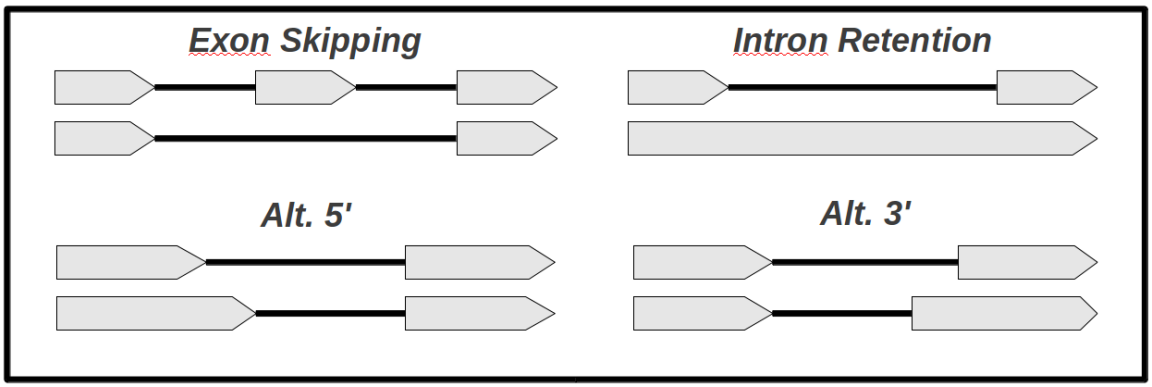

Figure 1.2: Schematic description of the four main forms of AS discussed in this thesis.

Several forms of AS are described in the literature (for example, see [93]), but these may be distilled into four canonical types: exon skipping, intron retention, alternative 5' sites and alternative 3' sites [151] (see Figure 1.2). By convention we depict transcripts, introns and exons from left to right: the left end is the *5'* or *upstream* end, while the right end the *3'* end or *downstream* end (Figure 1.1). Exon skipping, the most common form of AS in mammals, occurs when an exon appears in some transcripts but is removed

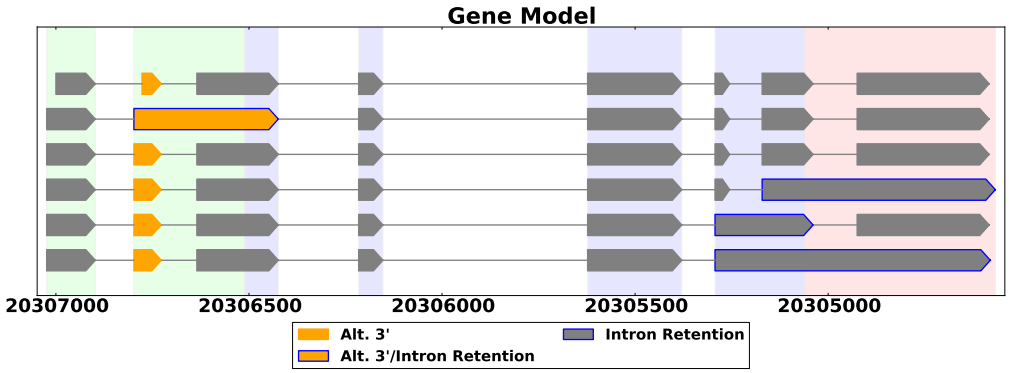

Figure 1.3: Example of the information depicted in a gene model for a gene from *A. thaliana*. This gene has six known transcripts, each of which represents a different combination of possible exons. The blue shaded regions represent regions that encode proteins, while the green and pink shaded regions represent non-coding regions.

from others. Intron retention is the most common form of AS in plants [166], and occurs when an intron is excised from some transcripts but retained in others. Alternative 5' and alternative 3' events occur when an intron is excised at two or more locations on its 5' end or its 3' end. Note that the terms "alternative 5' " and "alternative 3' " are applied relative to the intron involved in an event. When we refer to the corresponding exons, we use the terms *alternative donor* or *alternative acceptor* sites, where a *donor* site is the splice site at an exon's 3' end, and an *acceptor* site is the splice site at its 5' end.

The most accurate information on the transcripts that may arise from a gene are stored in databases of gene annotations known as *gene models* [89] (Figure 1.3). These databases include information about the locations of genes within a chromosome, the locations of each gene's exons, the transcripts in which they participate, and the coding regions associated with different transcripts [89, 47, 206]. Gene models are validated by curators who monitor the literature for annotations that capture a gene's important features [206]. The best evidence for updating gene models comes from biological "wet lab" assays that can determine the transcripts present in a sample. However, these methods are relatively slow and expensive, so automated annotation methods are used to predict the genes and transcripts to target in these assays [55, 218, 99]. For example, when the *A. thaliana* annotations were upgraded from version 9 to version 10 (TAIR9 to TAIR10), they included predictions from [55] that were confirmed using wet-lab assays.

Most research into AS relies on transcriptome sequencing as well as accurate gene annotations to capture the distinct exons and transcripts that are expressed. Sequencing technology has evolved rapidly in recent years, with each generation of sequencing tools producing larger volumes of data at ever-increasing speeds and at lower cost. The growing volume of new data, combined with decades' worth of legacy data, provide

an unprecedented opportunity to probe the transcriptome (the set of all possible transcripts) of many species. Thus there is an increasing demand for tools that can process large data sets to predict the possible exons and transcripts in a genome.

A key goal in transcriptome analysis is to predict exons or transcripts on a genome-wide basis. Genome-wide transcriptome studies make it possible to assess, for example, changes in transcription that may occur between different tissue types [144, 134, 204, 237, 157], across different stages of development [161, 64, 218, 248, 155], or between different organisms [242, 184, 31, 172]. The latest "next-generation" sequencing data known as *RNA-Seq* generates millions of sequences, or *reads*, that can provide evidence even for weakly-expressed transcripts, making it possible to discover prodigious numbers of novel transcripts and AS events. However, the relatively short length of RNA-Seq reads introduces challenges both for sequence alignment and for interpreting evidence for novel events. For example, RNA-Seq evidence for AS in one part of a gene may be impossible to reconcile with AS in another location. This can make it infeasible to reconstruct complete transcripts, but may allow us to predict the exons involved.

Transcriptome prediction problems tend to fall into one of three categories based on the amount of evidence available for predictions: *de novo* transcriptome assembly, reference-guided gene annotation, and gene annotation updating. In *de novo* transcriptome assembly, the goal is to assemble transcriptomes using only RNA-Seq data. Solutions to this problem are best suited to organisms without genome sequences or whose genomic sequences are in early development. Example approaches include the Trinity suite [63] and trans-ABySS [174]. In reference-guided annotation the goal is to predict transcripts using RNA-Seq data aligned to a reference genome. This problem arises when researchers wish to conduct transcriptome analysis in the absence of gene models. A typical approach is first to align RNA-Seq reads to the reference genome and then predict the transcripts present in the data; Scripture [68] and Cufflinks [218] are two methods that can make predictions based solely on these two kinds of evidence. Finally there is the challenge of updating gene annotations using RNA-Seq, a reference genome and curated gene models. Tools such as Cufflinks, IsoLasso [113] and IsoLasso/CEM [114] can predict novel transcripts based on these forms of evidence. Because they can use more evidence than the other approaches, these solutions have greater potential for making accurate predictions; however, the current state of the art falls well short of its potential. It is this last problem we address in our work.

In this thesis we describe three related approaches designed to update gene annotations by predicting the exons recapitulated in sequencing data. The first approach, called *SpliceGrapherXT*, uses evidence from the

latest "next-generation" sequencing data known as *RNA-Seq* to predict *splice graphs* that capture in a single structure all the ways in which exons in a given gene may be assembled. In these graphs, exons are depicted as nodes and introns are the edges that connect them, thus facilitating analysis and allowing researchers to visualize a gene's architecture easily. SpliceGrapherXT achieves high precision by predicting novel exons only when the evidence for them is unambiguous, leaving exons with ambiguous boundaries unresolved. Our second approach uses a realignment procedure to identify paths through splice graphs that are supported by RNA-Seq data, allowing us to resolve up to 50% of the exons. Our third approach demonstrates how we can use SpliceGrapherXT's splice graph predictions in conjunction with other methods to generate transcript predictions that are more accurate than those of competing methods.

Throughout this thesis we show that our approaches yield predictions for splice graphs and transcripts that are substantially more accurate than other state-of-the-art methods. Ultimately, however, we are interested to know how our methods may contribute to biological research. Thus we finish with overviews of three projects where SpliceGrapherXT has been used to gain insights into biological processes: changes in AS that occur during plant embryonic development; changes in AS caused by "knocking out" an important splicing regulatory gene, and early insights into the unusual behavior of mRNA *chimeras*. This work has resulted in a number of publications that describe the methods involved [179, 169, 177], along with the biological insights generated using these tools [99, 172, 58, 4].

Chapter 2

Background

We described briefly the concepts of splicing and alternative splicing in Chapter 1, but to understand the processes that SpliceGrapherXT and other tools try to predict, it is helpful to know more about the molecular interactions involved. Splicing is a complex process that excises an intron and splices together its flanking exons (Figure 1.1). Each time splicing occurs, protein complexes that form the *spliceosome* bind to locations close to the 5' and 3' sites at either end of an intron. The spliceosome consists of five small nuclear ribonucleoprotein molecules (snRNPs), identified as U1, U2, U4, U5 and U6, that are assembled in several stages [235]: First, the U1 snRNP binds near the 5' site (Figure 2.1, panel I). Next, the U2 snRNP binds near the 5' site and folds the intron onto itself, binding at a second intronic location called the *branch point*. The remaining snRNPs, U4, U5 and U6, join the complex and trigger the splicing process (Figure 2.1, panel II). First the intron is cleaved at its 5' site and the 5' site is joined to the branch site to form a structure called a *lariat* (Figure 2.1, panel III). Next, the intron is cleaved at its 3' site (usually characterized by an AG dimer), the two flanking exons are ligated (spliced) together, and the spliceosome dissociates from the merged exons (Figure 2.1, panel IV).

In addition to the five principal snRNPs, at least 300 additional proteins participate in the process, though their exact functions and number remain to be determined (see, e.g., [150]). Many of these proteins, notably the serine/arginine (SR) family and heterogeneous nuclear ribonucleoproteins (hnRNPs), regulate alternative splicing by enhancing or suppressing splicing (for reviews, see [165, 166, 30, 124, 93, 168, 172, 176]). These proteins may operate, for example, by assisting or preventing spliceosomal snRNPs from binding to pre-mRNA [145, 185, 223] or by competing with other splicing factors [251, 26]. Predicting the transcripts that may occur for a given gene is thus a challenging problem with enormous potential for generating new insights.

Most approaches attempt to predict the complete mRNA transcripts that may arise from a gene, while SpliceGrapherXT instead predicts splice graphs (see Figure 2.6). Both of these approaches depend on mapping query sequences to a reference genome, in effect, reverse-engineering the splicing process. Query sequences may be produced directly, by sequencing mRNA samples, or they may be inferred by recording

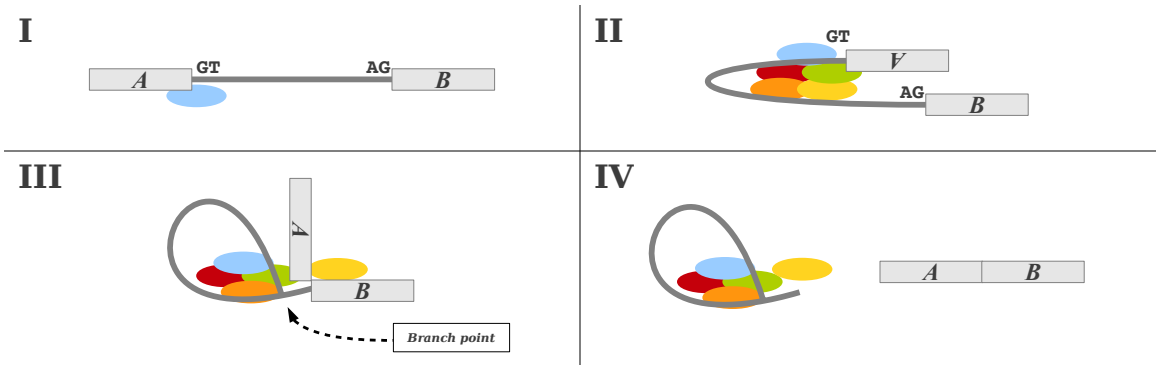

Figure 2.1: A simplified view of the splicing process. (I) This pre-mRNA transcript consists of two exons, A and B, and an intron depicted as a dark line between them. Spliceosome assembly begins when the U1 snRNP (blue) binds near the GT 5' site. (II) snRNP U2 is added to the complex, folding the intron onto itself. The remaining U4, U5, and U6 snRNPs then join the complex. (III) The intron is cleaved at its 5' site which forms a lariat structure by binding at the *branch site*. (IV) Finally the intron is cleaved at its 3' site, the exons are spliced together, and the snRNP complex dissociates.

where mRNA fragments bind to probes on microarray slides. The data provided by different technologies fall roughly into three categories: conventional sequences, such as expressed sequence tags (ESTs) or full-length cDNA (fl-cDNA); "next-generation" sequencing (NGS) *reads*, and microarray probe intensities. In the interest of completeness this list includes microarrays, but they have limitations that make the data unreliable for transcriptome prediction (for details, see [143, 71, 169]), thus microarray data are not included in this thesis.

ESTs and fl-cDNA are produced using conventional sequencing methods that are based on the Sanger chain-termination method originally developed in 1977 [188]. fl-cDNA represent the full length of an mRNA transcript, while ESTs can be smaller than a single exon. By convention we refer to these sequences collectively as ESTs. EST sequences for mRNA are mapped to a genome using sequence alignment algorithms that are designed for these conventional sequences. NGS reads tend to be shorter than ESTs, but are far more plentiful: in January 2011, GenBank reported a total of nearly 67 million EST records for all species [189], while a single NGS run can sequence up to 600 billion bases, generating up to 4 billion 150-nt reads [133]. As with ESTs, NGS sequences for mRNA (RNA-Seq) are mapped to a genome using sequence alignment, though the algorithms are designed specifically to handle the shorter RNA-Seq reads.

2.1 RNA-Seq Data

The sequences produced by NGS methods have characteristics that complicate the task of identifying the mRNA transcripts represented in a sample. An RNA-Seq read may shorter than a typical exon, making

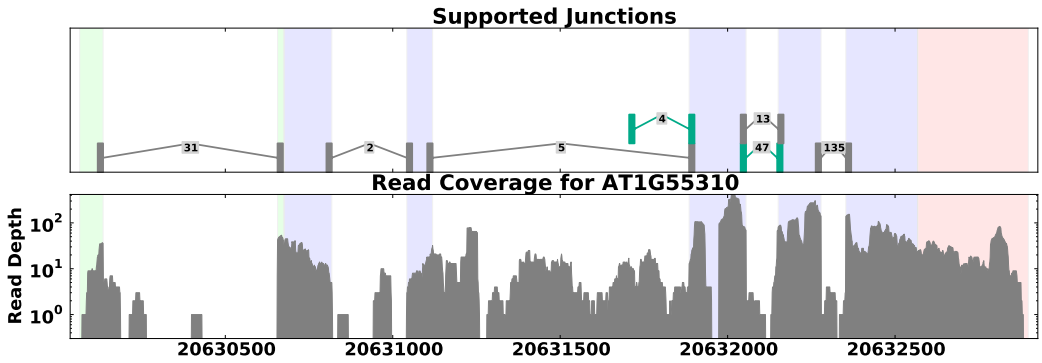

Figure 2.2: This figure generated by *SpliceGrapherXT* depicts read coverage for an *A. thaliana* gene in two ways. The top panel shows splice junctions that were recapitulated in the RNA-Seq data with labels showing the number of reads that aligned across each junction and novel splice junctions highlighted in green. The bottom panel shows the distribution of reads across the gene. Here the read depths range from 1 to over 300, demonstrating that read coverage can be highly variable even within a single exon region (vertical shaded bars). This variability can make it difficult to distinguish between weakly-expressed splice forms and background noise.

it difficult to identify a unique origin within a reference sequence. In addition, NGS base-call error rates tend to increase with read length, raising the chance of a mismatch when aligning a read to a reference sequence [195]. These ambiguities are exacerbated by the presence of paralogous genes that can give rise to reads that align well in multiple locations (called *multireads*). Much of the work on analyzing NGS reads has focused on aligning reads within exonic regions, and many methods exist for the problem of aligning reads without gaps, such as MAQ [109], BWA [107] and BowTie [102].

Accurate splice junction identification is crucial for making accurate transcript predictions, but the short length of RNA-Seq reads makes spliced alignment especially challenging. A splice junction may occur anywhere within a read, so the read may have just a few bases on one side of a junction. A common strategy is to split a read into smaller parts, implicitly confining the splice junction to just one piece (Figure 2.3). If one of the parts aligns to a unique location in the genome, these algorithms then try to align the remaining parts in the same vicinity, applying heuristics that dictate acceptable intron lengths or splice-site dimers [216, 10, 127]. A common heuristic, shown in Figure 2.3, is to seek locations that infer an intron bounded by GT and AG dimers. However, we demonstrated recently that algorithms that rely on these heuristics may predict spurious junctions [179]. Instead of heuristics, MapSplice [230] predicts junctions by using statistical measures to assess the quality of read alignments across each junction. SpliceGrapherXT [179, 177] and PALMapper [83] predict splice sites using sophisticated models that can recognize complex signals in the mRNA sequence surrounding splice sites. The splice junctions these tools predict can be used to

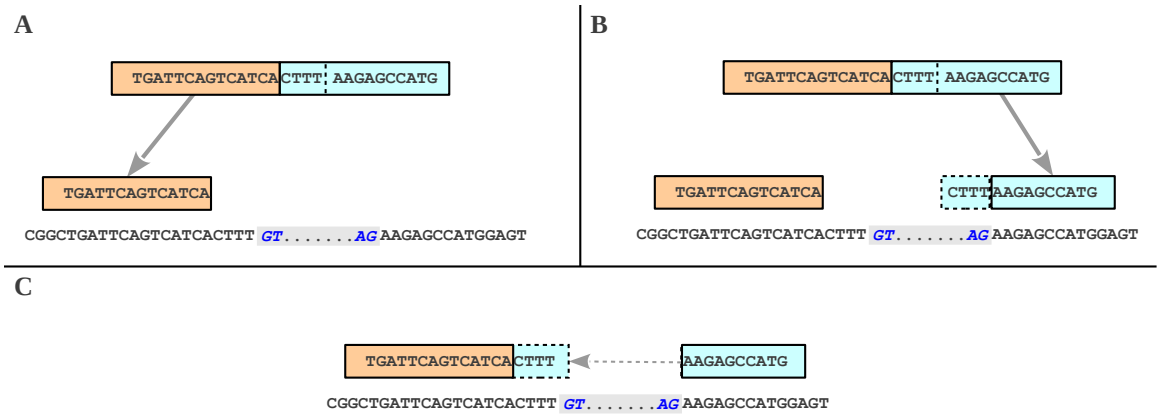

Figure 2.3: Depiction of a common approach to spliced alignments. Here an RNA-Seq read is split into two parts. (A) First, one part aligns completely to a location in the reference genome. (B) Next a "best" location is found for the second part. Normally this close to the location where the first part aligned, as determined by a predetermined range of acceptable intron lengths. (C) Finally, the sequences in the two parts are shifted until the two parts abut legitimate splice sites. In this case, GT and AG dimers are used to identify legitimate splice sites.

reconstruct the transcripts represented in the original sample.

2.2 Transcriptome Analysis Using RNA-Seq

Transcriptome analysis allows us to gain insights into a variety of genomic processes such as the discovery of novel AS events [144, 153, 55, 179] or the expression levels of transcripts for a particular RNA-Seq data set [113, 68, 148, 54, 111, 140]. Given this ability to make predictions on a genome-wide scale, we can also compare AS or transcript expression between different species [242, 184, 31]; different variants of a species [62, 182, 84, 149]; different cell types [144, 134, 204, 237, 157], or different stages of development [64, 218, 248, 155] (see also Chapter 7). NGS has radically changed the way researchers study the transcriptomes of model organisms. NGS tends to be cheap, fast and more sensitive to rare splice forms than microarrays or ESTs [232]. To investigate a transcriptome, researchers apply NGS to mRNA samples, generating datasets that consist of millions of RNA-Seq reads. In a typical study, researchers align RNA-Seq reads to a reference genome to determine the genes and splice forms represented in a sample (see, e.g., [144, 153, 55]). Read coverage (the distribution of reads that align within a region of interest) then provides evidence for exons and splice junctions recapitulated in the RNA-Seq data (see Figure 2.2).

The first studies that predicted transcripts from RNA-Seq data were performed mainly in mammals and focused on detection of exon skipping, the most common form of AS in mammals [144, 153, 204, 228, 209].

Reads that align across splice junctions provide evidence for junctions in a data set, so these methods focused on finding evidence for novel combinations of known exons. This approach has been effective for detecting novel exon-skipping events [204, 228], but it is inadequate for detecting other forms of AS. Comprehensive transcript or exon prediction tools should be able to infer exons involved in any kind of AS and should predict novel splice sites accurately.

Programs that predict mRNA transcripts from NGS data usually are designed to predict complete mRNA transcripts using RNA-Seq data and a reference genome as evidence. These methods first align reads to a genome and then assemble transcripts using the alignments as evidence. Cufflinks [216], Scripture [68], SLIDE [111], NSMAP [239] and TAU [160] were designed to predict transcript expression levels by first predicting mRNA transcripts and then using RNA-Seq coverage to estimate their expression. IsoLasso [113] and IsoLasso/CEM [114] use an alternative approach that attempts to predict transcripts and expression levels simultaneously. Cufflinks, the most popular of these tools, has been used to predict novel transcripts in studies of various species including mammals [216, 215, 122], plants [142, 117] and insects [64, 27, 8]. TAU [160] was designed and validated using the model plant *A. thaliana*, and it predicted an assortment of non-canonical splice junctions associated with splice forms that were later validated via RT-PCR [55]. However, experiments have shown that Cufflinks and TAU predictions suffer poor precision because they use evidence from false-positive splice junctions [179, 177]. Spurious exon predictions can also hurt transcript prediction accuracy as we discuss in Chapter 6.

In the absence of a reference genome, *de novo* transcriptome assembly packages such as ABySS [196] and Trinity [63] can use NGS reads to construct a coherent model for each gene. These methods construct de Bruijn graphs from overlapping k-mers and then trace paths through the graphs to predict genomic or transcript sequences. ABySS [196] has been used to predict alternative splicing patterns such as exon skipping, intron retention and alternative 5' splice sites [19]. The Trinity suite [63, 1] assembles reads into unique contigs that it then uses to predict alternatively spliced transcripts. These are merged into splice graphs that encompass all splice forms for a gene. By tracing all paths through a graph it is able to report the splice forms detected in the RNA-Seq data.

For organisms that are sparsely annotated, it may be appropriate to use a reference-guided method that requires only RNA-Seq alignments to a reference genome to make its predictions. However, these alignments alone may not be sufficient to resolve splice forms unambiguously because different transcripts from the same gene may have large proportions of their sequences in common, making it difficult to determine

from which transcript a read originated [101]. Thus reference-guided methods are sub-optimal for organisms with sufficiently well annotated genomes, because additional information from gene models is likely to make predictions more accurate.

Accordingly, Cufflinks was enhanced recently to incorporate gene models into its transcript predictions [173]. To account for annotated transcripts, it simulates "faux reads" from gene models to ensure that known transcripts will be represented in the data. However, this potentially introduces errors as the transcript prediction method must reassemble the faux reads correctly in the presence of real reads that can introduce considerable noise. SpliceGrapher was designed to use annotated gene models to establish a context for interpreting evidence from EST or RNA-Seq alignments, and to predict novel AS events only when the evidence for them is strong [179]. Our experiments showed that this conservative approach predicts fewer novel AS events than tools that predict full mRNA transcripts, but makes predictions with much higher precision. It is worth noting that even mature gene models may contain errors, so a tool that relies on them may predict novel transcripts when RNA-Seq alignments conflict with the annotations. However, curated gene models are the closest thing to a "gold standard" for transcriptomes and they should continue to improve as more data become available.

In this thesis we focus on methods that can update gene annotations automatically using a reference genome, extant gene annotations and RNA-Seq data. Because they use more information than *de novo* transcript assembly or reference-guided methods, we expect these methods to provide the highest accuracy possible in predicting novel exons or transcripts. The state-of-the-art algorithms in this domain are: Cufflinks [218], SLIDE [111], IsoLasso [113] IsoLasso/CEM [114], and iReckon [140]. Below we describe each of these methods in detail.

2.2.1 Cufflinks

Cufflinks focuses on precision in identifying isoforms by using a parsimony strategy that attempts to identify a minimum set of isoforms that explain all the reads in a data set [218]. To identify a minimal set of isoforms, Cufflinks builds an *overlap graph* of compatible reads where compatibility depends on spliced reads (Figure 2.4): if part of a read overlaps an intron inferred by another (spliced) read, then the two reads are incompatible. For each overlap graph, Cufflinks finds a *minimum path cover* to infer a set of paths that represent isoforms. To incorporate gene models into a prediction, Cufflinks uses gene models to generate "faux-read" alignments that cover each of the splice forms in the gene models [173]. It then includes these

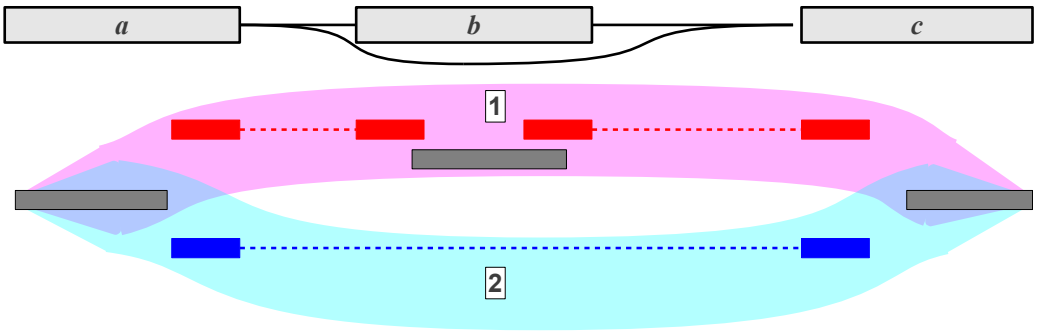

Figure 2.4: Cufflinks builds an overlap graph based on the way reads align within a region. Dark gray bars represent ungapped read alignments and colored bars connected with dashed lines represent spliced alignments. Here the reads in group 1 are incompatible with the reads in group 2 because part or all of each read in group 1 falls within the splice junction in group 2. The remaining read alignments are compatible with either group. In this case the overlap graph predicts the two paths illustrated in the splice graph at the top of the figure.

faux alignments together with real Tophat alignments to predict a set of isoforms. Cufflinks uses these isoforms in a second phase to predict transcript expression levels.

Trapnell et al. used Cufflinks to study transcriptional changes in mouse myoblast cell lines [218]. In part, they identified 70 genes in which the prevalent splice forms changed as cells transitioned from myocyte production to myotube fusion. In another study, Twine et al. used Cufflinks to find statistically significant differences in the expression of alternative splice forms between normal and diseased brain cells that yielded insights into the progression of Alzheimer's disease [219].

2.2.2 SLIDE

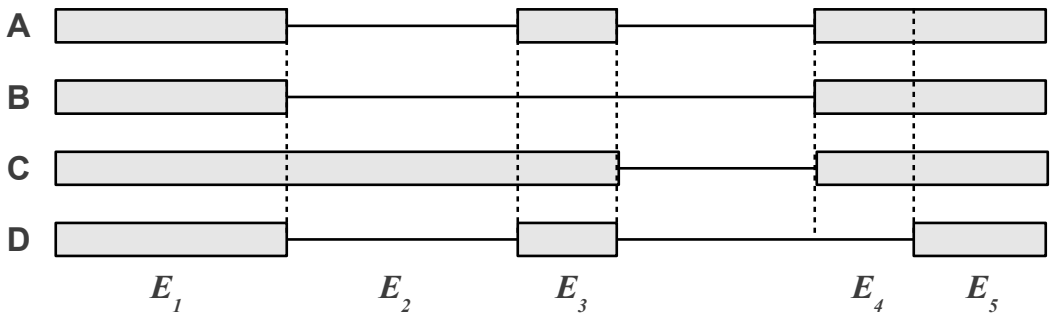

Figure 2.5: SLIDE uses *sub-exons* to model overlapping alternative exons in a gene. Reads that map to a sub-exon provide evidence for all of the transcripts in which it participates. Here, the transcripts $A - D$ are split into sub-exons E_1-E_5. E_2, E_3 and E_4 are used only in a few cases, while the remaining sub-exons appear in all transcripts.

An alternative method called "sparse linear modeling of RNA-Seq data for isoform discovery and

abundance estimation" (SLIDE) uses two forms of linear regression to discover mRNA isoforms and estimate their expression [111]. SLIDE models the relationship between RNA-Seq reads and transcripts as a probability of observing the reads in a data set, given a combination of transcripts. Transcripts are defined as sequences of distinct *sub-exons* identified from the overlapping regions of alternative exons in a gene (Figure 2.5). For a gene with n exons, the SLIDE method considers all $2^n - 1$ possible combinations when detecting the isoforms present in a data set.

Because the number of possible isoforms is usually much larger than the number of sub-exons, the authors use the Lasso method [213] to achieve sparsity in the number of predicted isoforms. Lasso is a regularized regression method for cases in which the number of parameters to be estimated exceeds the number of observations and where most of the parameters are expected to be zeros. Given a set of N observations $(X_i, y_i), 1 \leq i \leq N$, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{iJ})$ are J-dimensional independent variables and y_i are dependent variables, the Lasso estimate $\hat{\beta}$ is given by:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} (y_i - \sum_{j=1}^{J} \beta_j x_{ij})^2 + \lambda \sum_{j=1}^{J} |\beta_j|. \tag{2.1}$$

This formulation may be solved using quadratic programming if we assume $\beta_j \geq 0$ for all j. Note that λ controls the number of parameters in the final solution. In SLIDE, the predicted set of isoform proportions $\hat{p}$ is given by:

$$\hat{p} = \underset{p}{\operatorname{argmin}} \sum_{j=1}^{J} (b_j - \sum_{k=1}^{K} F_{jk} p_k)^2 + \lambda \sum_{k=1}^{K} \frac{|p_k|}{n_k}. \tag{2.2}$$

If there are J sub-exons and K possible isoforms for a particular gene, then b_j is the observed number of reads for sub-exon j, p_k is the relative proportion of the k^{th} isoform, n_k is the number of sub-exons in the k^{th} isoform and F_{jk} is the conditional probability of observing reads in the j^{th} sub-exon given that the reads are from the k^{th} isoform. SLIDE was developed and tested with *D. melanogaster* but has never been used to perform AS analysis as part of any other studies. In addition, although it has been used as a baseline for comparison with other methods, its performance falls well below that of state-of-the-art methods [140, 17].

2.2.3 IsoLasso and IsoLasso/CEM

IsoLasso [113] and Isolasso/CEM [114] are two related methods for simultaneously predicting transcripts and their abundances from RNA-Seq data. IsoLasso creates a large set of possible isoforms by enumerating over the possible combinations of *segments* (or sub-exons, see Figure 2.5) in a gene. It then uses the Lasso [213] method to minimize overall error while simultaneously minimizing the number of

expressed isoforms. Specifically, it minimizes the error between estimated and actual read coverage for each isoform, while constraining the number of isoforms:

$$X^* = \underset{X}{\operatorname{argmin}} \sum_{i=1}^{M} (\frac{r_i}{l_i} - \sum_{j=1}^{N} a_{ji} x_j)^2 + \lambda \sum_{j=1}^{N} x_j, \quad \text{s.t. } x_j \geq 0,\ 1 \leq j \leq N. \tag{2.3}$$

Here, X^* is the optimum set of expression levels, where x_j is the expression level for the j^{th} isoform; r_i is the observed number of reads that mapped to segment i, l_i is the length of segment i and a_{ji} is an indicator variable ($a_{ji} = 1$ if segment i is part of isoform j and 0 otherwise).

IsoLasso/CEM replaces the Lasso algorithm with a *component elimination EM* (CEM) algorithm that solves the same optimization problem while also accounting for potential biases in the data [114]. Section 2.2.5 outlines the biases that arise in RNA-Seq sequencing, including the non-uniform distribution of reads across transcripts and the influence of the underlying sequence [112]. To account for these different biases, IsoLasso/CEM uses a statistical model that leads to an expectation maximization (EM) solution in place of the quadratic programming solution used in IsoLasso. Briefly, reads are assigned to isoforms based on an initial estimate of each isoform's abundance and the estimates are then recomputed based on the reads. The CEM aspect of this algorithm combats overfitting by using a probability model that favors solutions with a small number of highly expressed isoforms [114].

2.2.4 iReckon

Another method that simultaneously predicts transcripts and their expression from RNA-Seq data is iReckon [140]. This method progresses in three stages: first it identifies all possible isoforms from gene models and RNA-Seq alignments by constructing a splicing graph for each gene. Next it uses ungapped alignment to realign reads to the isoforms, using alignment scores and paired-end insert sizes to identify the best alignment for multireads. Finally it uses an EM algorithm that determines the isoforms present in the data and computes their abundances simultaneously. This version of EM iteratively estimates each isoform's abundance based on the reads assigned to it, then reallocates reads to isoforms based on alignment scores and estimated expression levels. The method uses a regularization term to limit the number of isoforms in the final solution.

To model unprocessed pre-mRNA, iReckon first creates a pre-mRNA "transcript" for each gene using the entire gene sequence. The average read coverage across this transcript is then used as a baseline. iReckon compares the read coverage across each intron in the gene with the average expression level for

the pre-mRNA for that gene. When the coverage across an intron is significantly different from the pre-mRNA background distribution, it may be added as a retained intron. iReckon predicts only the most highly significant novel retained intron in any gene.

In this work we compare SpliceGrapherXT with three methods that represent different approaches to the transcript prediction problem: Cufflinks [218, 173], IsoLasso [113] and IsoLasso/CEM [114]. Cufflinks is the most popular of these tools, but Cufflinks and IsoLasso have both become standards for comparisons with algorithms that predict novel transcripts from RNA-Seq [113, 140, 236, 179, 132, 214, 80, 17]. All of these methods for updating existing gene annotations attempt to predict complete mRNA transcripts. Cufflinks first predicts a parsimonious set of transcripts and then predicts expression separately. IsoLasso and IsoLasso/CEM predict transcripts and their expression levels simultaneously, using read coverage to guide their predictions of which exons appear together in each transcript. These approaches are elegant, but they are also vulnerable to biases that arise in RNA-Seq data.

2.2.5 RNA-Seq Biases

Next-generation mRNA sequencing introduces biases that can influence the way RNA-Seq reads map to a genome [114, 42, 144, 111, 140]. These tend to fall into one of four categories: positional biases [42, 144, 114], sequence biases [112, 72, 114], mappability biases [193, 114] and unspliced pre-mRNA [140]. Positional biases occur when the distribution of reads across a transcript is non-uniform. For example, using primers that bind to the poly-A sequences characteristic of transcripts' 3' ends will lead to higher read coverage at those ends [144, 246]. Researchers have used positional biases to predict the strand of novel genes associated with a set of reads [246].

Sequence biases occur when the amount of read coverage is influenced by the underlying sequence. For example, Solexa sequencing tends to produce a higher proportion of reads for GC-rich regions [42]. These sequence biases may be particularly problematic for methods that attempt to predict transcript expression levels in the presence of AS. Mappability biases occur when reads fail to align to a genome due to sequencing errors or when they are discarded for aligning to multiple locations (multireads) [193, 144]. Mappability is an important consideration for methods that estimate expression levels from RNA-Seq. Some programs discard multireads rather than risk attributing them to the wrong gene or splice form [67, 54], while other approaches probabilistically allocate multireads to transcripts based on the relative expression levels of the associated genes [144, 32, 148, 106]. Compared with estimates that use only uniquely-aligned reads,

expression estimates that include multireads have been found to be more consistent with expression levels provided by other tools such as microarrays [144, 74].

Finally, samples of mRNA may include some unspliced pre-mRNA. The resulting sequences may map to intron regions, providing spurious evidence of intron retention. All of these forms of bias have important implications for algorithms that estimate transcript expression based on RNA-Seq or use transcript abundances to inform their transcript predictions [193]. Hence methods such as IsoLasso/CEM and iReckon use statistical models to address biases as they make their predictions. As described in Section 2.2.3, IsoLasso/CEM uses a quasi-multinomial statistical model to capture positional, sequence and mappability biases in its predictions [114]. iReckon specifically models unspliced pre-mRNA, sequencing bias and mappability bias as part of an EM approach to predicting transcripts and their abundances (see Section 2.2.4).

2.3 Predicting Transcripts vs. Splice Graphs

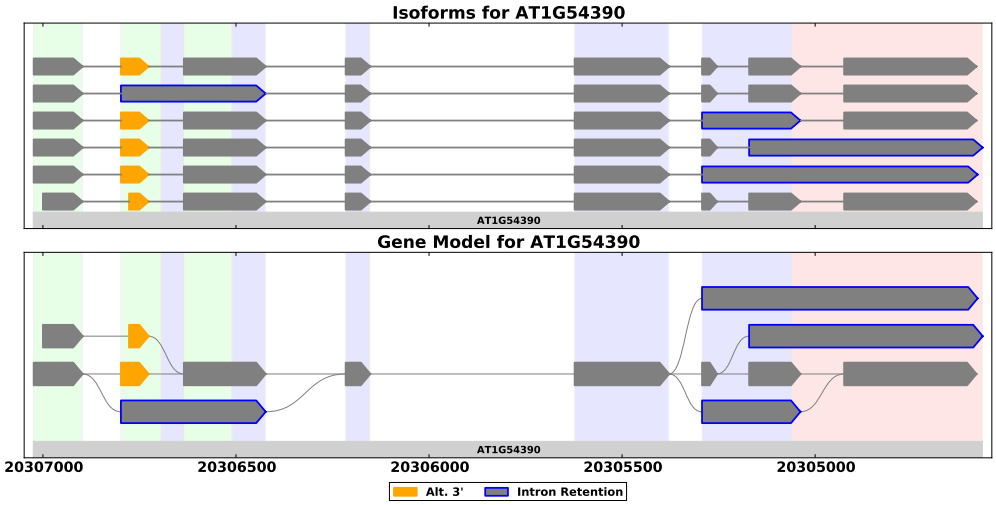

Figure 2.6: Above are the splice forms for a gene from *A. thaliana* shown as a set of transcripts (top) and as a splice graph (bottom). A splice graph is a compact representation that shows all the ways in which a gene's exons may be combined. These plots, generated by *SpliceGrapherXT*, use color coding to highlight alternative splicing events.

Methods for predicting the isoforms that arise from a gene usually attempt to predict complete mRNA transcripts (Figure 2.6). Full-length transcript prediction can provide complete information for updating an organism's gene models. For example, many of the predictions given by TAU using the TAIR9 version of the *A. thaliana* genome were incorporated into the subsequent TAIR10 annotations [208]. In addition, programs that estimate gene or transcript expression levels must have complete transcripts for making predictions [217,

68]. Shorter sequences will provide evidence for AS at specific locations within a gene, but it may be impossible to reconcile AS evidence at one locus with evidence at another locus.

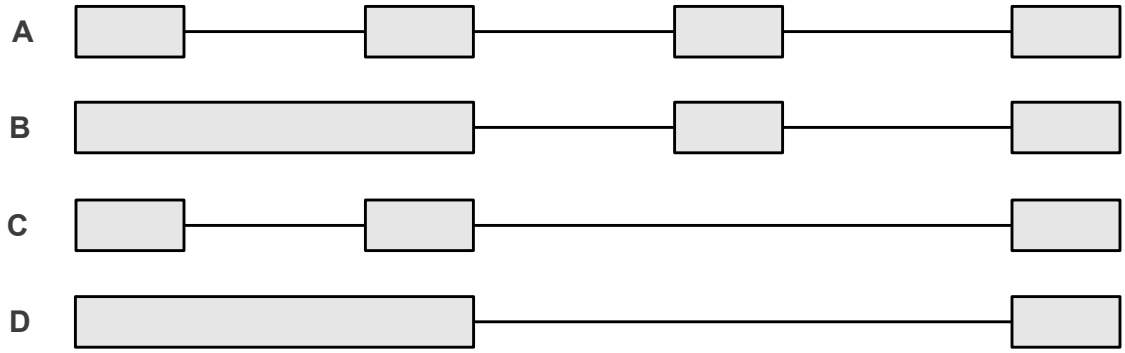

Figure 2.7: This example demonstrates that it may not be possible to ascribe RNA-Seq evidence to specific splice forms. Here, an intron retention event and an exon skipping event are represented in four distinct splice forms. Each splice form may be represented by a unique combination of exons and splice junctions, but it may not be possible to distinguish between the evidence from different combinations of splice forms.

To illustrate the difficulties in predicting full-length transcripts from RNA-Seq data, Figure 2.7 gives an example of a gene with two AS events (intron retention and exon skipping) that result in four splice forms. Here we assume that all four splice forms are present in a comprehensive data set. To predict transcripts, Cufflinks constructs a graph of gene features predicted from ungapped alignments (exons) and spliced alignments (introns). It then uses a minimum path cover algorithm to find a minimal set of paths through the graph that encompass all features. With our hypothetical data set, Cufflinks would select two of the splice forms that satisfy its minimum path cover algorithm, giving the false impression that only two splice forms could be represented in the data. In general this approach may make spurious predictions whenever two or more splice forms share the same elements.

For this reason, some methods focus instead on predicting splice graphs. Splice graph prediction is a fundamental problem in the analysis of RNA-Seq. It is a basis for downstream analysis and provides sufficient information for predicting AS. *Sircah* is an example of a package that can produce splice graphs from conventional EST or cDNA alignments [73]. We enhanced Sircah's AS detection rules and extended the package to provide statistics and protein predictions for genome-wide studies of AS [99, 172], but it is inadequate for making predictions from RNA-Seq data. Consequently we designed SpliceGrapher [179] to integrate RNA-Seq data, annotated gene models and EST alignments to produce comprehensive splice graph predictions. It uses inference rules (Appendix A) to predict AS events that are compatible with all available evidence.

In this thesis we focus on predicting gene architectures using the latest RNA-Seq data, but there is a considerable amount of EST and fl-cDNA data available as well. These data provide valuable evidence for developing accurate splice-site models [250, 192, 128, 179] and for studying transcriptomes [99, 172, 179]. Early studies relied primarily on visual inspection of EST alignments to resolve AS events [70, 194], but methods such as the Transcription Assembly Program (TAP) [88], the Program to Assemble Spliced Alignments (PASA) [69] and Sircah [73] helped to automate transcript prediction from ESTs. Both Sircah [73, 53] and SpliceGrapher [179, 178] can use EST alignments to produce splice graphs. However, Sircah is limited to using EST alignments for its predictions, while SpliceGrapher can use both EST alignments and RNA-Seq data. Thus SpliceGrapher is well positioned to take advantage of sequencing technologies such as RNA-Seq that yield ever-increasing read lengths.

The original SpliceGrapher method achieves high precision in predicting novel AS events, but it may miss some predictions when evidence for multiple AS events occurs in the same region of a gene. Thus we designed SpliceGrapherXT, described in Chapter 3, that replaces SpliceGrapher's AS inference rules with a comprehensive inference method for predicting novel exons. This new method provides much higher sensitivity to novel exons than the original method without sacrificing precision. In Chapter 5 we describe how we can improve sensitivity even further using a procedure for realigning RNA-Seq reads to a putative transcriptome generated from our splice graph predictions. Chapter 6 shows how these accurate splice graph predictions may be leveraged to make full transcript predictions more accurately than state-of-the-art methods. Finally, Chapter 7 describes several projects in which SpliceGrapherXT's predictions have helped to gain insights into biological processes at the forefront of modern transcriptome research.

Chapter 3

Accurate Splice Graph Prediction with SpliceGrapherXT

In our early work on SpliceGrapher, we used separate inference rules to predict each of the four canonical AS types: exon skipping, intron retention, alternative 5' sites and alternative 3' sites (see Appendix A). These rules yield highly accurate predictions [179], but sensitivity is limited because in many cases the separate inference rules cannot disambiguate evidence for different kinds of AS that may occur in the same gene region. To increase SpliceGrapher's sensitivity and to simplify its implementation, we developed a single, comprehensive inference method for predicting novel exons that covers all types of AS events at once. The new method provides dramatically improved sensitivity without sacrificing the accuracy we achieved with our original method.

3.1 The SpliceGrapherXT Pipeline

SpliceGrapherXT is designed to enhance existing gene models using RNA-Seq and EST data, in contrast with tools like Cufflinks [218, 173] or IsoLasso [113] that use existing annotations in a limited way. Relying on existing annotations as a baseline provides SpliceGrapherXT with a context in which to interpret RNA-Seq data. SpliceGrapherXT accepts as input a set of RNA-Seq alignments and annotated gene models, and predicts splice graphs and transcripts based on these data. SpliceGrapherXT's splice graph prediction pipeline consists of the following steps for each gene (Figure 3.1):

1. Align RNA-Seq reads to a reference genome.
2. Filter spliced alignments using splice-site classifiers.
3. Build an initial splice graph from the gene model.
4. Identify clusters of reads that overlap the gene.
5. Use patterns of donor and acceptor sites to predict novel exons.
6. Use spliced alignments to connect exons in the graph.

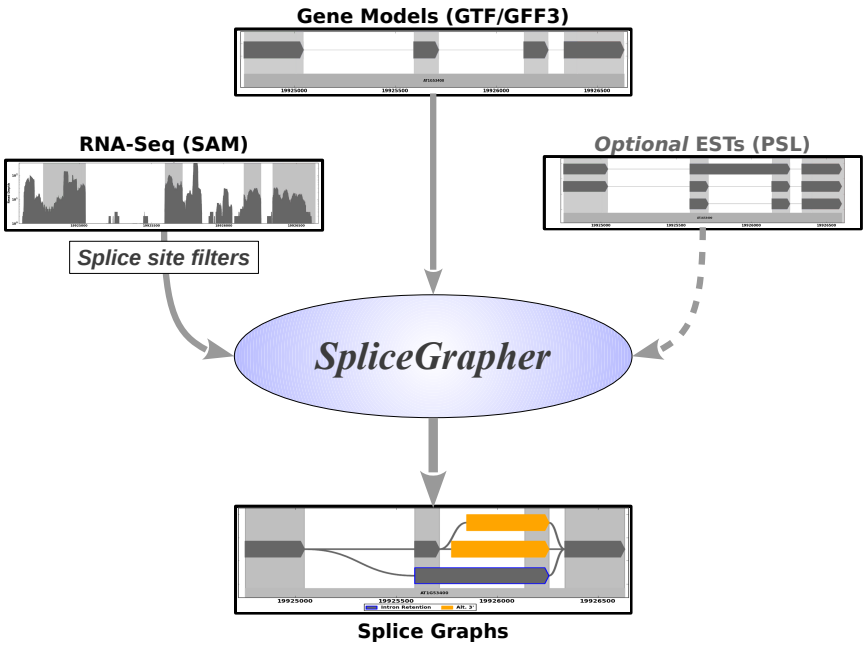

Figure 3.1: The SpliceGrapherXT prediction pipeline. SpliceGrapherXT predicts splice graphs using gene models, EST alignments and RNA-Seq data. RNA-Seq alignments may be provided from any RNA-Seq alignment tool that produces SAM or BAM output, such as Tophat or MapSplice. This is followed by filtering using highly accurate splice-site classifiers. SpliceGrapherXT incorporates the gene models, EST alignments and RNA-Seq alignments to produce a comprehensive splice graph prediction.

7. Save information about ambiguous regions that cannot be resolved.

8. (Optional) Resolve ambiguous regions in the splice graph by realigning the reads to a putative transcriptome.

9. (Optional) Employ PSGInfer or IsoLasso to predict isoforms from the resulting splice graph.

Figure 3.2 provides an example of a splice graph that SpliceGrapherXT predicted for a gene in *A. thaliana*, along with the graph constructed from the original gene model, the splice junctions that were recapitulated in the RNA-Seq data, and the read depth along the genomic region. SpliceGrapherXT combined these different forms of evidence and predicted only those novel exons it could resolve unambiguously.

The key differences between SpliceGrapherXT and the original SpliceGrapher lie in the algorithm used to predict novel exons (step 5 in our pipeline above), a realignment procedure designed to resolve ambiguous regions in the splice graph (step 8, discussed in Chapter 5), and a method for using predicted splice graphs to predict novel transcripts (step 9, discussed in Chapter 6). The original SpliceGrapher inference rules are discussed in detail in Appendix A, while RNA-Seq alignment and filtering (step 2) are discussed in

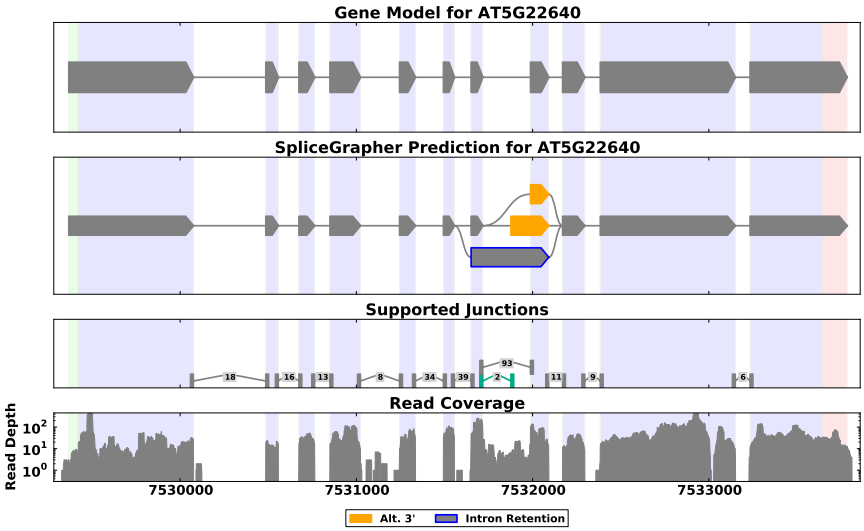

Figure 3.2: Example of a predicted splice graph in *A. thaliana* where RNA-Seq data were loaded along with gene model annotations to create a composite model that incorporates all available evidence. Here RNA-Seq coverage across an intron allowed SpliceGrapherXT to identify an intron retention event (exon outlined in blue). In addition, a novel splice junction (highlighted in green) provided SpliceGrapherXT with evidence for an alternative 3' splicing event (exons highlighted in orange). The numbers associated with splice junctions indicate the number of reads that aligned across them. Vertical bands in the background depict exon boundaries in the original gene model.

Chapter 4. Here we describe each of the remaining steps in the pipeline, beginning with building initial splice graphs from gene models.

3.1.1 Splice Graph Initialization

Each of the data sources used by SpliceGrapherXT—genome annotations, EST alignments and RNA-Seq data—requires a distinct interpretation for splice graph construction. Gene models and EST alignments are treated in a similar fashion and are used to establish the context for interpreting the evidence from RNA-Seq data.

Gene models are usually provided in the Gene Transfer Format (GTF) or the General Feature Format (GFF3) and may be obtained from centralized web repositories such as NCBI [2], UCSC [222] or EMBL [50]. These files provide details on every known exon in a genome, including the chromosome where it is found; its start and end coordinates; its strand, and the transcripts that include it. SpliceGrapherXT creates an initial splice graph from the information found in a gene model file: it incorporates exons directly into a splice graph and infers an intron whenever the corresponding exons are adjacent in a transcript.

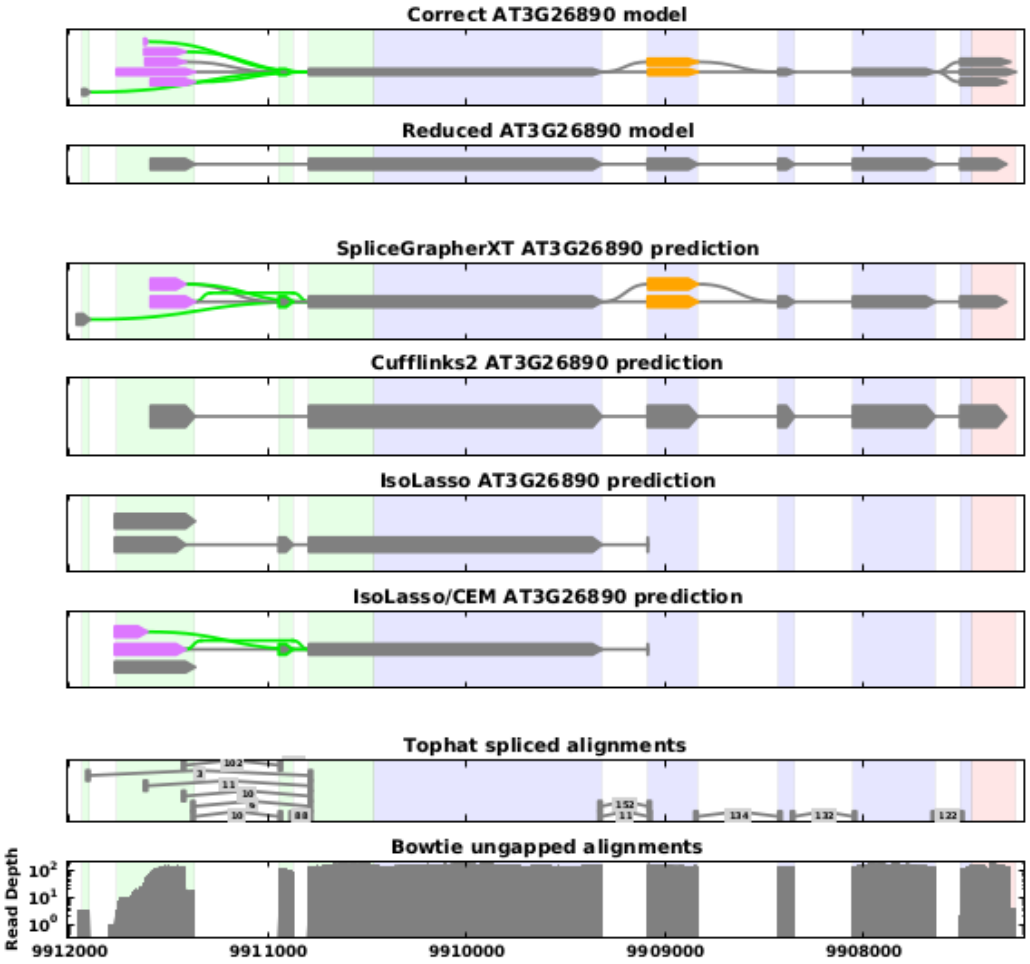

Figure 3.3: Example where evidence for novel exons is ambiguous and difficult to interpret. We simulated reads from the complete gene model, then provided a reduced model to Cufflinks and SpliceGrapherXT to make predictions. Cufflinks fails to make any novel predictions in this example. IsoLasso/CEM outperforms Cufflinks and IsoLasso, correctly predicting several novel exons, but an incomplete isoform. Distinct inference rules in the original *SpliceGrapher* method are able to resolve novel exons for an alternative 5' event and an alternative 3' event, but cannot make predictions for combinations of events. Using a single comprehensive inference rule, *SpliceGrapherXT* is able to resolve more of the novel exons recapitulated in the data and makes a more complete prediction than any other method.

ESTs are longer than RNA-Seq reads and thus provide more reliable transcriptional evidence. These sequences may be obtained from sites such as GenBank [146, 16] and are provided in the FASTA format [121]. Gapped alignment tools such as BLAT [92] or GMAP [238] align ESTs to a genomic reference sequence and store the alignments in Pattern Space Layout (PSL) files. Similarly to gene model files, PSL files provide details on the exons and introns inferred by each gapped EST alignment. SpliceGrapherXT converts this alignment information directly into splice graphs using the same procedures described above for gene models. See Appendix C for more details on the file formats.

3.1.2 Making Predictions from RNA-Seq Data

Figure 3.4: Depiction of the method SpliceGrapherXT uses to convert read coverage into read clusters. The top panel shows the novel read clusters inferred from the read coverage, while the bottom panel shows read coverage across a gene. Vertical shaded regions depict known exons. One region of contiguous coverage falls completely within the second intron (genomic position 4500), while two other coverage regions extend past known boundaries for the third and fourth exons. Each of these regions is converted into a read cluster (top panel) that may be used to make predictions.

To incorporate RNA-Seq data into a splice graph, SpliceGrapherXT loads read alignments in Sequence Alignment/Map (SAM) format [108] for all reads that map within the boundaries of some annotated gene. It then constructs putative exons from clusters of contiguous ungapped alignments (Figure 3.4) where the read depth remains above a minimum threshold (the default is 0). If a putative exon is contained within an existing exon, it is considered redundant and is discarded. SpliceGrapherXT resolves the remaining putative exons by applying a concise set of constraints on the splice junction evidence in the RNA-Seq data.

There are major differences between SpliceGrapherXT and the original SpliceGrapher in the algorithm used to predict novel exons (step 5 above). The original SpliceGrapher focused on predicting alternative splicing (AS) events and thus used separate inference rules for different kinds of AS. These isolated rules

are sometimes unable to resolve all the evidence for a gene, as shown in Figure 3.3. SpliceGrapherXT predicts novel exons using a cohesive set of constraints that derive from the following observation: when two or more acceptor sites appear upstream of two or more donor sites in the same cluster, the choice of exon boundaries is not well defined. We specify these criteria as two constraints imposed on the splice sites within any cluster of reads before we will predict an exon:

1. At most one donor site can be downstream of multiple acceptor sites.

2. At most one acceptor site can be upstream of multiple donor sites.

Each of these constraints must hold for SpliceGrapherXT to make a prediction.

The RNA-Seq spliced alignments that fall within a genomic region form a sequence of confirmed donor and acceptor sites. To apply the above criteria to these sequences, we use the symbols a and d to denote acceptors and donors, respectively, and convert sequences of confirmed splice sites within a read cluster into strings of a's and d's. For every such string, we express the constraints above as follows:

1. At most one a precedes one or more d's, which may be expressed using a pattern of the form a^+d.

2. At most one d follows one or more a's, which may be expressed using the pattern ad^+.

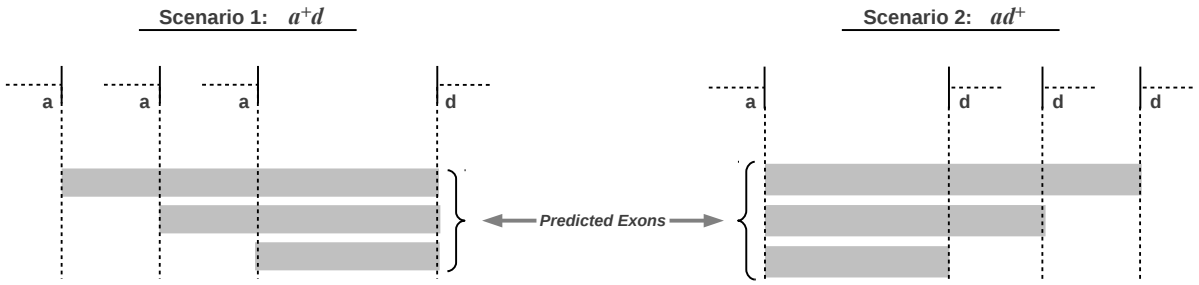

Figure 3.5: Correspondence between regular expression rules for sequences of acceptor and donor sites within read clusters, and the exon predictions that follow. Scenario 1 predicts splice sites at the 3' end of a putative exon, while scenario 2 predicts splice sites at the 5' end. We combine these expressions to predict the four primary AS events.

We define a simple regular language to recognize strings that obey these constraints and thus use a regular expression to identify sequences of acceptor and donor sites that meet our requirements (see Figure 3.5). For those sequences accepted by the regular expression, predicting novel exons then becomes a simple matter of pairing every acceptor site with every donor site within the corresponding sequence of splice sites. The original implementation applies these constraints implicitly in the way it resolves putative exons [179].

Using a regular language allows us to make these constraints explicit and eliminates the need for having separate inference rules for each type of AS event.

3.1.3 Overview of the Prediction Algorithm

Algorithm 1: SG_Predict($G, C, J, \mathcal{R}$)

Input: splice graph $G = (V, E)$ initialized from gene models and ESTs
read clusters C that overlap G
splice junctions J that overlap G
regular language $\mathcal{R}$

Output: updated graph G
unresolved nodes U

```
Novel exons N ← ∅
Unresolved exons U ← ∅
for all regions r ∈ C ∪ V do
  Create site string S from the donor and acceptor sites within r
  for all acceptors a ∈ r do
    for all donors d ∈ r downstream of a do
      Create exon e_ad with boundaries [a, d]
      if e_ad ∉ V then
        if R accepts S then
          N ← N ∪ e_ad
        else
          U ← U ∪ e_ad
        end if
      end if
    end for
  end for
end for
for all e_ad ∈ N do
  add edges using junctions that end with a or start with d
end for
V ← V ∪ N
```

The core of the SpliceGrapherXT method is the algorithm for updating splice graphs (Algorithm 1). Given an initial splice graph G, a set of read clusters C that overlap the gene, a set of confirmed splice junctions J that overlap the gene, and a regular language $\mathcal{R}$ that accepts resolvable patterns of acceptor and donor sites, the algorithm adds novel exons to the graph and stores unresolved exons for later processing. The algorithm iterates over every exon in a graph, plus every cluster that is not contained within an exon. Within these regions it finds all acceptor and donor sites associated with splice junctions from the gene models and from the filtered RNA-Seq alignments. Using the relative positions of acceptor and donor sites,

it creates a splice site string. It then creates exons from pairs of acceptor and donor sites and adds those not in the graph as novel or unresolved exons, depending on whether the regular language $\mathcal{R}$ accepts the splice site string. Finally the algorithm resolves the edges associated with novel exons and returns the updated graph along with any unresolved exons.

By running this procedure for every gene in an organism, SpliceGrapherXT can generate splice graph predictions for a complete genome-wide assessment of AS. The graphs can help to generate biologically-relevant insights by facilitating statistical analysis [99, 172], and by providing visualizations that are easy to interpret [73, 179].

3.2 Performance Evaluation

In this chapter we evaluate SpliceGrapherXT and other methods at the exon level. It is challenging to validate methods that predict novel exons because there is no "gold standard" for the exons in a genome. Even well-annotated gene models are likely to be missing exons and transcripts that simply have not yet been observed. Thus to evaluate prediction methods we use two similar approaches that we apply to simulated data and real data [140, 177, 9]: A subset of splice forms are removed (or "left out") from a set of gene models, and the reduced models are provided to each method to make predictions. Predictions are then evaluated using the left-out set that we also call our test set. This approach is a more formal version of an experiment we reported in [179] in which we made predictions for *A. thaliana* using one version of the gene models (TAIR9) and compared those predictions with a more recent version of the gene models (TAIR10).

Simulating reads provides us with a suitable "gold standard" of splice forms and exons that are represented in a test set, allowing us to compute recall, precision and accuracy for each method. Real data poses more significant challenges: first, there may be many splice forms in real data that are unknown to the gene models. Second, real data is usually noisier than simulated data and may contain errors or biases related to the sequencing process [152, 86]. However, we can use a similar approach to the one we use for simulated data. Again we create a left-out set, provide reduced gene models to each prediction method, and then estimate precision and recall for their predictions relative to splice forms in the left-out set. Throughout this thesis we use these two approaches to compare SpliceGrapherXT with Cufflinks and IsoLasso.

In all of our experiments described in this chapter we run steps 1 through 7 in the SpliceGrapherXT pipeline described in Section 3.1. This includes a filtering step that helps to make SpliceGrapherXT more precise by removing novel splice junctions that may be spurious. However, it may also reduce

SpliceGrapherXT's sensitivity if it removes false-negative junctions. The filtering step is described in detail in Chapter 4, where we also compare our results with and without filtering.

We ran our comparisons on two different species: human and the model plant *A. thaliana* that has a small, compact genome with short introns. Although both species have well-annotated gene models, *H. sapiens* has much more complex gene models with many more isoforms per gene: on average, 6.85 annotated splice forms per gene compared with 1.24 per gene for *A. thaliana*.

3.2.1 Methods for Comparison

We selected Cufflinks [218, 173], IsoLasso [113] and IsoLasso/CEM [114] as competitive methods for comparison in our experiments. Cufflinks and IsoLasso have become *de facto* standards for comparisons with algorithms that predict and quantify novel transcripts from RNA-Seq [113, 140, 236, 179, 132, 214, 80, 17]. In earlier work on *A. thaliana* we also compared SpliceGrapher with TAU [55], because it was developed for *A. thaliana* (in contrast to Cufflinks that was developed using mammalian genomes). However, TAU tends to produce a large number of potentially spurious predictions [179], so we no longer consider it a competitive comparison.

As described in Chapter 2, Cufflinks uses a parsimony strategy that attempts to identify a minimum set of isoforms that explain all the reads in a data set. When provided with gene models, Cufflinks combines Tophat alignments with faux-read alignments based on the gene models to predict a set of isoforms. IsoLasso and IsoLasso/CEM use methods that construct isoforms and estimate their abundances simultaneously [113, 114]. IsoLasso creates a large set of possible isoforms by enumerating over the possible combinations of exons inferred from read alignments or imported from gene models. It minimizes the sum of squared errors between estimated and actual read coverage for each isoform, while simultaneously constraining the number of isoforms. IsoLasso/CEM solves a similar optimization problem while also accounting for potential biases in the data [114]. These IsoLasso methods will accept gene models to augment the exons recapitulated in a data set, but we found that both methods make better predictions without gene models.

Both Cufflinks and IsoLasso have parameters that require tuning. Cufflinks uses a threshold on the predicted expression level to determine when to predict a transcript. The threshold is given as a fraction of the most abundant isoform for a gene, below which other isoforms will not be predicted. Set to 0, Cufflinks will predict any isoform with read coverage and should provide maximum sensitivity; we also found that when we set it to 10% (0.1), Cufflinks achieves a good balance between recall and precision. We refer to

these two settings as *Cufflinks*$_S$ (*sensitive*, threshold 0) and *Cufflinks*$_B$ (*balanced*, threshold 0.1). IsoLasso performance varies primarily with two parameters: a minimum expression value cutoff (`minexp`) similar to Cufflinks' threshold, and a coverage fraction cutoff (`u`) that controls its sensitivity to multiple isoforms. After extensive testing on simulated data we achieved the best performance using `minexp=0.0034` and `u=0.98` for IsoLasso and `minexp=0.002` and `u=0.78` for IsoLasso/CEM.

3.2.2 Simulation experiments

In our simulation experiments we generate paired-end reads, align them to a reference genome using Tophat, and provide the alignments to each method to make its predictions. We use an organism's complete set of gene models as the reference for generating the paired-end reads, then select a single representative transcript from every gene as input to the prediction methods. We then compare each method's predictions with the set of left-out transcripts to assess performance. Preliminary experiments showed that providing SpliceGrapherXT and Cufflinks with multiple isoforms instead of a single isoform led to similar performance, so we chose to highlight these methods' success in this somewhat more challenging task. As noted in Section 3.2.1, IsoLasso performed better without gene models.

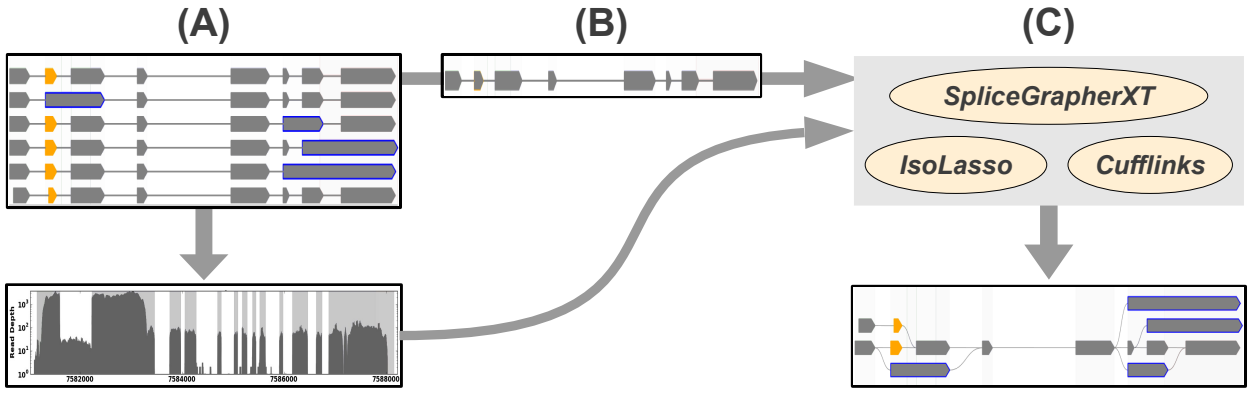

Figure 3.6: Experimental protocol for our simulation experiments. (A) We provide a complete set of gene models to the FluxSimulator to simulate RNA-Seq reads for each transcript. (B) Next we select a single representative transcript for the gene and provide it to the prediction methods as a baseline. (C) Each method then makes its predictions based on the representative transcript and the simulated reads.

In these experiments (Figure 3.6) we randomly select 1000 genes that produce multiple isoforms and use the FluxSimulator [66] to generate 1 million paired-end reads (500,000 pairs) from those genes. We then use each method's pipeline to generate predictions. The SpliceGrapherXT pipeline uses Tophat [216] for RNA-Seq alignments and then filters the output. The Cufflinks and IsoLasso pipelines use Tophat for RNA-Seq alignments and make predictions from alignments without any filtering steps. We have compared

Tophat with MapSplice [230] and found that MapSplice yields higher sensitivity to novel splice junctions than Tophat. However, Cufflinks does not perform well with MapSplice output, as it relies on meta-data that Tophat includes in its alignments. In addition, we wanted to focus our comparisons on exon prediction methods without any possible bias due to the choice of alignment algorithms. Hence we selected Tophat for all alignments in our experiments.

The FluxSimulator provides a detailed simulation for each step in the RNA-Seq generation process [66]. It applies a modified version of Zipf's Law [59] to randomly assign an expression level to each transcript in a gene. This means that a couple of isoforms may be highly expressed in a gene, while other isoforms may have expression levels that are orders of magnitude lower. Thus some exons from the gene models may have no read coverage and may be impossible to predict from the generated reads. With this in mind, our test set includes any exon from the left-out set that can be identified from the simulated read information. All simulated reads have headers that include information about where the read originated. This allows us to determine which sequences, and thus which exons, are represented in the simulated reads.

3.2.3 Real data

Evaluation on real data is challenging because gene model annotations are incomplete and because we have no *a priori* knowledge of which genes and transcripts are represented in an RNA-Seq data set. As in our simulation experiments, we provide Cufflinks, SpliceGrapher and SpliceGrapherXT with a representative isoform from each gene, and then compare all methods' predictions with the rest of the isoforms.

As with our simulated data, to assess each method's performance we establish test sets of exons and isoforms by removing all but one isoform from every gene model. We then use Cufflinks to determine the isoforms from the left-out set that are represented in the data. To do this, we provide Cufflinks with the complete set of gene models and ask it to estimate expression levels only for isoforms from the gene models (i.e., without predicting any novel splice forms). Cufflinks provides 95% confidence intervals on the estimated expression levels that we can use to determine which isoforms are likely to be expressed in the data. Our test set thus includes just the isoforms whose estimated expression levels have strictly positive lower bounds.

To evaluate our precision results for real data in more detail, we also look at how the exons predicted by each method are distributed between those found in the complete gene models (test set) and those that are novel or possibly spurious predictions (unknown). For well-annotated species like human, we expect that a

higher proportion of predicted exons should come from the left-out set. For *A. thaliana* that is more sparsely annotated, we expect a higher proportion to be novel.

We used RNA-Seq data sets with read lengths of at least 75nt to reflect the current technology. Our reads come from two different species: *H. sapiens* and the model plant *A. thaliana*. For *H. sapiens* we downloaded RNA-Seq data that were used as a control for comparison with CLIP-seq reads[240] (NCBI accession SRP009861, sample SRX111920). This control set consists of 28M 75-nt read pairs (56M reads) sequenced from RNA isolated from untreated HeLa cells. The authors used these data explicitly to detect alternative splicing in those cells. For *A. thaliana* we used 76-nt data consisting of approximately 32.5M read pairs (65M reads, NCBI accession SRA047499) [135]. We expected these reads to yield good results as they were derived from a cDNA library that was normalized to increase coverage across genes.

3.3 Results

For each method we report the number of true positive predictions, precision and recall. In the case of real data however, precision is under-estimated, since a prediction that is not part of the known gene models may be a novel exon or an isoform that is as-of-yet, unknown. This phenomenon is more pronounced in *A. thaliana*, whose genome is not as well annotated as the human genome.

3.3.1 Simulation experiments

Table 3.1: Performance averaged over 10 simulation runs for *H. sapiens* (left) and *A. thaliana* (right) with 1,000 randomly-selected AS genes. Approximately 1 million paired-end reads were generated for each run (500,000 pairs). Gene models were reduced to a single isoform before being supplied to each method. *Reference* shows the number of exons present in the full gene models for the randomly selected genes; the *test set* shows the number of exons removed from gene models. Cufflinks$_S$ is Cufflinks at the sensitive setting that reports any transcript, while Cufflinks$_B$ reports transcripts with the threshold set to 0.1. Results show the number of correctly predicted exons (TP), and the recall and precision for each method.

	H. sapiens			*A. thaliana*		
Method	TP	Recall	Precision	TP	Recall	Precision
Reference	19,956	—	—	10,168	—	—
Test set	9,349	—	—	2,623	—	—
SpliceGrapher	573	0.06	**0.95**	816	0.31	**0.96**
SpliceGrapherXT	1,053	**0.11**	0.93	1,207	**0.46**	0.95
Cufflinks$_S$	590	0.06	0.32	620	0.24	0.32
Cufflinks$_B$	574	0.06	0.31	571	0.22	0.33
IsoLasso	550	0.06	0.13	520	0.20	0.22
IsoLasso/CEM	546	0.06	0.12	607	0.23	0.24

In our experiments on simulated data, SpliceGrapherXT achieves approximately twice the recall of any other method and three times the precision at the exon level (Table 3.1). Cufflinks$_S$ achieves higher exon recall than Cufflinks$_B$ in both species, with approximately the same precision. IsoLasso's recall is slightly lower than Cufflinks, and with much lower precision. In general we found that IsoLasso had difficulty constructing any novel isoforms even when we provided it with gene models, and overall its performance was better without them. Our experiments also illustrate the improvement of SpliceGrapherXT over the original SpliceGrapher as it yields much higher recall with nearly the same precision. A representative example is shown in Figure 3.3.

Table 3.2: Prediction results on simulated data for the four kinds of alternative splicing. Overall, SpliceGrapherXT is able to predict correctly up to 33% more AS events than SpliceGrapher, with the most dramatic improvement in IR events. SpliceGrapherXT is able to recall fully 69% of the left-out AS events in *A. thaliana* and more than 21% of events in *H. sapiens*. The two SpliceGrapher methods have much higher precision than any of the other methods.

AS Statistics for 1,000 Genes (*A. thaliana*)										
	IR		ES		Alt. 5'		Alt. 3'		Overall	
Method	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
SpliceGrapher	0.33	**0.98**	0.54	**0.98**	0.54	**0.98**	0.65	**0.99**	0.52	**0.98**
SpliceGrapherXT	**0.65**	0.95	0.70	**0.98**	**0.67**	**0.98**	**0.73**	**0.99**	**0.69**	0.97
Cufflinks$_S$	0.51	0.24	**0.88**	0.91	0.62	0.19	0.69	0.27	0.64	0.27
Cufflinks$_B$	0.34	0.41	0.62	0.98	0.40	0.61	0.45	0.80	0.43	0.64
IsoLasso	0.03	0.18	0.45	0.49	0.17	0.30	0.22	0.44	0.18	0.38
IsoLasso/CEM	0.04	0.18	0.62	0.63	0.24	0.35	0.37	0.53	0.28	0.46

AS Statistics for 1,000 Genes (*H. sapiens*)										
	IR		ES		Alt. 5'		Alt. 3'		Overall	
Method	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
SpliceGrapher	0.04	**0.96**	0.27	0.95	0.12	**0.94**	0.12	**0.93**	0.17	**0.95**
SpliceGrapherXT	**0.07**	0.86	0.32	0.93	**0.13**	0.93	0.13	**0.93**	**0.21**	0.93
Cufflinks$_S$	**0.07**	0.37	**0.33**	0.86	0.11	0.17	**0.16**	0.21	**0.21**	0.42
Cufflinks$_B$	0.05	0.57	0.22	**0.99**	0.07	0.50	0.10	0.61	0.14	0.79
IsoLasso	0.00	0.12	0.18	0.38	0.03	0.20	0.05	0.30	0.09	0.35
IsoLasso/CEM	0.00	0.11	0.20	0.41	0.03	0.22	0.06	0.32	0.11	0.37

We find a similar pattern when we break performance down across AS types: SpliceGrapherXT again achieves higher recall and precision than any other method, predicting more AS events than the original SpliceGrapher at nearly the same level of precision (Table 3.2). Of note is the fact that SpliceGrapherXT's recall for IR events is approximately twice as high as SpliceGrapher IR events in both species. As noted in [179], IR is arguably the most difficult form of AS to predict from RNA-Seq. By contrast, IsoLasso and

IsoLasso/CEM yield very low recall and precision for IR events. IsoLasso/CEM achieves higher precision, but neither method achieves even half the recall or precision of SpliceGrapherXT. Cufflinks$_S$ recalls more ES events in both species than SpliceGrapherXT, and more alternative 3' events in *H. sapiens*, but it suffers from extremely low precision because it uses unfiltered spliced alignments (see Chapter 4). Cufflinks$_B$ raises Cufflinks' precision considerably, but at the cost of lower recall. Cufflinks' susceptibility to parameter settings is especially noticeable for alternative 3' and alternative 5' events, where raising the threshold from 0 to 0.1 can increase precision by a factor of three but decreases recall by up to 60%.

In our simulations, recall and precision are both higher for AS events than for exons overall. Recall is considerably higher because approximately 75% of the missing exon predictions in our simulations (counted as false-negatives) are not counted as AS (see Appendix B for details on counting AS events). These typically are alternative initial or terminal exons that are difficult to predict. Precision also increases slightly as a small proportion (15%) of false-positive exon predictions also are not counted as AS. Similar to the false-negatives, these are due to incorrectly predicted initial or terminal exons.

3.3.2 Real data

Table 3.3: Prediction performance for all methods on real data for *H. sapiens* (left) and *A. thaliana* (right). Shown are the number of exons removed from the original gene models (left out) followed by the statistics for each method.

	H. sapiens			*A. thaliana*		
Method	TP	Recall	Precision	TP	Recall	Precision
Left out	87,799	—	—	14,617	—	—
SpliceGrapher	2,173	0.025	**0.67**	2,246	0.15	**0.22**
SpliceGrapherXT	3,965	**0.045**	0.59	2,413	**0.17**	0.16
Cufflinks$_S$	2,006	0.023	0.16	1,636	0.11	0.02
Cufflinks$_B$	2,015	0.021	0.10	996	0.07	0.04
IsoLasso	1,308	0.013	0.04	890	0.06	0.03
IsoLasso/CEM	1,359	0.014	0.04	990	0.07	0.03

We tested the ability of each of the methods to identify novel exons and isoforms with real paired-end RNA-Seq data using 75-nt *H. sapiens* reads [240] and 76-nt *A. thaliana* reads [135] (see Table 3.3). In *H. sapiens* SpliceGrapherXT predicts almost twice as many of the left-out exons as either of the Cufflinks runs, with up to six times the precision. In *A. thaliana* SpliceGrapherXT predicts nearly 50% more of the left-out exons than Cufflinks with up to eight times the precision. IsoLasso and IsoLasso/CEM both exhibit the worst performance of all the methods, predicting less than half as many left-out exons as SpliceGrapherXT in either species, and at very low precision.

Table 3.4: Novel exons predicted by each method for *A. thaliana* and *H. sapiens*, where exons are novel relative to the reduced gene models. Predictions are split into two categories: those in the test set (complete gene models) and those that are either novel or spurious predictions (unknown). SpliceGrapherXT predicts more exons in the test set than the other methods. Cufflinks predicts up to 20 times as many unknown exons as SpliceGrapherXT. The IsoLasso methods perform as poorly as Cufflinks, but with fewer novel exons overall.

Category	SpliceGrapherXT	Cufflinks$_S$	Cufflinks$_B$	IsoLasso	IsoLasso/CEM
A. thaliana					
Test set	2,413	1,636	996	890	990
Unknown	13,184	104,350	13,187	35,152	37,433
H. sapiens					
Test set	3,965	2,006	2,015	1,308	1,359
Unknown	2,717	55,273	17,875	31,971	32,578

To investigate these results in more detail, we take a closer look at the distributions of predicted exons for each method (Table 3.4). For each species we present the number of novel predicted exons that were in the test set (analogous to true positives) and the number that were either false positives or as-yet-unknown. As noted above, we expect *H. sapiens* to be more completely annotated than *A. thaliana*. Each of the methods' predictions are consistent with this hypothesis. In *H. sapiens* SpliceGrapherXT predicts a majority of exons from the test set while in *A. thaliana* there are a majority of unknown exons. Both Cufflinks versions also predict a higher proportion of left-out exons in *H. sapiens* than *A. thaliana*, though Cufflinks$_S$ predicts vast numbers of unknown exons in both cases. IsoLasso and IsoLasso/CEM, which had the worst performance, also predict a higher proportion of left-out exons in *H. sapiens* than *A. thaliana*.

3.4 Conclusion

We developed SpliceGrapher, a Python package designed to enhance existing gene annotations by predicting splice graphs from RNA-Seq and EST data. We replaced our initial SpliceGrapher implementation with SpliceGrapherXT, a novel method that predicts AS from RNA-Seq data by applying a simple set of constraints to patterns of acceptor and donor sites. Our results show that this approach gives SpliceGrapherXT much greater sensitivity than the original SpliceGrapher when predicting novel AS events, without sacrificing precision. In addition, SpliceGrapherXT is efficient: in our experiments with real data from *H. sapiens* and *A. thaliana* we were able to generate a complete set of predictions for either species in little more than an hour.

We compared SpliceGrapherXT to Cufflinks and IsoLasso; where the other methods make limited use of gene models, SpliceGrapherXT uses gene annotations as a basis on which to construct its splice graphs.

This allows SpliceGrapherXT to make meaningful predictions even for genes that have low read coverage, and helps it resolve AS events that are otherwise hard to detect from RNA-Seq data. When patterns of acceptor and donor sites are unresolvable, SpliceGrapherXT stores the information for later evaluation.

Both Cufflinks and IsoLasso were sensitive to the parameter settings we selected, so both required extensive tuning to obtain the best performance on all of our data sets. SpliceGrapherXT has parameters that it uses to filter incoming alignment data, such as a read coverage threshold for accepting read clusters or spliced alignments, but these do not impact its performance greatly, and we were able to use the default settings for all experiments.

Prediction of AS requires accurate alignment across splice junctions, which is a difficult task, especially when the junction falls near one end of a read. In the experiments in this chapter, SpliceGrapherXT included a filtering step that improved its precision by removing novel splice junctions that were possibly spurious. In the next chapter we will see how SpliceGrapherXT uses a machine learning approach to recognize spurious junctions and to filter the output of spliced alignment tools. In later chapters we explore ways to resolve some cases of unresolvable splice site patterns by realigning the RNA-Seq reads to a set of putative transcripts. We will also look at how SpliceGrapherXT's predictions may be used to predict complete mRNA transcripts.

Chapter 4

Filtering Spliced Alignments

In Chapter 3 we showed that SpliceGrapherXT can make accurate and sensitive splice-graph predictions from RNA-Seq evidence. However, because SpliceGrapherXT bases its novel AS predictions largely on evidence from spliced alignments, its predictions will be only as accurate as the alignments themselves (see Figure 4.1). Our goal, then, is to ensure that the spliced alignments we provide to SpliceGrapher are as accurate as possible. To achieve this, we develop splice-site models that can discriminate between real and spurious splice junctions and use these models to filter spliced alignments.

4.1 Related Work

Reads that span splice junctions create significant challenges for alignment tools. A splice junction may occur anywhere within a read, so the read may have just a few bases on one side of a junction. Such a short sequence may align in multiple locations, making it difficult to identify its true origin. Many alignment tools use heuristics to restrict the number of candidate locations: for example, by establishing limits for permissible intron lengths, or by focusing on locations that are bounded by canonical (GT-AG) or semi-canonical (GC-AG) splice-site dimers [10, 12, 216, 23, 79].

Heuristics that look for GT-AG and GC-AG junctions can work reasonably well, as up to 99% of splice junctions in mammals are flanked by these dimers [25]. Coupled with an organism-specific distribution of known intron lengths, and anchor sizes of 8-10nt, the chances of a spurious alignment become relatively small. Our experiments have shown that Tophat [216] can report correct splice junctions up to 97% of the time [179], but even a 3% error rate with tens of thousands of spliced reads can result in hundreds of spurious junctions. Further, if we consider only novel spliced alignments, Tophat's accuracy drops as low as 32% [179]. It is these novel splice junctions we require to make novel exon predictions, so we need more accurate splice-site models than these simple heuristics to discriminate between real and spurious splice junctions (Figure 4.1).

The first studies that filtered RNA-Seq data for predicting novel transcripts used splice-site models that were developed for a specific set of experiments [153, 246]. For example, Pan et al. developed a logistic

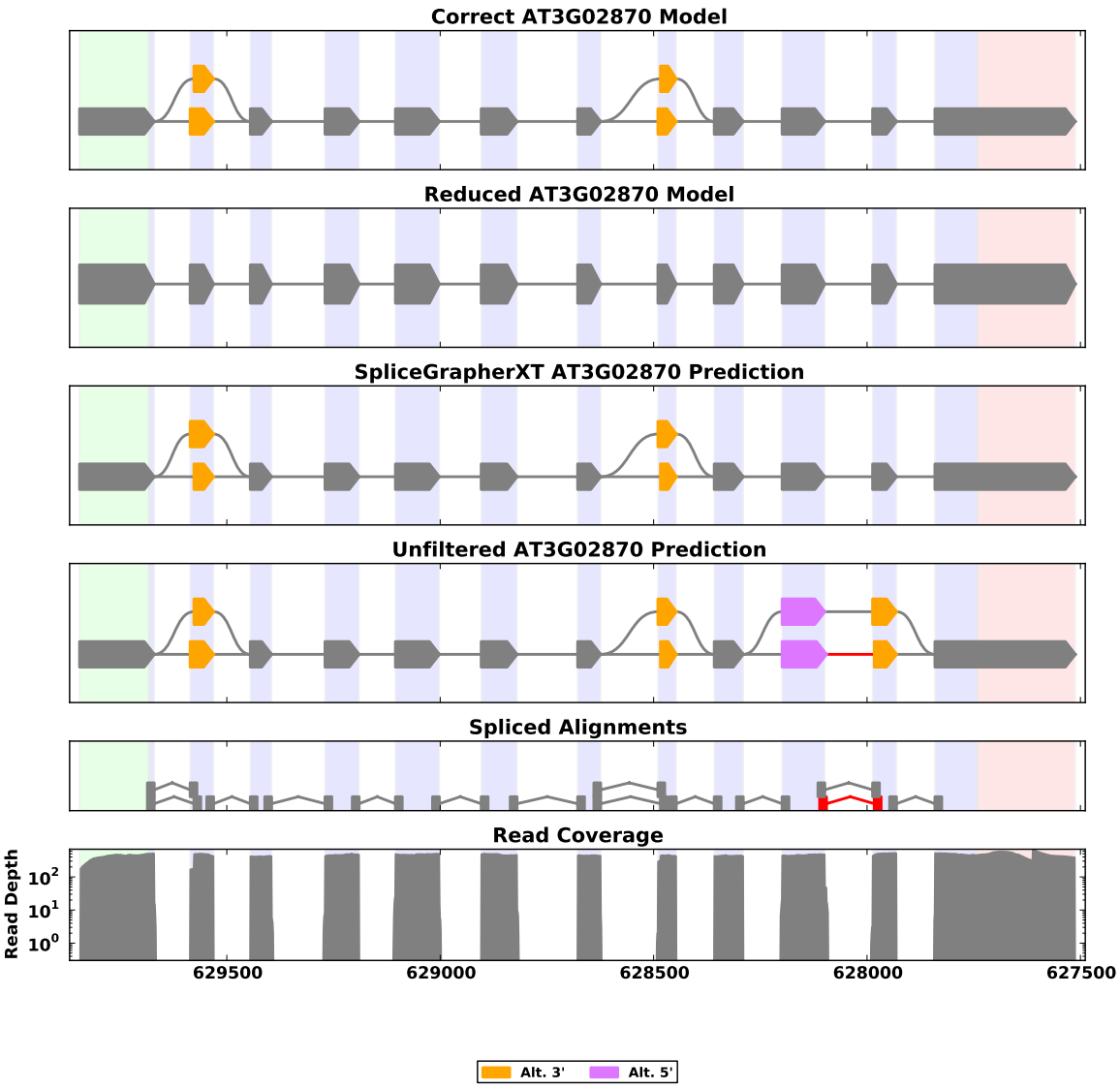

Figure 4.1: Example of a SpliceGrapherXT prediction based on unfiltered spliced alignments for simulated RNA-Seq data. By using simulated reads we can identify real and spurious junctions in the alignments. With filtered alignments, SpliceGrapherXT makes the correct prediction (third panel), but a false-positive splice junction in the unfiltered data (fourth and fifth panels, highlighted in red) causes SpliceGrapherXT to make a spurious prediction.

regression model that could discriminate between true and false spliced alignments in *H. sapiens* [153]. They achieved high accuracy using features of the spliced alignments such as the number of reads uniquely aligned to a sequence; the distribution of mismatches and gaps in the alignments, and how many aligned bases fell on either side of a junction. Yassour et al. [246] used a position-specific scoring matrix (PSSM) to learn motifs in the DNA sequence around known splice sites in *S. cerevisiae*, then accepted novel splice sites whose PSSM scores fell into the 95th percentile for their data. These studies illustrate two kinds of features that can help characterize correct spliced alignments: those based on alignment statistics and those based on sequence characteristics.

Spliced alignment algorithms have since been developed that use sophisticated splice site models to validate their spliced read alignments [39, 216, 41, 83, 230, 23, 210]. Most of these tools perform spliced alignments by splitting the reads into smaller segments and aligning the segments to genomic sequences (see Figure 2.3). When a segment maps to a unique location these methods next try to align the read's remaining segments to a location nearby (usually constrained by a bound on intron length). Tophat [216] is arguably the most popular of these tools. It accepts spliced alignments when they infer introns that are bounded by canonical GT-AG splice-site dimers and have lengths within specified limits. Another tool, PalMapper [192, 39, 83], is an example of a sequence-based method that incorporates support vector machine (SVM) classifier scores into its alignment scores. These SVM classifiers use sequence-based kernels to recognize signals in the short sequences flanking GT or AG dimers. MapSplice [230], computes alignment statistics for the set of reads aligned across a junction. These include an *anchor significance* statistic that reflects how far reads extend on either side of a junction, and Shannon's entropy to measure how uniformly reads are aligned across a junction. In our experiments we found that MapSplice can provide high sensitivity, resolving 6-8% more alignments than Tophat, but in practice its results tend to be inconsistent, often failing to align paired-end reads, and providing inadequate alignment information for subsequent processing with tools such as Cufflinks.

We want SpliceGrapherXT to work with a variety of alignment tools including the most popular, Tophat [217]. At the same time we want to trust the spliced alignments SpliceGrapherXT uses for its predictions. Our solution was to add to SpliceGrapher sequence-based SVM classifiers similar to [192, 39] that it can use to filter spliced alignments from other tools.

4.2 RNA-Seq Alignment

Accurate RNA-Seq alignment—particularly spliced alignment—is critical for making accurate splice graph predictions. In earlier work we performed spliced alignment by creating a database of putative splice junction sequences and aligning reads to the database with an ungapped alignment tool [179]. To build our database, we first used splice-site classifiers to predict all possible donor and acceptor sites within a genome (recall that a *donor site* is the splice site that defines an exon's 5' end, while an *acceptor site* marks its 3' end). We then constructed putative junction sequences by pairing every known or predicted donor site in a gene with all known and predicted acceptor sites downstream of it in the same gene. Alignment then consisted of two phases: first we aligned all reads to a reference genome. Those reads that did not align in the first phase we aligned to the splice junction database in the second phase (similar approaches have been used in other studies [144, 153, 204]).

This approach worked well for a relatively simple and compact genome such as *A. thaliana*, but the splice site database can become too large to be practical for complex genomes with many possible combinations of donor and acceptor sites. A more efficient approach is instead to perform alignments with a spliced alignment tool and use our splice-site classifiers to filter out alignments with spurious splice sites. This eliminates the need for a splice site database and means that classifiers will be applied only to the relatively small fraction of novel splice-sites accepted by the alignment tool. We demonstrated this method successfully on Tophat alignments for *H. sapiens* [179].

We selected Tophat for performing RNA-Seq alignments in our experiments for several reasons: First, Cufflinks, one of our benchmarks (Section 3.2.1), relies on meta-data that Tophat provides in its alignment files, so Tophat was required to achieve the best possible Cufflinks performance. Tophat was also used in the development of IsoLasso [113], the other method we use as a benchmark. Second, by using the same alignment tool for all methods, our results reflect only the differences in the exon prediction methods. Finally, Tophat is arguably the most popular spliced alignment tool (see, e.g., [22, 129, 33, 243, 126, 120, 18]), so our results could have implications for a broad variety of other studies.

4.3 Splice site classifiers

Researchers have used various machine learning approaches to model splice sites, including linear discriminants [199], quadratic discriminants [249], neural networks [21, 76, 156], Hidden Markov Models

(HMMs) [170], and SVMs [205, 40, 163, 164, 200, 14]. In general, the problem of splice site recognition is relatively easy for discriminative classifiers; ROC scores for predicting GT or AG sites often exceed 95% (for examples, see [200, 163, 179] and Figure 4.2). The most successful methods use SVM classifiers with string kernels applied to DNA sequences [85]. These kernels use features based on combinations of k-*mers*: strings of length k taken from the sequences on either side of a splice site (for a complete exposition of string kernels, see [14, 191]). These kernels can map nucleotide sequences into a high-dimensional space that allows classifiers to learn the intricate nucleotide patterns characteristic of splice sites.

4.3.1 SpliceGrapher methods

For a particular organism we create a classifier for each kind of splice site, as each site's splicing signals may be distinct. For example, the GT and GC dimers both correspond to donor sites, but they are recognized by different splicing protein complexes [25]. To create splice site classifiers, SpliceGrapher extracts positive and negative example sequences for splice site donor and acceptor dimers such as GT, GC and AG, following the procedure described in [163]. Briefly, we use known splice sites as positive examples for a given dimer and for negative examples we use all other occurrences of the dimer found within genes. Training examples then consist of intronic and exonic sequences taken from either side of a site.

SpliceGrapher's classifiers discriminate sequences using an implementation of the weighted-degree kernel [163]. This kernel represents a sequence in a feature space of k-mers associated with positions in the sequence. Kernel parameters include exon and intron sequence lengths on either side of a splice site, k-mer length, number of mismatches to allow within a k-mer, and whether to allow shifts in k-mer position (for a detailed overview, see [14]). SpliceGrapher iterates over combinations of these parameters to identify the best-performing combination. It uses the PyML package [15] to train and test these SVMs using the training examples described above. For each classifier we generate training data, select the best parameters for the model and store the final model in a portable format used by SpliceGrapher scripts. Here we describe the pipeline that automates this process using GT dimers as an example, but the process is the same for any dimer.

To build a classifier for GT dimers we generate a balanced set of training data using N positive and N negative examples, where N is a pipeline parameter (the default is 1000). We create a training example using the intronic and exonic sequences flanking GT dimers. For positive examples we find all splice sites in the gene models that are associated with a GT dimer and randomly select N sites. To generate a negative

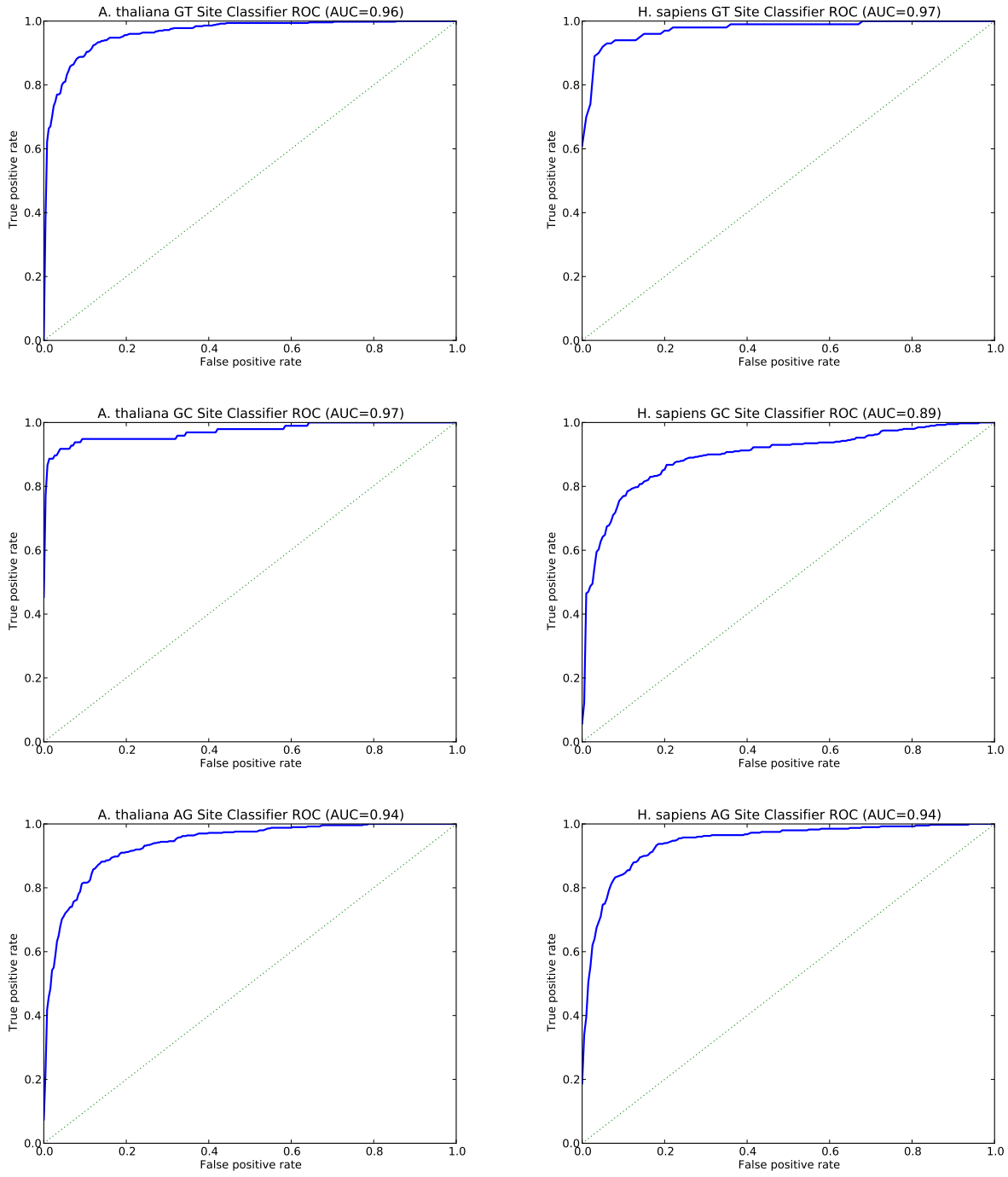

Figure 4.2: ROC curves that demonstrate the accuracy of the SVM classifiers SpliceGrapher generated for GT and GC donor sites and AG acceptor sites in *A. thaliana* (left), *H. sapiens* (right).

example, we first select a gene at random from the gene models. Within that gene we select a GT dimer at random that is not associated with a known splice site and create an example from the sequences flanking that dimer. We repeat this procedure until we have N negative examples.

Given a set of sequences as our training data, the next step is to select the parameters that will yield the best classifier performance. The user provides a range of possible values for each of the model parameters outlined above. The algorithm iterates over all possible combinations, using 5-fold cross-validation to evaluate the classifier on each combination. The classifier with the best ROC score is selected and its associated parameters and data are stored in a file.

4.4 Performance evaluation

Several studies have demonstrated the accuracy of SVM classifiers for predicting splice sites [163, 164, 39, 83]. As we reported in [179], the classifiers we generated for *A. thaliana*, *V. vinifera* and *H. sapiens* all achieve high accuracy. To assess the effectiveness of our pipeline for building classifiers for different species, we downloaded gene models for 107 different species from the ENSEMBL website [51]. We used the SpliceGrapher pipeline to generate classifiers for GT donor sites and AG acceptor sites for each of the species. On average, our GT donor site classifiers achieved ROC scores over 0.96, while AG acceptor site classifiers achieved scores over 0.94 (see Figure 4.2 and Appendix D for details). This demonstrates that SpliceGrapher's splice-site classifiers can be used to filter alignments in nearly any species that has enough gene model annotations to generate a training set.

Our objective in developing accurate splice site models is to ensure that the spliced alignments we provide to SpliceGrapher are as accurate as possible. As explained in Chapter 3, simulated RNA-Seq data allows us to establish a ground truth for evaluating our results, but lacks much of the noise we find in real data. Real data allows us to evaluate different methods in the presence of noise, but inevitably will include novel splice sites that we cannot verify without running expensive assays. Thus we used both real and simulated data to evaluate our classifiers' effectiveness in predicting valid splice junctions.

4.4.1 Filtering alignments with simulated RNA-Seq data

For our simulation experiments we used FluxSimulator to generate 1 million paired-end reads in *A. thaliana* and *H. sapiens* as described in Chapter 3. We used Tophat to align the simulated reads to a reference genome, creating a set of unfiltered alignments, then used SpliceGrapherXT's classifiers to obtain filtered

alignments from the unfiltered alignments. We used SpliceGrapherXT to predict splice graphs from the unfiltered alignments and from the filtered alignments, and collected statistics for the number of correctly predicted exons, recall and precision for both sets of predictions. We repeated the experiments ten times to obtain average measurements.

Table 4.1: Exon prediction statistics using filtered or unfiltered splice junctions with alignments from simulated data. Shown are the number of correctly predicted exons, along with recall and precision statistics for the predictions. When SpliceGrapherXT makes predictions with filtered alignments, the number of correctly predicted exons can drop by more than 20% compared with predictions using unfiltered data, while precision increases by just 1.4%.

	A. thaliana			*H. sapiens*		
Method	TP	Recall	Precision	TP	Recall	Precision
Test set	2,627	—	—	9,504	—	—
Filtered	1,189	0.453	**0.951**	1,042	0.110	**0.927**
Unfiltered	1,320	**0.503**	0.938	1,319	**0.139**	0.920

In our simulations, we observe a clear tradeoff between precision and recall when we apply the filters: predictions from unfiltered data yield higher recall than predictions from filtered data, but the filtered data yields higher precision (Table 4.1). However, in these simulations, small increases in precision incur relatively large decreases in recall. In *A. thaliana* precision increases by just 1% while recall decreases by nearly 10%.; in *H. sapiens* precision increases by less than 1% while recall decreases by over 20%. Thus we investigated our filtering process more closely.

Table 4.2: Evaluation of SpliceGrapherXT's classifiers on simulated data. The classifiers give us very high precision in predicting real splice sites, but we find that a large proportion of the splice junctions our classifiers reject are false-negatives.

Species	TP	FP	TN	FN	Precision	Accuracy
A. thaliana	1,134	43	41	169	0.96	0.85
H. sapiens	2,906	29	87	643	0.99	0.82

Because our simulated reads are generated from the complete gene models, assessing the filtering process is relatively straightforward: We count as true-positive predictions those junctions accepted by the filters that are present in the complete gene models; true negatives are junctions rejected by the filters that are not in the gene models; false positives are predictions accepted by the filters but not in the gene models, and false negatives are those rejected by the filters but present in the gene models. From this evaluation we found that up to 88% of the splice junctions SpliceGrapherXT's classifiers rejected are false-negatives in the simulated data (Table 4.2). However, we also found that the classifiers give us extremely high precision:

fully 96% to 99% of of positive predictions are real junctions. One possible reason for these results is the way we generate our training data. We have good evidence for the positive examples in our training sets, as they use sequences flanking donor or acceptor sites from curated gene models. However, for negative examples we use the sequences flanking random donor or acceptor dimers found in the genes. As a result, our negative examples may include some as-yet undiscovered splice sites, so our classifiers may learn to predict some real splice sites as spurious.

4.4.2 Filtering alignments with real RNA-Seq data

Table 4.3: Exon prediction statistics for filtered and unfiltered splice junctions. Shown are the number of correctly predicted exons from the gene models, along with recall and precision statistics for the predictions relative to known exons. When SpliceGrapherXT makes predictions with filtered alignments, the number of correctly predicted exons drops by no more than 11% compared with predictions using unfiltered data, while precision increases by up to 23%.

	A. thaliana			*H. sapiens*		
Method	TP	Recall	Precision	TP	Recall	Precision
Test set	14,617	—	—	97,504	—	—
Filtered	2,413	0.165	**0.155**	3,956	0.041	**0.593**
Unfiltered	2,465	**0.169**	0.126	4,506	**0.046**	0.516

As discussed earlier, simulated RNA-Seq data lacks much of the noise present in real data, so we investigated the impact of SpliceGrapherXT's filters on its predictions using the real data for *A. thaliana* and *H. sapiens* described in Chapter 3. As in our simulation experiments, we ran SpliceGrapherXT on both filtered and unfiltered data and used the same methods described in Chapter 3 to measure recall and precision relative to splice forms in the left-out set. In this case the results are far more encouraging (Table 4.3). In *A. thaliana*, SpliceGrapher's predictions with filtered data incur a mere 2.1% drop in recall but provide a dramatic 23% increase in precision. In *H. sapiens*, predictions using filtered data yield a 15% increase in precision with an 11% drop in recall. Thus on real data we find a much more appealing tradeoff between recall and precision from our filtering. In addition, we argue that results for real data provide a more realistic picture of the advantages of filtering than results from simulated data that may make the spliced alignment task much easier.

In earlier work we found that up to 68% of Cufflinks novel AS predictions were associated with false-positive splice junctions [179]. These false-positive junctions can have a strong impact on the accuracy of the novel exon and transcript predictions Cufflinks produces (Figure 4.3). By filtering spliced alignments, SpliceGrapherXT is able to avoid using possibly spurious splice junctions in its predictions. The success of

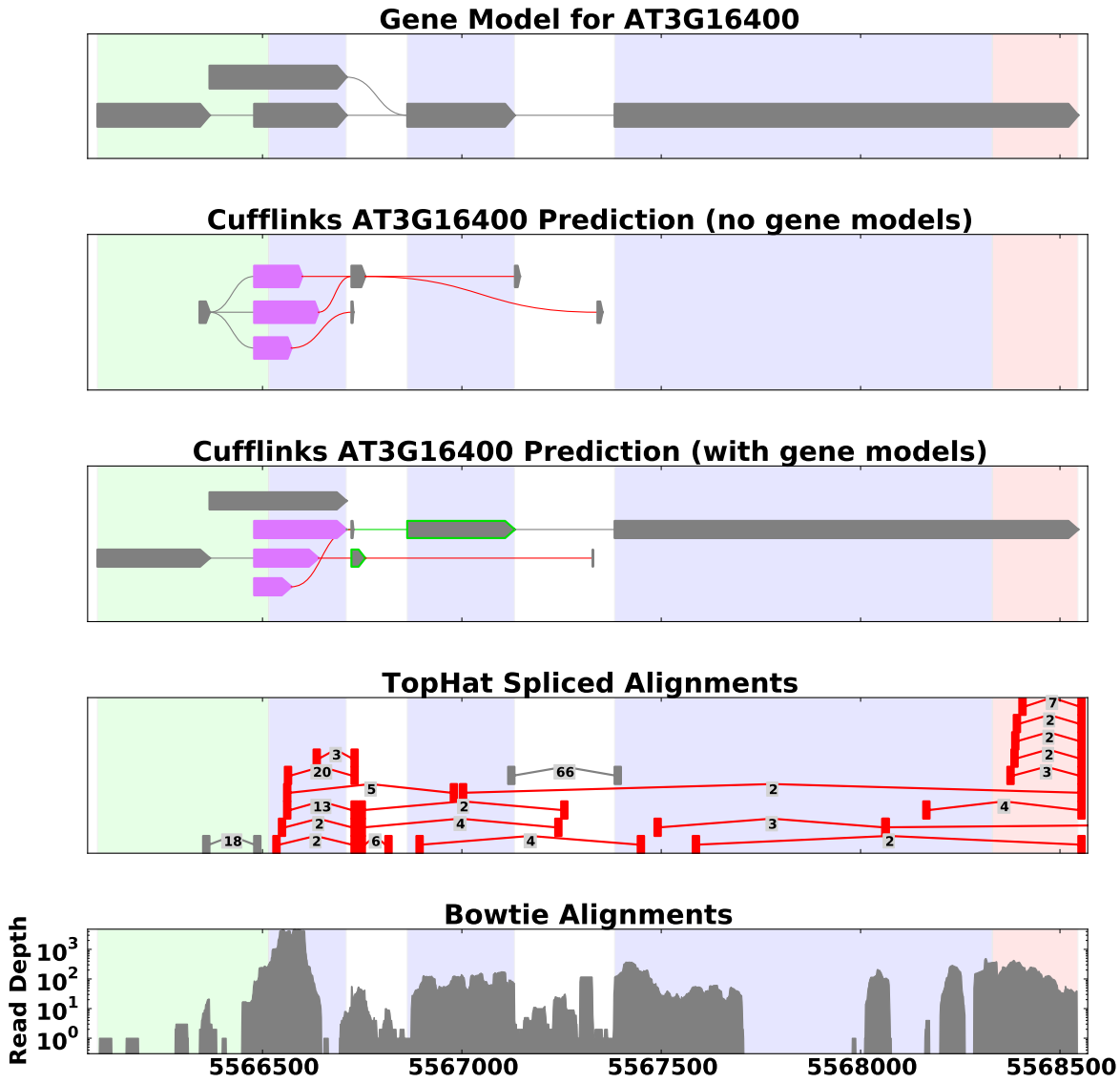

Figure 4.3: Cufflinks prediction for a gene in *A. thaliana*. Cufflinks attempts to find a set of transcripts that are consistent with the preponderance of false-positive splice junction evidence (highlighted in red). The resulting predictions likely contain spurious exons.

this approach is evident from our results filtering real data as discussed above, along with the exon prediction results discussed in Chapter 3.

4.5 Conclusion

SpliceGrapherXT's novel splicing predictions rely heavily on evidence for novel junctions from spliced alignments, but some spliced alignment tools may detect a large amount of false-positive novel junctions. A number of splice-site models have been developed that can predict legitimate splice sites with high accuracy. Spliced alignment tools such as PalMapper [83] use sophisticated splice site models to eliminate false-positives from their alignments, but these tools are slow and rarely used. Instead we have incorporated a similar model into a method for filtering spliced alignments. SpliceGrapherXT can use its splice-site classifiers in two different ways. By applying the classifiers to every donor or acceptor site in a genome, it can build a database of predicted splice junction sequences for every gene. This database can then be used as a set of reference sequences to perform spliced alignments using ungapped alignment tools. This was the method used in our original work [179]. Building a database of predicted junctions allows users to bypass the difficulties associated with spliced alignment programs, but these databases may become prohibitively large for complex genomes such as *H. sapiens*. Alternately, SpliceGrapherXT can use its classifiers to filter output from spliced alignment tools as described in this chapter. This allows us to use SpliceGrapherXT with a variety of spliced alignment tools while assuring the accuracy of our predictions. We have evaluated SpliceGrapher's classifiers using both of these approaches and found that they yield similar results.

Our method is able to recognize legitimate splice sites in the alignments while reliably rejecting false-positives. However, our method sacrifices some sensitivity to achieve its high precision. We accept this trade-off because we want SpliceGrapherXT to generate as few false-positive predictions as possible, and because there is as yet no known "gold standard" for dimers associated with known non-splicing sites. Despite this tradeoff, as we demonstrated in Chapter 3, SpliceGrapher's recall in predicting novel exons is still much higher than other methods. Thus our classifiers are a key reason that SpliceGrapher is able to combine a high level of recall with high precision as well.

Chapter 5

Resolving Ambiguities Using Realignment

When SpliceGrapherXT is unable to resolve a prediction from RNA-Seq evidence, it stores the exons that might be inferred from the evidence, and labels each one *unresolved*. Labeling these putative exons as unresolved serves two purposes: First, it allows SpliceGrapherXT to annotate graphs with evidence of novel AS even when it cannot resolve specific events. Second, the stored information makes it possible to resolve some of the putative exons if new evidence becomes available.

One way to extract additional information from a set of RNA-Seq reads is to realign them to a set of putative transcripts. Similar approaches have been used successfully in other domains such as genome assembly [44, 96, 29, 181] and isoform prediction [140]. This chapter describes a method we developed that realigns RNA-Seq reads to a set of putative transcripts we compile from unresolved exons. We find that in some cases, reads that align to these putative transcripts provide enough evidence that SpliceGrapherXT can resolve up to 30% of the exons that the inference method alone could not.

5.1 Realignment Pipeline

Initially, SpliceGrapherXT predicts novel exons conservatively and records information about those it could not resolve due to ambiguous combinations of acceptor and donor sites. We wish to resolve those ambiguous loci by realigning RNA-Seq reads to a database of putative transcripts in order to find compelling evidence for exons that SpliceGrapherXT could not resolved in its inference step. We thus created a pipeline that constructs putative transcripts from unresolved exons, aligns reads to these transcripts, and resolves exons that have sufficient coverage from the alignments (see Figure 5.1).

5.1.1 Creating putative transcripts

When SpliceGrapherXT encounters unresolvable evidence during its initial splice graph predictions, it constructs putative exons from each combination of ambiguous acceptor and donor sites and inserts them into the predicted graph, labeling each of them as *unresolved*. In addition, it records all possible edges between these unresolved exons and neighboring exons in the graph using splice junctions from the gene models and from the RNA-Seq alignments. For each predicted graph with unresolved exons, SpliceGrapherXT

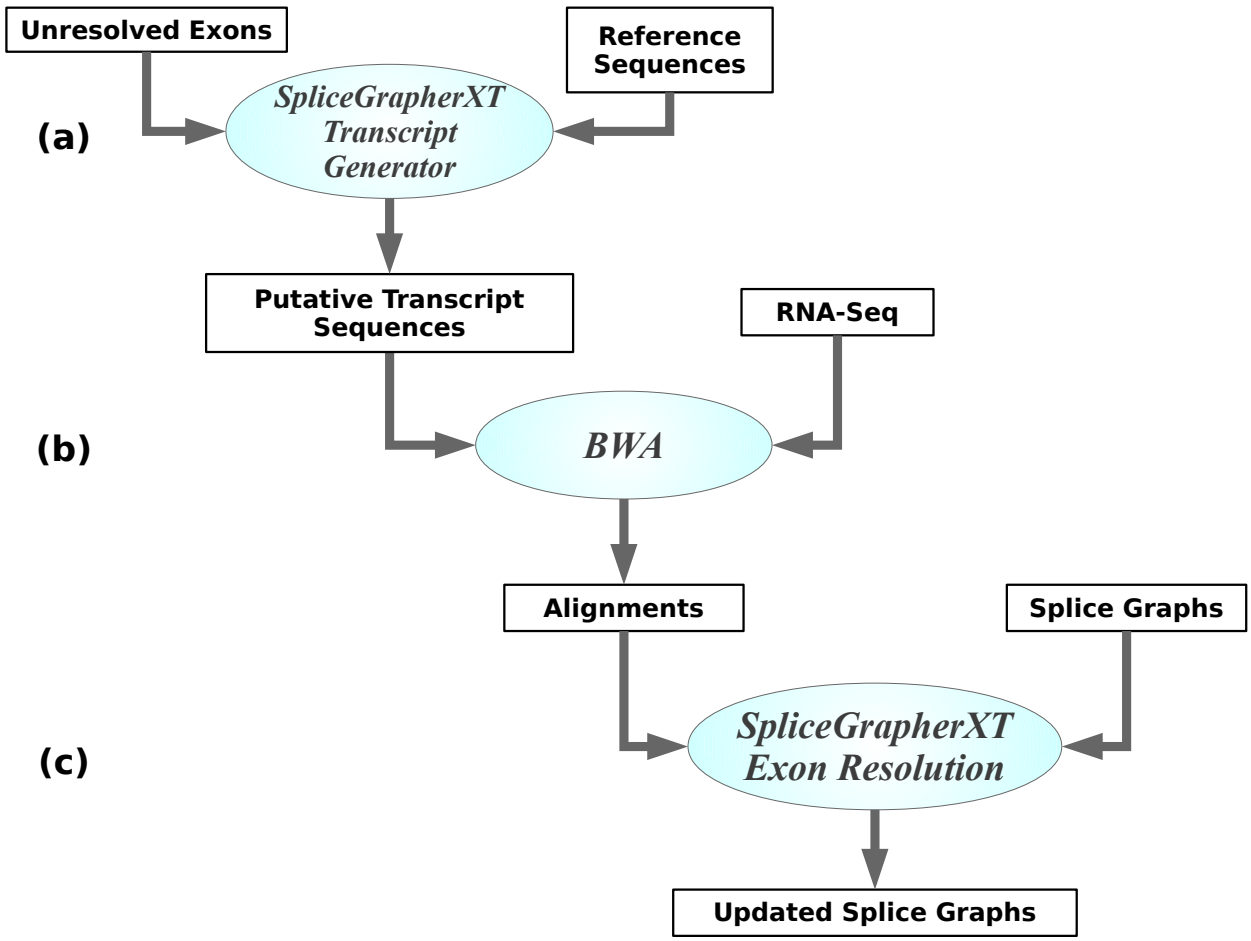

Figure 5.1: Flow of data through the SpliceGrapherXT realignment pipeline: (a) unresolved exons from predicted splice graphs are used to generate a putative transcriptome; (b) RNA-Seq reads are aligned to the transcriptome using BWA, and (c) aligned reads are used as evidence to resolve exons and that are incorporated into the splice graphs.

generates transcripts as follows: First it inserts all unresolved exons into the graphs, connecting edges to their putative neighboring exons. Next, it constructs putative transcripts by traversing all paths through the graph and concatenating the DNA sequences for the exons along each path. Traversing all possible paths in every graph for an organism can potentially yield an intractably large putative transcriptome, but in our experiments with human and Arabidopsis the full realignment procedure takes slightly longer than the initial predictions; a single run on real data for *H. sapiens* or *A. thaliana* took at most 75 minutes.

5.1.2 Realigning reads

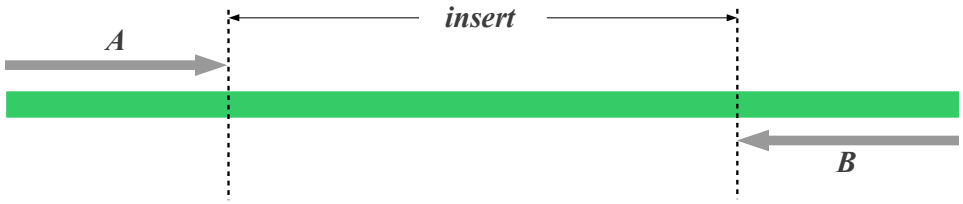

Figure 5.2: Depiction of a paired-end read alignment. The two reads in the pair, A and B, should align to the same sequence (green) and on opposite strands. The distance between the two reads infers an *insert* distance that must fall within an acceptable range for BWA alignments. BWA determines this range by estimating the insert size distribution from the RNA-Seq data.

Spliced alignment tools such as Tophat and MapSplice use the Bowtie ungapped alignment tool as a backend. However, we obtained more alignments, more resolved exons and more accurate realignment predictions using BWA [107] in the realignment step. Our pipeline thus uses BWA to realign reads to the putative transcript sequences and uses the following procedures to remove potentially spurious alignments. For single-end reads we simply remove any reads that align to two or more transcripts, but paired-end reads are handled differently. In paired-end sequencing, reads are obtained starting from either end of an RNA fragment, so that the sequences are oriented in opposite directions (Figure 5.2). This process usually leaves a gap between the reads; the distance between them is called the *insert* size. Studies have shown that these sizes tend to be normally distributed in RNA-Seq samples (see, e.g., [107, 90]). Accordingly, to filter paired-end data, we require that both reads in a pair align uniquely to the same transcript but on opposite strands. In addition, the BWA alignment tool estimates the insert size distribution from the RNA-Seq data and accepts only pairs with insert sizes in that range [107].

5.1.3 Resolving exons

We use the reads aligned to a putative transcript to resolve exons. For each transcript associated with a splice graph, SpliceGrapherXT looks for read coverage across unresolved exons. Whenever reads cover an entire exon plus anchor regions on either side (the default being 10 bases), SpliceGrapherXT adds the exon to the graph along with the junctions associated with that transcript. Recall that unresolved exons arise from regions with multiple choices for acceptor and donor sites, so the anchor regions allow us to discriminate between them. In addition, there may be many possible graph edges that could connect a putative exon to other exons in a graph. By adding only the junctions inferred by a transcript, we ensure that only edges supported by read coverage are added to the graph.

5.2 Related Work

iReckon uses a realignment procedure similar to ours to improve the accuracy of its isoform expression estimates [140]. The main difference in our approach is that we generate putative transcripts only for those genes that contain unresolved exons. To our knowledge, these are the only two AS discovery methods that use this procedure. Realignment methods have been used successfully in the area of DNA sequence assembly to verify or refine assemblies [44, 96, 29, 181].

Sequence assembly methods that work with next generation sequencing (NGS) data often rely on realignment procedures to correct errors in initial assemblies. For example, Meraculous [29] performs *de novo* assembly from NGS paired-end reads. It builds an initial set of contigs from the reads, then realigns the reads to these contigs to resolve gaps between them. SEQuel [181] is a post-processing tool designed to correct errors in genome assemblies. A genome assembler constructs an initial set of contigs based on NGS reads, then SEQuel realigns reads to these contigs and uses the realignment information to augment the original contigs. Finally, "assisted assembly" [61] is an algorithm designed to perform assembly when coverage is weak. It generates a *de novo* assembly from the reads, aligns the reads to a homologous reference genome, and uses the alignment information to enlarge the initial contigs.

In each of these scenarios, realigning reads to a set of putative sequences provides evidence for refining assemblies or isoform predictions. For SpliceGrapherXT, the realignment procedure allows it to refine its splice graph predictions. By generating putative transcripts it can perform realignment using an ungapped alignment tool, bypassing some of the challenges associated with spliced alignment.

5.3 Evaluating the pipeline

Now we can determine how many unresolved exons our realignment pipeline can resolve, and how these predictions might change SpliceGrapher's ability to predict novel exons and AS events. Using the same protocols for real and simulated data outlined in Chapter 3, we evaluate the performance of SpliceGrapherXT's realignment pipeline and compare it to SpliceGrapher's initial predictions and to Cufflinks.

Table 5.1: Exon prediction results averaged over 10 simulation runs for *H. sapiens* (left) and *A. thaliana* (right) with 1,000 randomly-selected AS genes. Approximately 1 million paired-end reads were generated for each run (500,000 pairs). Gene models were reduced to a single isoform before being supplied to each method. *Reference* shows the number of exons present in the full gene models for the randomly selected genes; the *test set* shows the number of exons removed from gene models. Results show the number of correctly predicted exons (TP), and the recall and precision for each method.

	H. sapiens			*A. thaliana*		
Method	TP	Recall	Precision	TP	Recall	Precision
Reference	19,956	—	—	10,323	—	—
Test set	9,349	—	—	2,655	—	—
SpliceGrapherXT	1,053	0.11	**0.93**	1,198	0.45	**0.95**
Realigned	1,086	**0.12**	**0.93**	1,299	**0.49**	0.93
Cufflinks$_S$	590	0.06	0.32	620	0.24	0.32
Cufflinks$_B$	574	0.06	0.31	571	0.22	0.33
IsoLasso	550	0.06	0.13	520	0.20	0.22
IsoLasso/CEM	546	0.06	0.12	607	0.23	0.24

In our simulation experiments, the realignment procedure provides increased recall, especially in *A. thaliana*, at the cost of a slight decrease in precision (Table 5.1): recall increased by 3% in *H. sapiens* with a 0.2% reduction in precision, while in *A. thaliana* recall increased by 8% while precision dropped by 2%. Cufflinks predicts far fewer exons in both species, with low precision owing to a large number of false-positives. In addition, we confirmed that when SpliceGrapherXT cannot predict a definite set of exons, the realignment procedure can resolve them in many cases (Figure 5.3).

To assess the ability of each method to predict novel AS events accurately, we compared the AS events in the predicted graphs with those in the complete gene models (see Table 5.2). Note that since these predictions are based on reduced gene models, any predicted AS is considered novel. SpliceGrapherXT's initial predictions capture 69% of the AS events in *A. thaliana*, and 21% in *H. sapiens*, while Cufflinks$_S$ predicts 64% and 21%, respectively, though with much lower precision. The realignment procedure improves SpliceGrapherXT's sensitivity, predicting 76% of the events in *A. thaliana* with no appreciable drop in precision. The largest improvement is with intron retention events in *A. thaliana*, where the realignment procedure

Table 5.2: Recall and precision statistics for SpliceGrapherXT initial predictions (SpliceGrapherXT), realignment procedure predictions (Realigned) and Cufflinks across the four types of AS events. Overall, the realignment procedure correctly predicts up to 7% more AS events than the initial predictions without sacrificing precision. The procedure correctly predicts 76% of the AS events in *A. thaliana* and 21% in *H. sapiens*, with the largest increase in intron retention events for *A. thaliana*. Cufflinks performance does not match either the initial predictions nor the realigned predictions.

AS Statistics for 1,000 Genes (*A. thaliana*)										
	IR		ES		Alt. 5'		Alt. 3'		Overall	
Method	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.
SpliceGrapherXT	0.65	**0.95**	0.70	**0.98**	0.67	**0.98**	0.73	**0.99**	0.69	**0.97**
Realigned	**0.75**	0.90	0.75	**0.98**	**0.73**	0.97	**0.80**	0.97	**0.76**	0.95
Cufflinks$_S$	0.51	0.24	**0.88**	0.91	0.62	0.19	0.69	0.27	0.64	0.27
Cufflinks$_B$	0.34	0.41	0.62	**0.98**	0.40	0.61	0.45	0.80	0.43	0.64
IsoLasso	0.03	0.18	0.45	0.49	0.17	0.30	0.22	0.44	0.18	0.38
IsoLasso/CEM	0.04	0.18	0.62	0.63	0.24	0.35	0.37	0.53	0.28	0.46

AS Statistics for 1,000 Genes (*H. sapiens*)										
	IR		ES		Alt. 5'		Alt. 3'		Overall	
Method	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.	Recall	Prec.
SpliceGrapherXT	**0.08**	**0.86**	0.32	0.93	0.13	**0.93**	0.13	**0.93**	**0.21**	**0.93**
Realigned	**0.08**	**0.84**	**0.33**	0.93	**0.14**	0.92	0.14	0.92	**0.21**	0.92
Cufflinks$_S$	0.07	0.37	**0.33**	0.86	0.11	0.17	**0.16**	0.21	**0.21**	0.42
Cufflinks$_B$	0.05	0.57	0.22	**0.99**	0.07	0.50	0.10	0.61	0.14	0.79
IsoLasso	0.00	0.12	0.18	0.38	0.03	0.20	0.05	0.30	0.09	0.35
IsoLasso/CEM	0.00	0.11	0.20	0.41	0.03	0.22	0.06	0.32	0.11	0.37

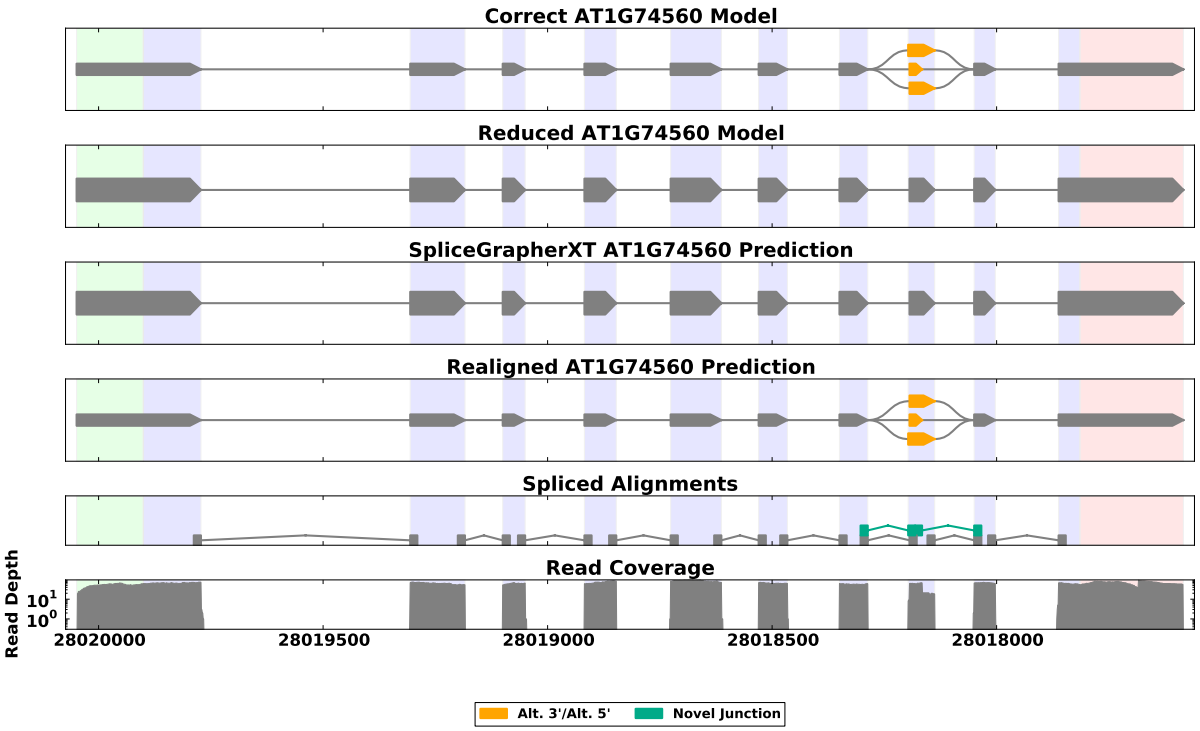

Figure 5.3: An example of the success of the realignment procedure using simulated data for *A. thaliana*. In this case, novel splice junctions (highlighted in green) can give rise to four possible novel exons that cannot be distinguished. SpliceGrapherXT stores information about this evidence for the realignment procedure, which finds compelling evidence for three of the possible exons and makes the correct prediction.

increases the number of correct predictions by over 15%. Realignment does not have any noticeable effect on AS prediction in *H. sapiens* for our simulated data.

Table 5.3: Prediction performance for the realignment procedure on real data for *H. sapiens* (left) and *A. thaliana* (right). Shown are the number of exons removed from the original gene models (left out) followed by the statistics for each method. SpliceGrapherXT made the initial predictions using filtered MapSplice alignments, then the realignment procedure used BWA alignments to putative transcripts to augment the initial predictions. The Cufflinks and IsoLasso predictions were made using unfiltered Tophat alignments.

	H. sapiens			*A. thaliana*		
Method	TP	Recall	Precision	TP	Recall	Precision
Left out	87,799	—	—	14,038	—	—
SpliceGrapherXT	3,965	0.045	**0.59**	2,413	0.17	**0.16**
Realigned	4,367	**0.050**	0.58	2,833	**0.20**	**0.16**
Cufflinks$_S$	2,006	0.023	0.16	1,636	0.11	0.02
Cufflinks$_B$	2,015	0.021	0.10	996	0.07	0.04
IsoLasso	1,308	0.013	0.04	890	0.06	0.03
IsoLasso/CEM	1,359	0.014	0.04	990	0.07	0.03

Our results for real data show that the realignment procedure again increases exon recall while maintaining the same level of precision (Table 5.3). It increases the number of correctly predicted exons in *H. sapiens* by more than 10% and in *A. thaliana* by more than 22%, again, with no appreciable loss in precision. The realignment procedure correctly predicts more than twice as many exons as Cufflinks$_S$ in *A. thaliana* and 73% more in *H. sapiens*, while it predicts nearly three times as many exons as the IsoLasso methods.

Table 5.4: Comparison of paired-end and single-end read performance averaged over 10 runs for *H. sapiens* (left) and *A. thaliana* (right) with 1,000 randomly-selected AS genes. Results show the recall and precision for predictions made using either paired-end reads or single-end reads. Only with Cufflinks is there a noticeable difference in recall and precision between the two kinds of reads.

	H. sapiens		*A. thaliana*	
Method	Recall	Precision	Recall	Precision
SpliceGrapherXT				
single-end	0.113	0.931	0.457	0.970
paired-end	0.113	0.930	0.452	0.948
Realigned				
single-end	0.117	0.927	0.493	0.937
paired-end	0.116	0.928	0.489	0.925
Cufflinks				
single-end	0.066	0.267	0.233	0.323
paired-end	0.063	0.318	0.219	0.334

5.3.1 Paired-end vs. single-end reads

We expected the realignment procedure to leverage information from paired-end reads to resolve exons that the inference method could not. To test the hypothesis that paired-end reads provide better evidence for exon resolution than single-end reads, we performed simulations using the same protocol as before. The only difference is that instead of simulating 1 million read pairs, we simulated 2 million single-end reads for 1,000 randomly-selected multi-isoform genes. The results (Table 5.4) show almost no difference between the results for single-end and paired-end reads. Cufflinks precision increases with paired-end data, as its overlap graphs make explicit use of paired-end data when it is available.

The reason SpliceGrapherXT's realignment procedure does not appear to benefit from paired-end reads may be traced to the constraints on paired-end reads: both reads in a pair come from the same mRNA fragment; both reads in a pair must align to the same reference sequence; they must align on opposite strands, and the alignment positions must infer an insert size that is compatible with the distribution for the data set. Altogether, these constraints reduce read coverage for a paired-end experiment but do not appear to hurt prediction performance.

5.4 Discussion

Here we have presented a realignment procedure that is robust, and can increase SpliceGrapherXT's sensitivity to novel exons and AS events while maintaining high precision. The procedure is also efficient, since it is only applied to a small set of genes and the procedure relies on ungapped alignments that tend to be more efficient than spliced alignment. We found the realignment procedure required just an hour on average for the five sets of 76-nt *A. thaliana* reads, about the same as the original predictions.

In the last three chapters we have shown that RNA-Seq data makes accurate splice graph prediction challenging. The relatively short read lengths can make it difficult to disambiguate evidence for many different AS events that may appear in the same genomic region. SpliceGrapherXT uses three different methods to address these ambiguities: the inference method described in Chapter 3 provides an efficient approach for making accurate predictions when the evidence is definitive; the filtering method described in Chapter 4 limits the number of spurious junctions that find their way into SpliceGrapherXT's predictions, and the realignment procedure described here allows it to resolve exons in some cases even in the presence of conflicting evidence. In Chapter 6 we will now show how SpliceGrapherXT's splice graph predictions may be used to predict complete mRNA transcripts.

Chapter 6

Using Splice Graphs to Predict Transcripts

We now turn our attention to converting SpliceGrapherXT's splice graph predictions into predictions for complete mRNA transcripts. To date, most methods that predict mRNA transcripts have focused on predicting the transcripts and their expression levels simultaneously [113, 68, 148, 54, 111, 140]. However, comparisons on both simulated and real data show that these methods suffer from poor accuracy [179, 140]. With SpliceGrapherXT we have shown that by focusing on accurate exon prediction rather than whole transcript prediction, we can predict splice graphs with high sensitivity and precision. A relevant question, then, is whether we can use our predicted splice graphs to improve the accuracy of transcript predictions in other methods.

Accordingly, we present two procedures for predicting transcripts from SpliceGrapherXT's predicted splice graphs: one uses PSGInfer [103] to estimate the relative frequency for each path in a splice graph, and a second uses IsoLasso [113] to predict expression levels for a set of transcripts. Both tools are designed to identify transcripts recapitulated in RNA-Seq data and quantify their expression, but each is limited in its capacity to predict novel transcripts. PSGInfer does not try to predict novel exons, but may predict novel isoforms from novel combinations of known exons. IsoLasso is able to predict novel exons and isoforms, but as our experiments in Chapters 3 and 5 demonstrate, its sensitivity falls below that of SpliceGrapherXT or Cufflinks. Here we show that we can improve transcript prediction accuracy for tools like PSGInfer and IsoLasso when we provide them with SpliceGrapherXT predictions.

The basis of our method is to use predicted splice graphs to generate a set of putative transcripts that we can provide to these other methods to predict transcript frequencies or expression levels. We then predict novel transcripts whenever a putative transcript has a substantial frequency or estimated expression level. We use an approach that is similar to our realignment procedure, but instead of generating putative transcripts only for graphs with unresolved nodes, we generate transcripts by traversing all the paths in every graph that has multiple paths. In this case, we store the putative transcripts as a set of gene models that we can provide to these other tools to predict either probabilistic splice graphs or transcript expression levels.

6.1 PSGInfer

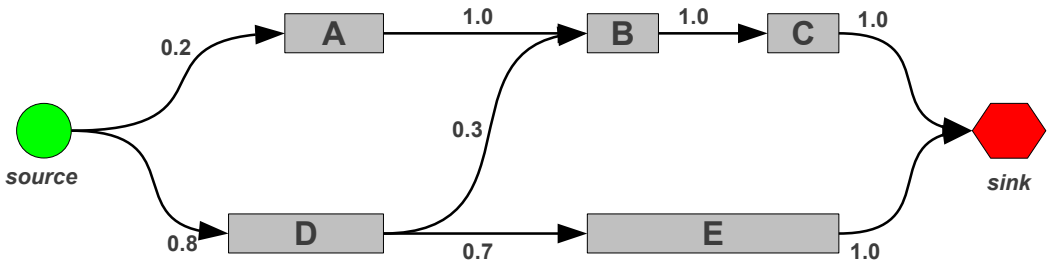

Figure 6.1: A probabilistic splice graph (PSG) with edge weights. A PSG is an augmented splice graph with two additional nodes (source and sink) and weights along each edge. The relative frequency of a transcript is given by the product of its edge weights. In this example, the relative frequency of the transcript ABC is $0.2 \times 1 \times 1 \times 1 = 0.2$, for transcript DE it is $0.8 \times 0.7 \times 1 = 0.56$, and for transcript DBC it is $0.8 \times 0.3 \times 1 \times 1 = 0.24$. Note that the transcript relative frequencies sum to 1.

In our first approach we provide our putative gene models to PSGInfer [103], a method designed to predict the relative frequency for each path through a splice graph based on the read coverage associated with the edges along the path. We then predict the transcripts given by the paths whose relative frequency exceeds a predetermined threshold.

PSGInfer performs its optimization over a *probabilistic splice graph* (PSG), an augmented splice graph that includes source and sink nodes as well as weights along every edge (Figure 6.1). PSGInfer relies on gene models for its exon and intron coordinates and thus is not designed to predict novel exons. However, putative gene models based on SpliceGrapherXT predictions can provide it with novel exons and thus allow it to predict frequencies for novel transcripts. Note that PSGInfer predictions also allow us to resolve exons in some cases. If PSGInfer predicts a high enough frequency for a transcript, we predict the transcript along with every exon in it; if one of the exons is unresolved, we can now resolve it.

PSGInfer begins by converting a set of gene models into a splice graph in the same manner as SpliceGrapher. A source node is added to each graph with edges leading to every root node (one without incoming edges) in the graph (Figure 6.1); similarly, a sink node is added with edges leading from every leaf node (one without outgoing edges) in the graph. Initially edges are assigned uniform weights: if there are n edges exiting a node in the graph, the initial weight for each outgoing edge is set to $w = \frac{1}{n}$. PSGInfer then uses the expectation maximization (EM) algorithm to converge to a set of edge weights that maximizes the likelihood of the observed read coverage. Once weights have been predicted for the edges in a graph, every

path through the graph is assigned a frequency that is the product of the edge weights along the path (see the examples in Figure 6.1).

To predict transcripts we select a suitable threshold and predict those paths whose frequency exceeds the threshold. For the example in Figure 6.1, if we set a frequency threshold of $f = 0.3$, then we would predict the transcript given by the path DE but not the other paths.

6.2 IsoLasso

Our second approach is similar to the first: we provide our putative transcripts to a tool that can predict expression levels for a set of transcripts. We use IsoLasso [113] and IsoLasso/CEM [114] for this procedure due to their popularity as standard benchmarks [113, 132, 140] and the ease of integrating them with our method. Recall from Chapter 3 that the IsoLasso methods are designed to predict transcripts and their expression levels simultaneously from RNA-Seq data. The main difference between them is the optimization method being used: IsoLasso uses the LASSO [213] method, while IsoLasso/CEM uses component elimination EM (CEM).

Compared with other methods, the IsoLasso methods exhibit poor sensitivity in predicting novel exons, regardless of whether we provide them with gene models (see Chapters 3 and 5). However, these methods optionally may be freed from predicting exons or transcripts and used solely to predict expression levels for a given set of transcripts. Thus their role here will be to use the putative gene models we provide them to estimate expression levels for isoforms that are recapitulated in RNA-Seq data. We can then predict transcripts by accepting those whose expression level exceeds some threshold.

6.3 Performance Evaluation

In our experiments we compare the transcripts predicted by each method with a set of left-out transcripts. To compare transcripts, we use the same procedure described in [113, 218, 132]: two transcripts match if and only if their sets of splice junctions match. This rule implicitly matches every exon in the two transcripts except for the beginning of the first exon and the end of the last exon. We use this approach to match transcripts because it is extremely difficult for these methods to predict exon boundaries without evidence for splice junctions, which will not occur at the start or end of a transcript.

Our methods use threshold values on two different kinds of values: estimated expression levels (FPKM) for the IsoLasso methods and relative frequencies for PSGInfer. For expression levels we need only choose

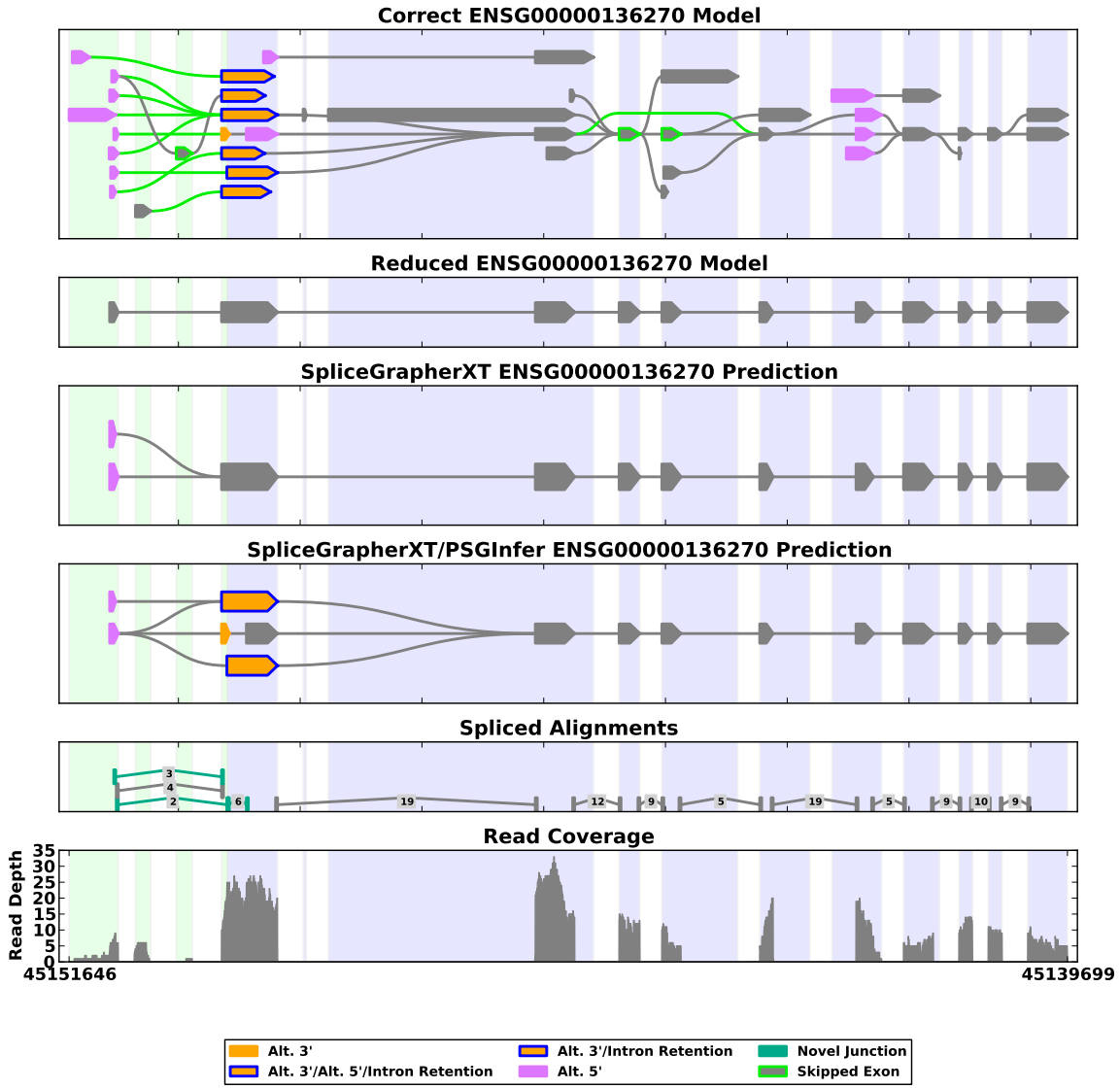

Figure 6.2: Example of an updated prediction for a gene in *H. sapiens* using PSGInfer with simulated data. SpliceGrapherXT is able to predict an alternative 5' event on the left side of the graph (third panel down), but there is also a a pattern of acceptor and donor sites ($aaadad$) that does not satisfy the SpliceGrapherXT criteria. By providing several unresolved splice forms to PSGInfer it is able to disambiguate these events correctly. With the relative frequency threshold set to the higher threshold of 0.15, PSGInfer added two transcripts with frequencies 0.23 and 0.24; two additional unresolved transcripts had frequencies of 0.04 and 0.03 and were omitted from this prediction. Human genes are difficult to predict due to complexity that is exacerbated by numerous alternative initial and terminal exons.

a reasonable value for identifying any isoform as being expressed. Values of FPKM $\geq$ 1 [27, 37, 34], FPKM $\geq$ 3 [233] and FPKM $\geq$ 5 [34] are used in the literature, so we chose FPKM $\geq$ 1 as it is the most sensitive of these. For PSGInfer predictions, we expect expect relative isoform frequencies to obey Zipf's Law (see, e.g., [59, 35, 186]), meaning that one or two isoforms may have high relative frequencies in a gene, while other isoforms can have frequencies that are orders of magnitude lower. We must balance this behavior against the high false-positive rate we are likely to get if we set our threshold too low. Thus for our experiments we selected two thresholds for PSGInfer predictions: $f \geq 0.01$ (PSGInfer$_{.01}$) to allow detection of weakly expressed isoforms, and $f \geq 0.15$ (PSGInfer$_{.15}$) to provide higher precision.

We found that these methods yield various levels of precision and recall that can make it difficult to determine which method is best overall. Thus in these experiments we include the F_1 score [225] in addition to recall and precision. Given precision P and recall R, the F_1 measure is defined as:

$$F_1 = \frac{2PR}{P+R} \tag{6.1}$$

This measure provides a single metric for comparing prediction methods that gives equal weighting to recall and precision [105].

6.3.1 Simulated data

Table 6.1: Transcript prediction performance averaged over 10 simulated runs in *A. thaliana* and *H. sapiens*. Shown are the number of true-positive and false-positive predicted isoforms along with the recall, precision and F_1 score. Results are shown for SpliceGrapherXT with PSGInfer, SpliceGrapherXT with IsoLasso and SpliceGrapherXT with IsoLasso/CEM. Also shown are results for the high-sensitivity version of Cufflinks (Cufflinks$_S$) and the balanced performance version (Cufflinks$_B$).

	A. thaliana				*H. sapiens*			
Method	TP	Recall	Precision	F_1	TP	Recall	Precision	F_1
SpliceGrapherXT updates								
PSGInfer$_{.01}$	**942**	**0.74**	0.65	0.69	**282**	**0.16**	0.10	0.12
PSGInfer$_{.15}$	796	0.63	**0.86**	**0.72**	180	0.10	0.13	0.11
IsoLasso	665	0.52	0.71	0.60	187	0.10	0.09	0.10
CEM	920	0.72	0.64	0.68	264	0.15	0.10	0.12
Unassisted methods								
IsoLasso	501	0.39	0.40	0.40	233	0.13	0.05	0.07
CEM	629	0.49	0.43	0.46	**281**	**0.16**	0.05	0.08
Cufflinks$_S$	546	0.43	0.26	0.33	250	0.14	0.09	0.11
Cufflinks$_B$	617	0.49	0.56	0.52	239	0.13	**0.15**	**0.14**

For our simulation experiments we used the same data described in Chapter 3: ten sets of simulated data each for human and *A. thaliana*. In our simulations, all of the SpliceGrapherXT update procedures

except IsoLasso provide much higher recall than Cufflinks and all yield higher precision in *A. thaliana* (Table 6.1). In *H. sapiens* the results are mixed, as both PSGInfer$_{.01}$ and the unassisted IsoLasso/CEM achieve the highest recall, while Cufflinks$_B$ achieves the highest precision and F_1 scores. The PSGInfer$_{.01}$ update procedure provides the highest recall of any method in both species, while PSGInfer$_{0.15}$ consistently yields the highest precision of the update procedures.

The differences between making predictions in *A. thaliana* and in *H. sapiens* are clear from these simulations: all of the SpliceGrapherXT update methods have recall that is much higher in the less well-annotated *A. thaliana* than in human, with a similar increase in precision. With fewer transcripts per gene in the *A. thaliana* gene models, fewer transcripts are represented in the simulated RNA-Seq data, making the prediction task much easier than it is for human.

We also used our simulations to confirm that these update procedures may be used to resolve exons. The example in Figure 6.2 illustrates how PSGInfer may be used for this purpose. SpliceGrapherXT provides several unresolved splice forms from a gene in *H. sapiens* for PSGInfer to use in its predictions. Using the frequency threshold of 0.15, PSGInfer is able to correctly predict two transcripts from a set of four unresolved transcripts. This example also illustrates the complexity of many genes in *H. sapiens* that can make it challenging to predict novel transcripts or novel exons. Not only are there a large number of AS events confined to relatively small regions, but there are also numerous alternative initial and terminal exons across the gene.

6.3.2 Real data

We tested our isoform prediction procedure on the real data from *A. thaliana* and *H. sapiens* described in Chapter 3, using PSGInfer and IsoLasso to predict the isoforms recapitulated in the data (Table 6.2). The results for real data are more subtle than for simulated data. In *A. thaliana*, the PSGInfer and CEM update methods have better recall than the Cufflinks methods, but with lower precision than Cufflinks$_B$. PSGInfer at the high threshold exceeds the precision of Cufflinks$_B$ substantially, and achieves the highest F_1 score of any method in that organism. On human data the differences between the update procedures and Cufflinks are more noticeable, where the update procedures yield precision up to five times as high and combined F_1 scores roughly double those of Cufflinks.

Prediction is much harder with real data than with simulated data. Real data can include a considerable amount of noise generated during the sequencing process, as well as evidence for novel transcripts. As a

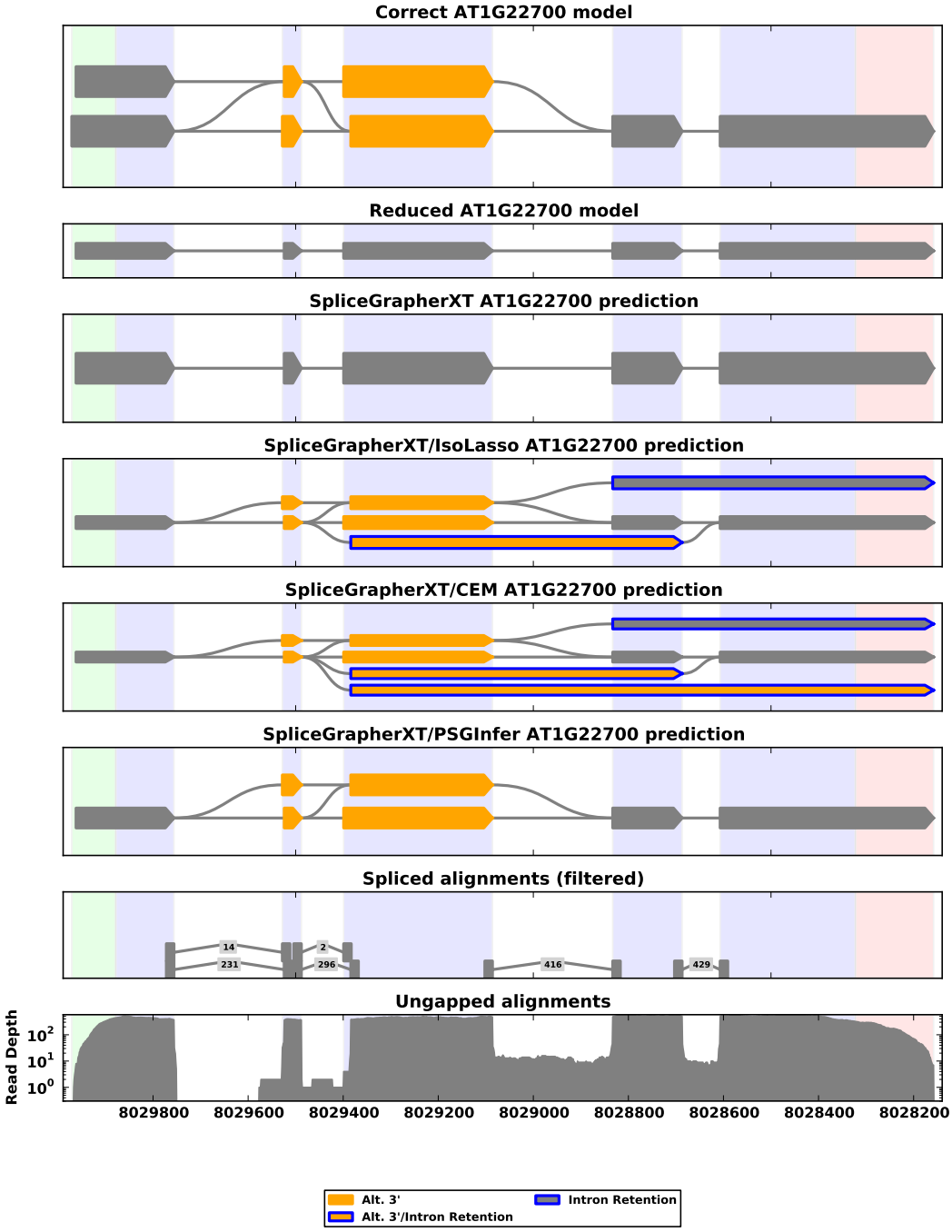

Figure 6.3: Example of predictions made from real RNA-Seq data in *A. thaliana*. In this gene from the reverse strand, a large read cluster runs for approximately 1,400 positions (from genomic position 8,029,600 down to 8,028,200), making the evidence within the cluster unresolvable for SpliceGrapherXT. Each of the three transcript prediction methods produces a slightly different prediction. The two IsoLasso methods predict novel IR events, some of which could be spurious. PSGInfer makes a more conservative prediction that matches the gene models but does not predict any novel IR.

Table 6.2: Transcript prediction performance on real data from *A. thaliana* and *H. sapiens*. Shown are the number of left-out isoforms correctly predicted, the proportion correctly predicted (recall), the proportion of true-positives (precision), and the F_1 score. Here the differences are more subtle than for simulated data; the PSGInfer update method achieves the highest F_1 score in *A. thaliana* while the CEM update method achieves the best score in *H. sapiens*.

	A. thaliana				*H. sapiens*			
Method	TP	Recall	Precision	F_1	TP	Recall	Precision	F_1
SpliceGrapherXT updates								
PSGInfer$_{.01}$	**1,388**	**0.269**	0.047	0.081	1,675	0.060	0.059	0.060
PSGInfer$_{.15}$	991	0.192	**0.162**	**0.176**	1,270	0.046	0.058	0.051
IsoLasso	999	0.194	0.064	0.096	1,355	0.049	0.091	0.064
CEM	1,283	0.249	0.067	0.106	1,569	0.057	**0.101**	**0.072**
Unassisted methods								
IsoLasso	627	0.122	0.025	0.042	709	0.026	0.006	0.010
CEM	749	0.145	0.028	0.047	832	0.030	0.004	0.007
Cufflinks$_S$	1,143	0.221	0.019	0.035	**2,825**	**0.102**	0.023	0.037
Cufflinks$_B$	1,004	0.195	0.097	0.129	1,594	0.057	0.022	0.032

result, recall and precision on real data are much lower for all methods than with the simulated data. This is especially evident in *A. thaliana*: the methods predict a large number of novel transcripts, so precision and recall are low relative to the gene models.

6.3.2.1 Resolving exons

Table 6.3: Comparison of the number of unresolved nodes that were resolved by the realignment procedure and each of the three update procedures on real data from *A. thaliana* and *H. sapiens*. The PSGInfer update procedure is able to resolve more nodes than any other method. Whereas the realignment procedure is able to resolve 25-29% of unresolved nodes, the PSGInfer update procedure can resolve up to 59%, or twice as many unresolved nodes.

	Unresolved	Update Procedure				
Organism	Nodes	Realignment	IsoLasso	IsoLasso/CEM	PSGInfer$_{.01}$	PSGInfer$_{.15}$
A. thaliana	21,999	5,474	7,782	8,901	12,072	2,139
H. sapiens	2,854	823	1,273	1,289	1,680	532

We also used the real data to investigate each method's potential to predict novel exons and to resolve exons that SpliceGrapherXT's initial predictions left unresolved (Table 6.3). We first note that there are approximately ten times as many unresolved exons in *A. thaliana* as in *H. sapiens*. This may be traced again to the relatively sparse *A. thaliana* annotations; it is evident that there are a large number of novel exons that remain to be resolved and added to the gene models (one of the key motivations for developing SpliceGrapherXT in the first place). We find that the update procedures are able to resolve even more

Table 6.4: Comparison of novel exon predictions from the update procedures along with the original predictions and the results of realignment on the real data from human and *A. thaliana*. The combination of SpliceGrapherXT with sophisticated transcript prediction algorithms is able to predict more novel exons than the simple realignment procedure at comparable precision levels.

	H. sapiens			*A. thaliana*		
Method	TP	Recall	Precision	TP	Recall	Precision
Test set	97,504	—	—	14,617	—	—
SpliceGrapherXT	3,956	0.041	0.59	2,413	0.17	0.16
Realigned	4,362	0.045	0.58	3,062	0.21	0.15
SpliceGrapherXT updates						
PSGInfer$_{0.01}$	4,653	**0.048**	0.56	3,483	**0.24**	0.13
PSGInfer$_{0.15}$	4,301	0.044	**0.60**	3,112	0.21	**0.18**
IsoLasso	4,506	0.046	0.57	3,248	0.22	0.14
IsoLasso/CEM	4,539	0.047	0.57	3,396	0.23	0.14

exons than the realignment procedure. Given the relative simplicity of the SpliceGrapherXT realignment procedure (Chapter 5), it is not too surprising that more sophisticated transcript prediction methods like PSGInfer or IsoLasso are able resolve more exons. In particular, PSGInfer is able to resolve nearly 30% of unresolved exons in *H. sapiens*. When we compare the accuracy of these predictions (Table 6.4) we find the precision across all the update methods to be comparable to the initial predictions and the realignment procedure, while the number of novel exon predictions may increase by more than 10%.

6.4 Conclusion

We have presented two methods for predicting novel transcripts by using SpliceGrapherXT to construct putative gene models from predicted splice graphs. We then provide these gene models to tools that can use them to predict transcript frequencies or expression levels. Both tools we selected, PSGInfer and IsoLasso, are extremely limited in the transcript predictions they can make without the benefit of good gene models. When we provide these tools with SpliceGrapherXT predictions, they are able to predict novel transcripts with much higher fidelity than Cufflinks. In addition, we found that these update procedures can resolve novel exons with higher sensitivity than our realignment procedure, and with similar precision.

Recently iReckon has been shown to provide higher precision and sensitivity than IsoLasso [140] for transcript prediction, so we plan to compare our results with iReckon. Incorporating iReckon into our pipeline may also be a promising direction for future work. Together, our results suggest that an approach that combines SpliceGrapherXT's accurate splice graph predictions with transcript expression estimation methods may give us the most complete picture of a gene's splicing behavior from RNA-Seq data.

Chapter 7

From Splice Graphs to Biological Insights

We have shown that SpliceGrapherXT can predict novel exons from RNA-Seq data more accurately than other methods and that it can improve transcript predictions, but ultimately we want to know how we can use the tool to derive insights from NGS experiments. In earlier studies with ESTs and RNA-Seq data we showed that splice graphs can aid statistical analysis of alternative splicing across a genome [99, 179], and that they can facilitate comparisons between gene families [172]. Here we discuss three studies that highlight the benefits and limitations of using SpliceGrapherXT to analyze RNA-Seq data.

Each of the projects has involved many different kinds of analysis including computational analysis and wet-lab assays. In the descriptions that follow we focus on the unique contributions that SpliceGrapherXT has provided to each project. We begin by showing how SpliceGrapherXT's genome-wide AS statistics may lead to insights about the subtleties of embryonic development in plants. Next we discuss a method for using splice graphs to address the differences in RNA-Seq data for biological or technical replicates, along with a method for comparing AS activity between two different variants of the same organism. Finally we show how the compact form of a splice graph may be used to create intuitive visualizations for complex chimeric mRNA. The first two projects are ongoing collaborations with Dr. Reddy's lab at Colorado State University. The third project was a collaboration with Dr. Valencia's lab at the Spanish National Cancer Research Center and was published earlier this year [58].

7.1 Assessing AS Across Embryonic Developmental Stages

A key concept in embryonic development is the notion of the *hourglass model*. In the early 19^{th} century, Karl von Baer proposed a set of four evolutionary "laws" pertaining to embryonic development. According to von Baer's third law, young embryos from different species resemble one another, but as development proceeds, species-specific features begin to appear and the embryos of different species progressively diverge from one another. By the late 20^{th} century this law had evolved into the *hourglass model* [48, 162]: distinct species appear markedly different from one another at the earliest developmental stages, they proceed through a *phylotypic* stage in which their characteristics are more similar, and then diverge progressively

again as they complete embryonic development.

Recent studies have provided evidence supporting the hourglass model in metazoan embryos [43, 87] and in plants [161]. To date these studies have focused on the relative expression levels of "young" and "old" genes, where young genes are recent evolutionary developments common only to closely-related species while old genes are common even among distantly related species. These studies provide evidence that expression levels for old genes peak during the phylotypic stage while the levels for young genes peak in earlier and later stages [43, 161].

Following the work of Quint et al. [161] Dr. Reddy obtained RNA-Seq data for the model plant *A. thaliana* across seven developmental stages and we assessed the amount of AS detected in each stage. For this project we aligned the RNA-Seq reads using MapSplice [230] and used SpliceGrapher's splice-site filters to remove potentially false-positive splice junctions. The number of aligned reads across the seven stages ranged from 12.9 million (*Mature* stage) to 26.4 million (*Torpedo* stage). To eliminate possible bias due to these differences in read coverage, we sampled 12 million aligned reads at each stage before performing our analysis. We then used SpliceGrapherXT to predict splice graphs for each developmental stage using only the prevalent splice form for each gene in the TAIR10 annotations. Using the prevalent forms not only provides us with evidence for AS specific to these RNA-Seq data sets, it also helps to eliminate possible biases owing to variation in the extent to which different genes are annotated.

7.1.1 AS Activity Peaks Near Phylotypic Stage

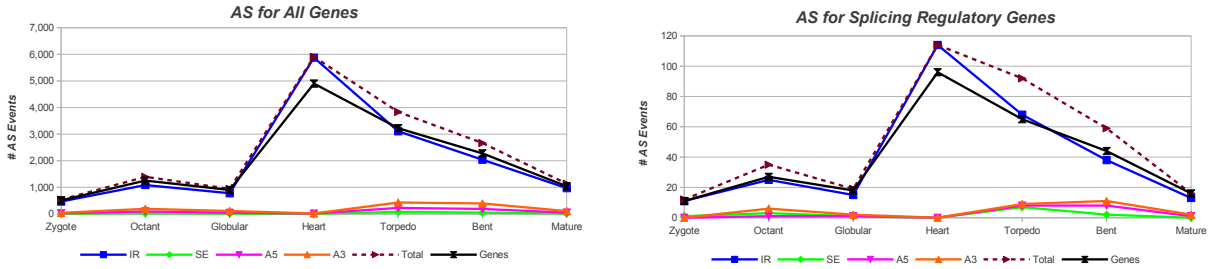

Figure 7.1: AS events predicted across seven embryonic development stages in *A. thaliana*. We find that AS activity appears to peak in the *Heart* stage, and remains relatively high into the *Torpedo* stage that is presumed to be the phylotypic stage (left). Splicing regulatory genes exhibit the same pattern of AS activity, suggesting a possible role before and during the phylotypic stage (right).

Comparing the amount of predicted AS across all stages, we find that AS activity appears to peak in the *Heart* stage (Figure 7.1, left), and is dominated by IR events (blue line). Previous studies have shown that

the *Torpedo* stage in *A. thaliana* appears to be the phylotypic stage [161]. Thus AS activity peaks just before the phylotypic stage and remains relatively high through the *Bent* stage before returning to pre-phylotypic levels. This same trend is evident in splicing regulatory genes (Figure 7.1, right), suggesting a possible role during the phylotypic stage.

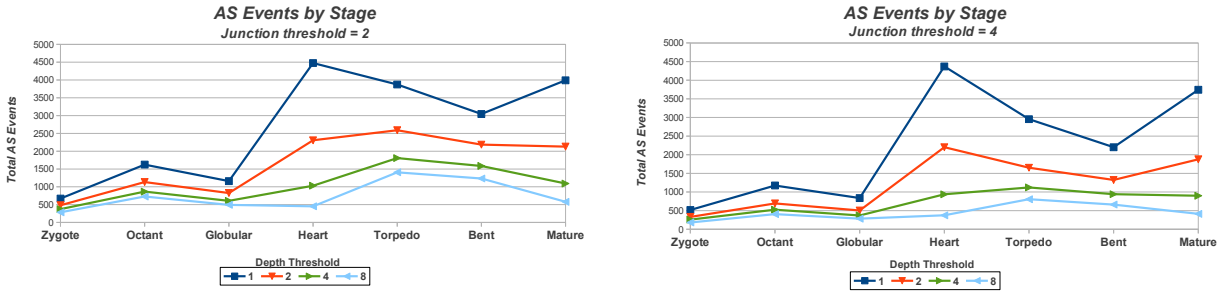

Figure 7.2: AS predictions across the embryonic stages in *A. thaliana* at low splice junction thresholds (left) and higher thresholds (right). The prediction depth threshold was increased from 1 to 8 to illustrate the effect of more stringent thresholds on the amount of AS predicted at each stage. At low thresholds we predict more AS for the *Heart* stage, but at higher thresholds we predict more AS in the *Torpedo* stage.

SpliceGrapherXT uses two thresholds to filter the RNA-Seq data it uses to make its predictions. The first is a threshold on the minimum average read depth required for a cluster of aligned reads to be used as evidence; The second is a threshold on the minimum number of spliced reads that support a novel splice junction before it is used as evidence. By default, SpliceGrapherXT accepts all read clusters and novel junctions with at least two supporting reads. These thresholds have the potential to change the amount of AS predicted for a particular data set. and hence could result in patterns of AS activity different from those we observe in Figure 7.1.

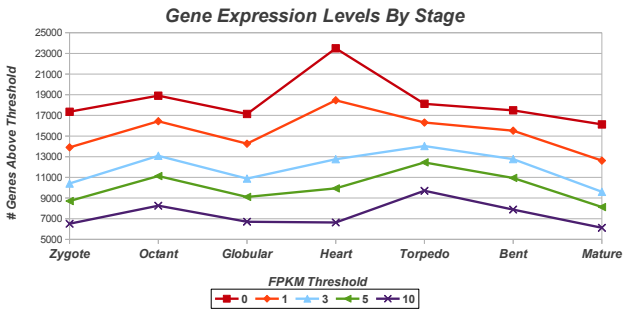

Figure 7.3: Number of genes expressed across embryonic development stages at different FPKM levels. We provided Cufflinks with the complete Arabidopsis gene models for it to provide estimated expression levels. We then counted those genes whose predicted expression levels exceeded a given FPKM threshold. There are more genes in the *Heart* stage expressed at low thresholds, but more genes in the *Torpedo* stage are expressed at higher thresholds.

To account for this possibility, we ran SpliceGrapherXT several more times using different thresholds each time and looked at the pattern for each run (Figure 7.2). The number of AS events predicted at each stage drops substantially as we increase the depth threshold, most noticeably in the *Heart* stage. At relatively high thresholds we predict more AS in the *Torpedo* stage than in the *Heart* stage. Thus although we have more evidence for AS in the *Heart* stage than in the *Torpedo* stage, the evidence appears to come from relatively weakly expressed transcripts.

We see similar trends when we consider the number of genes at different expression levels across the stages (Figure 7.3). To estimate the number of genes expressed at each stage, we first used Cufflinks to estimate expression levels for all known Arabidopsis genes by providing it with the complete set of gene models. We then counted those genes at each stage whose predicted expression levels exceeded a given FPKM threshold. By increasing the threshold we can assess the number of strongly expressed genes detected at each stage. At low FPKM thresholds we find many more expressed genes at the *Heart* stage, but at higher FPKM thresholds we find more genes in the *Torpedo* stage.

From these results it appears that many genes are both weakly expressed and alternatively spliced in the *Heart* stage, just prior to the phylotypic *Torpedo* stage. Whether this AS activity helps to initiate the phylotypic stage or whether it is a side-effect of other regulatory processes will require further analysis.

7.1.2 Old Genes Exhibit More AS Than Young Genes

Recently Quint et al. reported that the relative expression levels of young and old genes change across developmental stages, with old genes exhibiting the highest expression levels in the phylotypic *Torpedo* stage [161]. In their analysis they established a set of 13 *phylostrata* in which they assigned an evolutionary age to the genes in the *A. thaliana* genome. For each gene they used sequence similarity to identify homologs (genes with common ancestry) in other species. The resulting phylogenetic tree contained 13 nodes representing evolutionary distance from *A. thaliana*; its closest neighbors are other plants in the genus *Arabidopsis*, while the most distant neighbors are prokaryotes (mostly single-celled organisms such as bacteria). They defined a phylostratum as the most distant phylogenetic node containing at least one species with a detectable homolog for a given gene. The resulting phylostratigraphic map contains 13 phylostrata denoted *PS1* through *PS13*. PS1 includes the evolutionarily oldest genes with homologous sequences in prokaryotes, and PS13 includes the evolutionarily youngest genes with no homolog in any other species. To distinguish between young and old genes, the authors established a cutoff: genes in PS1-PS3 are considered

old, while those in phylostrata PS4-PS13 are considered young. This yielded phylostratum identifiers for 25,371 genes, of which 10,800 (42.6%) are considered old while the rest are young.

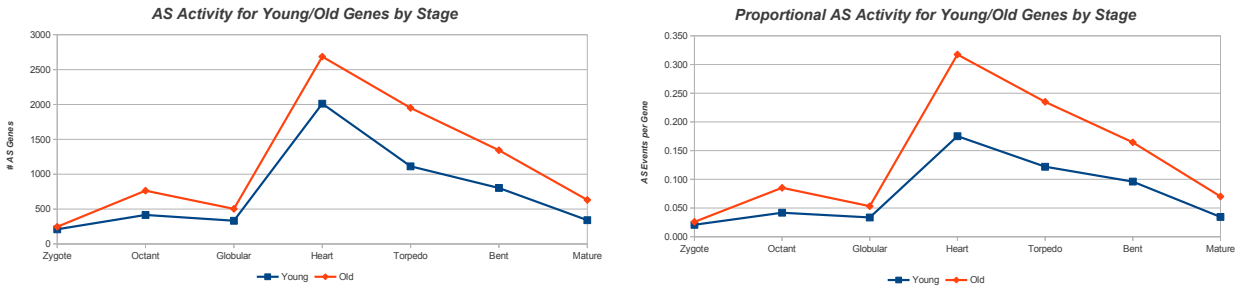

Figure 7.4: AS events predicted for young and old genes across the developmental stages in *A. thaliana*. Old genes represent just 42.6% of all genes, but have the most evident AS across all stages (left). This is especially evident when we look at the average amount of AS per gene in each group (right).

Using these same phylostrata, we compared the relative levels of AS activity for young and old genes across the developmental stages (Figure 7.4). Although old genes represent the smaller proportion of genes, they have more evidence of AS than young genes. This is especially evident when we look at the proportion of AS that occurs within each group (Figure 7.4, right). This evidence suggests that old genes undergo about twice as much AS activity per gene as young genes at the *Heart* stage. There is evidence that old genes simply have more splice forms than young genes [183], and that they are more sensitive to AS regulation [234]. Thus when an environment is conducive to AS, old genes may react more sensitively than young genes.

Overall our results suggest that there is indeed a dramatic increase in AS activity during embryonic development, and that this activity occurs disproportionally in old genes. Additional analysis will be required to determine how AS changes between stages and which AS events may be unique to each stage.

7.2 Assessing AS Between Variants

The serine/arginine-rich (SR) family of proteins plays a key role in mediating alternative splicing [165, 166, 65, 187, 172, 38] in a multitude of organisms (see, e.g., [247, 211, 125, 172, 165, 167]). However, the evidence to date suggests that plants possess the greatest variety of SR proteins of any organism studied [172, 13]. Hence investigating SR-protein genes in plants such as *A. thaliana* provides opportunities to investigate a wide array of SR gene behaviors, and could yield insights that apply across a broad spectrum of organisms. In our collaboration with Dr. Reddy's lab at CSU, we are investigating the influence of a particular SR gene, *SR45*, that has been associated with regulation of other SR genes [6], and in particular with recognition

of intronic 5' splice sites [38]. We are using RNA-Seq data based on mRNA samples from two different *A. thaliana* variants: one variant in which the SR45 gene has been *knocked out* (manipulated to render it inactive) and a wild-type (unmodified) variant.

When comparing two variants in the same species, it is important to understand whether the differences we see between data sets are due to differences between the variants themselves or if it is due to the variability inherent in RNA-Seq data. Because of this variability, and due to the relatively low cost of RNA-Seq data, technical or biological replicates have become an important component in these comparative studies [60, 138, 18, 46]. The data for our SR45 study consist of two replicates each for the wild-type variant (21.6 million and 22.5 million 76-nt reads) and the SR45 knock-out (18.3 million and 20.5 million 76-nt reads).

We first aligned the reads to the TAIR10 version of the genome, which yielded between 17.4 million and 21.3 million reads aligned per sample. To eliminate any possible bias due to read coverage, we sampled 17 million alignments from each data set. We used SpliceGrapherXT [179, 177] to predict splice graphs from the RNA-Seq alignments for each of the replicates. To assess the alternative splicing in our samples, we provided a reduced set of gene models to SpliceGrapherXT to make its predictions. Following the same protocol described in previous chapters, we created a reduced set of gene models from the TAIR10 version of the Arabidopsis gene models by removing all but the prevalent splice form for each gene. By using these reduced gene models as a reference, we ensured that any AS events in our predicted splice graphs were due to AS activity in the RNA-Seq samples.

Due to inherent variation in the RNA-Seq coverage, the evidence for AS may vary considerably from one sample to the next, even for samples from the same organism under the same conditions. Consequently, splice graph predictions may also vary widely across any set of replicates. Thus to produce a consistent set of predictions, we computed the intersections of the predicted splice graphs for every pair of replicates in each variant. The intersection of two splice graphs is the intersection of their vertex and edge sets (see Figure 7.5). The intersection of predicted graphs from two replicates will thus contain only those AS events that are common to both replicates.

To identify AS events that were unique to each variant, we ran SpliceGrapherXT using two different thresholds for RNA-Seq evidence. A high threshold ($\tau = 4$) meant that SpliceGrapher only accepted clusters of reads with an average depth at least 4 and splice junctions with at least 4 supporting reads. A low threshold ($\tau = 1$) meant that SpliceGrapher accepted all read clusters and all supported splice junctions. An

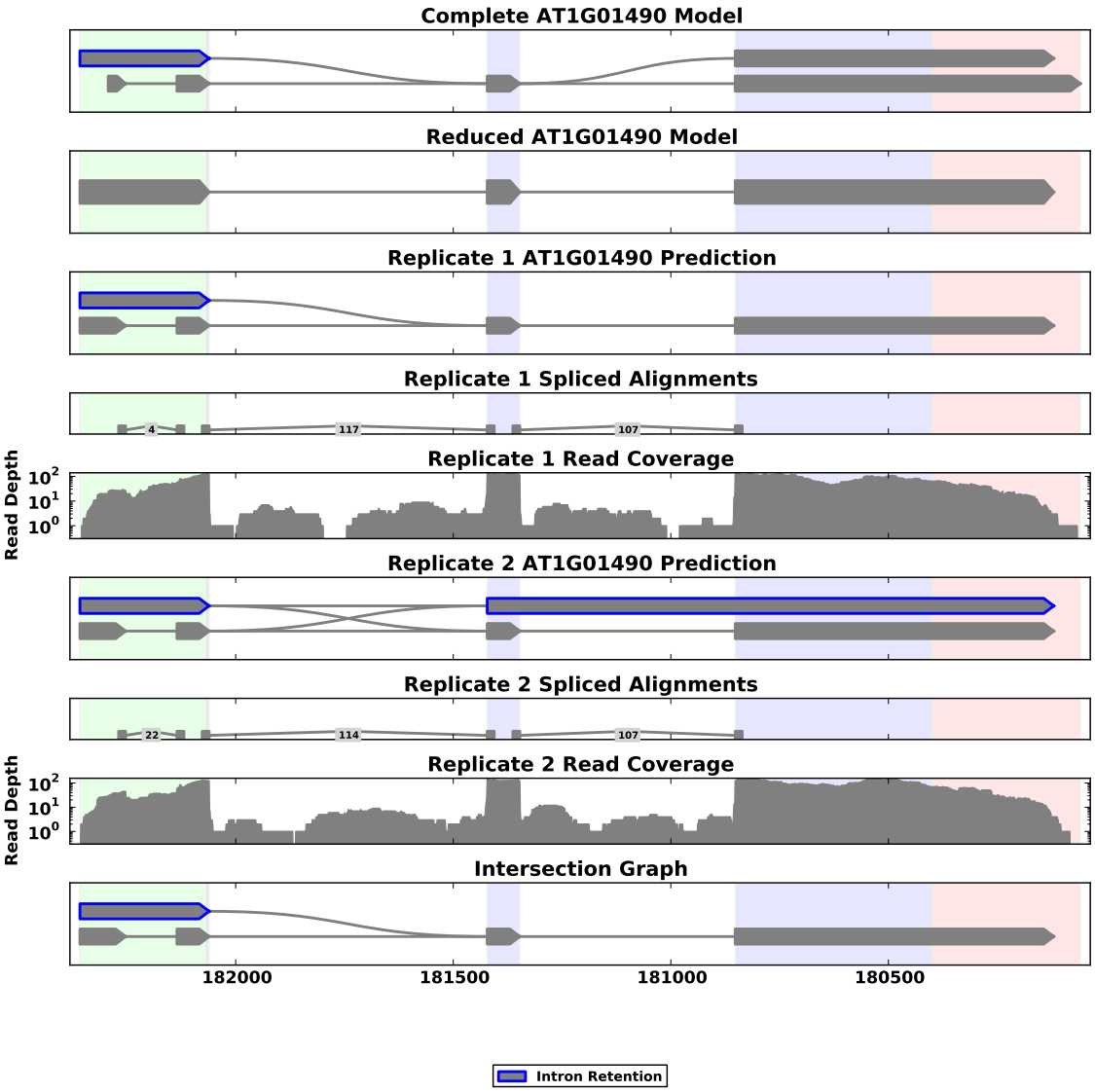

Figure 7.5: Example of a graph intersection based on two SR45 knockout gene replicates. The top panel shows the prevalent form of the gene provided to SpliceGrapherXT. The middle two panels show the resulting predicted graphs for the two SR45 knockout replicates. The bottom panel shows the intersection graph consisting of the exons (vertices) and introns (edges) common to both of the replicates.

Table 7.1: AS statistics based on RNA-Seq data for wild-type *A. thaliana* and SR45 mutants. Predictions were made by removing all but the prevalent isoform from the gene models for each gene, so the number of AS events is relative to the prevalent isoforms. Predictions were generated for each run separately (two lanes for SR45, two lanes for wild-type), then intersections of the predicted graphs were created for the SR45 and wild-type variants to identify AS events that were consistent within each variant. Shown are the total AS events found in the wild-type replicates and the SR45 mutant replicates, along with the number of exons that SpliceGrapherXT was unable to resolve. Also shown are AS events unique to each replicate along with AS events found in both (SR45 ∩ Wild-type).

		Alternative Splicing					
			Events				
Source	Unresolved Exons	Genes	IR	ES	Alt. 5'	Alt. 3'	Total
Total Wild-type	7,984	2,910	1,970 (59%)	170 (5%)	377 (11%)	834 (25%)	3,351
Total SR45	5,141	2,973	2,027 (58%)	230 (7%)	379 (11%)	856 (25%)	3,492
Unique Wild-type	4,689	940	885 (85%)	45 (4%)	41 (4%)	68 (7%)	1,039
Unique SR45	1,846	1,003	942 (80%)	105 (9%)	43 (4%)	90 (8%)	1,180
SR45 ∩ Wild-type	3,295	1,970	1,085 (47%)	125 (5%)	336 (15%)	766 (33%)	2,312

AS event was then considered unique to a variant if it was predicted at the high threshold for that variant but not predicted even at the low threshold for the other variant. These thresholds yielded predictions that we consider unambiguous for each of the variants (Table 7.1). Overall SpliceGrapherXT was able to predict slightly more AS events in the SR45 knockout than in the wild-type, with the biggest difference (35%) in exon-skipping events. However, these predictions omit a large number of unresolved exons: there are more than twice as many unresolved exons unique to the wild-type data, suggesting that there is more evidence for AS in wild-type. Hence we must be cautious about using these AS statistics to compare results from two different species or variants as they reflect only those AS events SpliceGrapherXT was able to predict unequivocally.

7.2.1 AS Within Novel Genes

To identify possible alternative splicing within novel genes, we first used Cufflinks to predict genes and transcripts. This gave us an initial set of 198 putative novel genes, of which 77 overlapped existing gene models, leaving 121 putative novel genes. Cufflinks did not assign a strand to any of the putative genes, so we inferred the strand using spliced alignments for those genes that contained them. This yielded just 9 candidates for which we could infer a strand. Despite the presence of spliced alignments, Cufflinks predicted each of these as single-exon genes. Using these Cufflinks predictions as gene models, we ran SpliceGrapherXT on both the wild-type and the mutant RNA-Seq alignments to predict splice graphs for these novel genes. Figure 7.6 shows an example of a novel gene that has well-supported evidence for

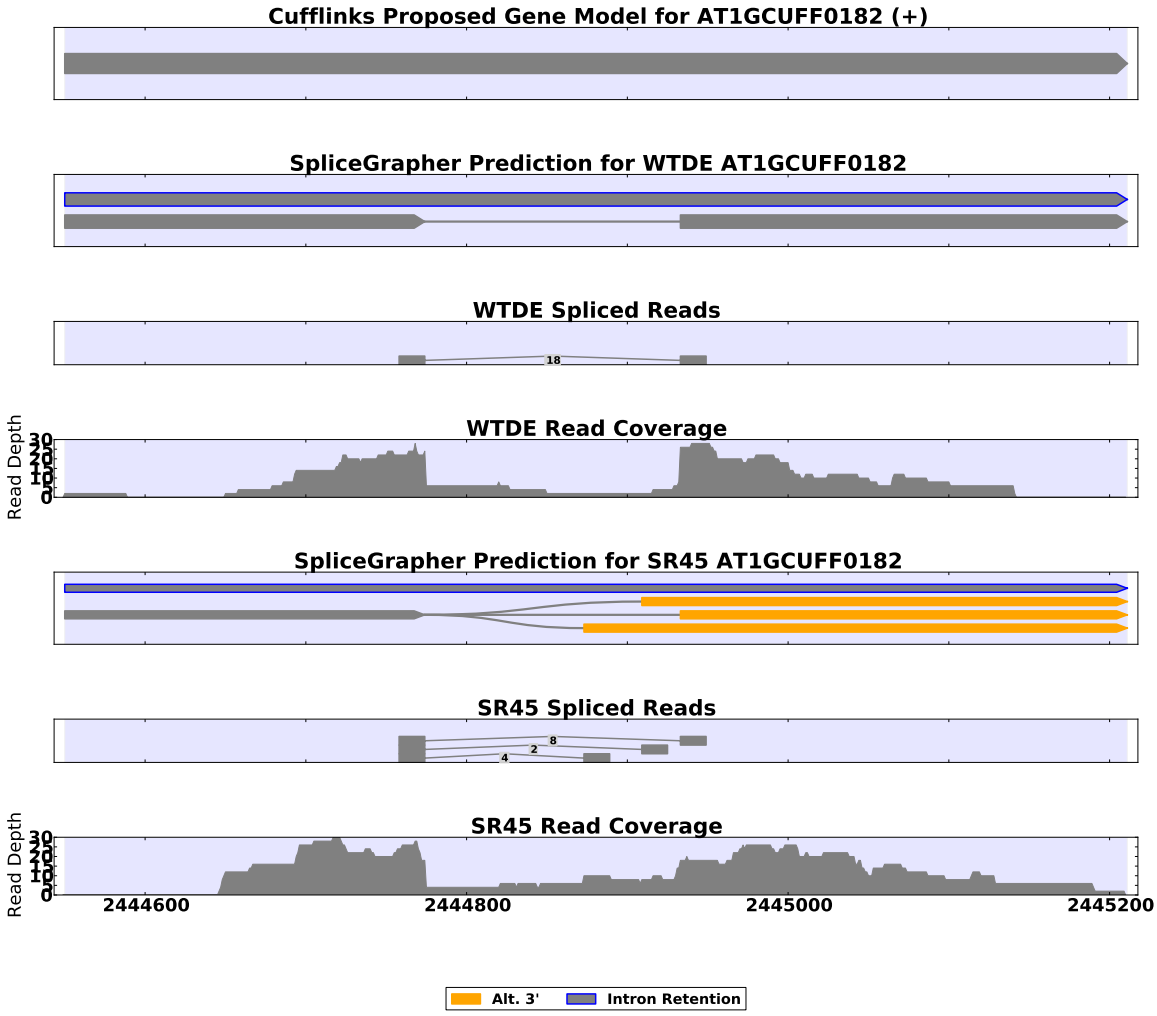

Figure 7.6: This figure depicts a novel single-exon gene predicted by Cufflinks. We resolved the gene strand using spliced alignments that fell within the gene boundaries and predicted splice graphs using the Cufflinks gene model as a reference. The alignment data for this novel gene provide evidence for AS events in the SR45 mutant that do not appear in the wild-type data.

different splice forms in the wild-type and SR45 mutant data. In this case we were able to use Cufflinks predictions to bootstrap SpliceGrapherXT so that it could predict AS for genes that do not have annotations in the TAIR10 gene models. In the future we would like to enable SpliceGrapherXT to predict novel genes and the splice forms within them without having to use Cufflinks to bootstrap the process.

7.3 Visualizing Chimeras

In mythology, a *chimera* is a monster with the head of a lion, the body of a goat and a serpent's tail. Like these mythical creatures, chimeric mRNA transcripts are cobbled together from mRNA molecules that come from different genomic locations [198, 244, 115] (Figure 7.7). These transcripts can be composed of distinct elements from within a gene [1]; from gene elements on opposite strands [100]; from distinct genes on the same chromosome [180], or from distinct genes on different chromosomes (see, e.g., [91, 45, 244, 175]). Initially these chimeric transcripts were perceived primarily to be products of gene fusions associated with cancerous cells [158, 98, 141] or mere alignment artifacts [94, 139]. However, recent evidence suggests that chimeric splicing also occurs in normal cells and may be subject to regulatory processes [3, 110].

Several models have been proposed to explain how chimeric transcripts form. Originally chimeras were associated with gene fusion events that occur when genetic DNA is translocated, or moved to a distant location in a genome from its original position. In this way, components from two genes may fuse into a single gene that, when expressed, produces chimeric transcripts [158, 98, 141]. More recent models of chimeric mRNA fall under the category of *trans-splicing*, defined as splicing that occurs between exon acceptor and donor sites from two separate pre-mRNA molecules [244]. For example, exons from tandem genes may be joined in the same transcript or spliced together to form a chimera [154]. Although trans-splicing has been found in numerous studies [91, 45, 100, 78, 244, 154, 245, 175], it remains unclear how the cell's splicing machinery may select exons to join from distinct pre-mRNA transcripts. Obtaining a better understanding of chimeras is a challenging task: not only are chimeric transcripts relatively rare, but it can be difficult to distinguish real chimeras from experimental artifacts [139]. Despite these challenges, there is evidence that chimeras may be more common than previously believed [139, 57, 58].

---

[1] Horiuchi et al. have shown that chimeric mRNA from the *longitudinals lacking* (*lola*) gene may arise from trans-splicing of exons at distinct locations within the gene [78]. They provide evidence that at least one alternatively selected exon has its own promoter region and can thus be transcribed independently. It may then be trans-spliced to other transcripts from the same gene.

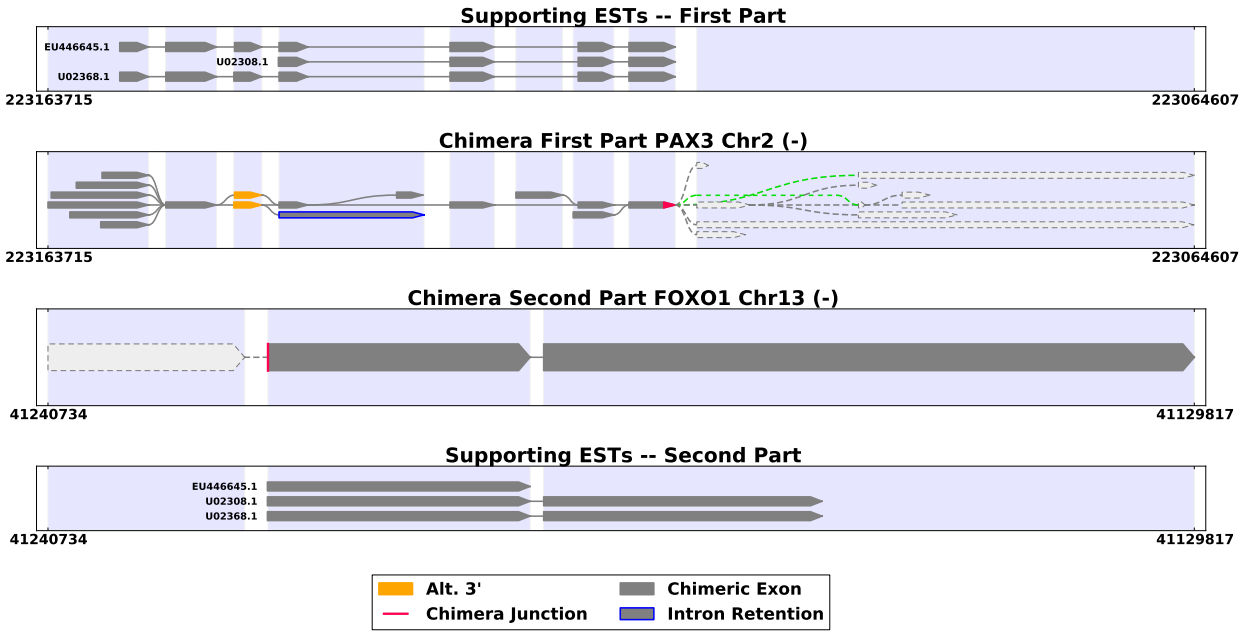

Figure 7.7: Splice graph visualization of an mRNA chimera from *H. sapiens* that involves genes on two different chromosomes. The top panel shows the first parts of three ESTs that aligned to the first gene. Each EST is labeled with its unique identifier. The second panel shows the gene model for the first gene with the chimeric junction highlighted in red. The third panel shows the gene model for the second gene with the second part of the chimeric junction highlighted in red (start of second exon). The bottom panel shows the ends of the ESTs that align to the second gene. Note that the 3' ends of the ESTs in the top panel align with the first half of the junction while the 5' ends of the ESTs in the bottom panel align with the second half of the junction.

To aid in interpreting these data, we used SpliceGrapher modules to develop methods for visualizing chimeric transcripts inferred from EST data [58]. In our visualizations we included several features to aid in analysis (Figure 7.7). Each chimera is characterized by a splice junction that begins in one gene and ends in another. We highlight these locations using bold red markers on the graphs. The gene features that may take part in a chimera will be upstream of the first part of the chimeric junction and downstream of the second part. Those gene regions that are not included in a chimera are obscured by rendering them with grey dashed lines.

ESTs provide supporting evidence for the chimeras in the study. We depict each EST as a separate transcript along with its unique identifier. Displaying the ESTs and the genes in this manner makes it easy to see the extent of the EST evidence for a chimera, what parts of each gene are likely to participate in a chimera, and the potential transcripts that could arise from each chimera. These SpliceGrapher visualizations are now part of an extensive database (*ChiTaRS*, now at UniProt [224]) of well-supported chimera examples from fruit fly, mouse and human [58].

7.3.1 Future Work

Our work has demonstrated the value of having good visualization tools for interpreting the evidence for chimeras based on ESTs. The ChiTaRS database also includes chimeras predicted from RNA-Seq data [58, 130, 131]. In addition, several tools have been developed that can predict chimeras recapitulated in RNA-Seq data [190, 95, 82, 116]. Thus an obvious next step will be to provide visualizations that include this RNA-Seq evidence.

Most methods for predicting chimeras from RNA-Seq rely on paired-end reads to detect chimeric events by looking for read pairs that map to exons within distinct genes. They then filter candidates by removing pairs of paralogous genes from their candidate sets [190, 95, 82, 116]. However, the trans-splicing model associated with chimeras is slightly different than the model associated with normal splicing [137], raising the possibility that the two processes are mediated by different splicing signals. Thus it is possible that more accurate predictions may be obtained by instead training classifiers to recognize patterns that are unique to chimeric sequences. As a first step one could use an an approach similar to the approach described for classifying splice sites in Chapter 4. A key difference will be the way in which positive and negative training examples are obtained. The ChiTaRS database provides information about chimeras that have been confirmed using mass spectrometry [57, 58] and should provide good evidence for known chimeric splice

sites.

7.4 Conclusion

In this chapter we have surveyed three projects in which SpliceGrapher has been used to generate biological insights. It has played a key role in our analysis of AS across different stages of embryonic development, providing compelling evidence that vibrant AS activity precedes the key phylotypic stage, and raising questions about how this AS activity may influence gene products in later stages. Splice graphs also provide an efficient structure for comparing the splicing activity between two samples. This means that we can compensate for high variability between biological or technical replicates by accepting only those predictions that are common to all replicates. By using graph intersections to identify predictions common to replicates allows us to make confident predictions across replicates, but other comparisons are more challenging. Finally, splice graphs provide compact representations of genes that make it easy to visualize complex phenomena such as complex gene models and trans-spliced chimeras. By including annotations for supporting evidence such as ESTs or RNA-Seq data, these visualizations can aid in developing and evaluating methods for predicting different kinds of chimeras.

Chapter 8

Conclusion

We have presented SpliceGrapherXT, a Python package designed to enhance existing gene annotations by predicting splice graphs from RNA-Seq and EST data. SpliceGrapherXT predicts AS from RNA-Seq data by applying a simple set of constraints to patterns of acceptor and donor sites. We have shown that this approach gives SpliceGrapherXT much greater precision and greater sensitivity than other state-of-the-art methods when predicting novel exons. SpliceGrapherXT uses gene annotations as a basis on which to construct its splice graphs, allowing it to make meaningful predictions even for genes with low read coverage, and to resolve AS events that are otherwise hard to detect from RNA-Seq data. In addition, it uses highly accurate splice-site models to filter out potentially spurious evidence for novel splice junctions provided by spliced alignment tools. When SpliceGrapherXT is unable to resolve patterns of acceptor and donor sites unequivocally, it saves information about potential novel exons. It can then use a realignment procedure to resolve up to 50% of unresolved exons while maintaining its high precision. These methods are highly efficient: predictions for a whole genome along with realignments to resolve exons will take about two hours on a typical workstation.

Once SpliceGrapherXT has predicted a set of splice graphs for a genome, we can use the predictions to improve the performance of transcript prediction methods. SpliceGrapherXT constructs putative gene models from predicted splice graphs and provides these gene models to tools such as PSGInfer and IsoLasso that can use them to predict transcript frequencies or expression levels. Our experiments demonstrate that when we provide IsoLasso/CEM with SpliceGrapherXT predictions, its recall and precision both improve dramatically on real data. In addition, SpliceGrapherXT-assisted PSGInfer and IsoLasso predictions can yield higher recall and precision than Cufflinks on both simulated and real data.

We have also used SpliceGrapherXT to generate biological insights using different experimental protocols. Our early work demonstrated the advantages of using splice graphs to assess AS in species with limited annotations [99] and to compare AS in highly conserved genes [172]. Since then SplicegrapherXT has played a key role in RNA-Seq based projects for comparing AS between different samples, such as mutant and wild-type *A. thaliana* variants, or comparing the amount of AS across different stages of plant

embryonic development.

Tools that predict complete mRNA transcripts from RNA-Seq data rely on sophisticated algorithms for mapping reads to their corresponding transcripts [218, 173, 113, 111, 114]. However, the ambiguities that arise from biases and noise in RNA-Seq data can make this problem difficult or impossible to solve. By addressing the slightly easier problem of predicting novel exons, we have been able to develop a relatively simple approach that yields better recall and substantially better precision than these other methods.

8.1 Future Work

As encouraging as our results have been, there are several avenues we would like to pursue to improve SpliceGrapherXT's prediction accuracy and utility. These include the ability to predict novel genes and their AS activity, establishing a model to account for noise to improve SpliceGrapherXT's sensitivity to novel IR events, and comparisons with other state-of-the-art methods.

8.1.1 Gene Prediction

In Chapter 7 we described an *ad hoc* procedure for assessing AS in novel genes: we first used Cufflinks to predict genes and transcripts in *A. thaliana*, then used as gene models those Cufflinks predictions that did not correspond to any known gene models. By combining these predicted genes with RNA-Seq we were able to find several novel genes with evidence of AS. However, we would like to add to SpliceGrapherXT the ability to detect novel genes and any AS that may be present in those genes. In addition, alternative initiating and terminating exons remain a significant challenge for methods that predict novel exons or transcripts.

Tools like Cufflinks and Scripture can predict genes using heuristics such as the distance between coverage regions and inferring strands using the spliced alignments within a region [218, 68]. However, there are two key aspects to this problem that may yield better predictions: identification of gene boundaries and identification of transcript start sites (TSS) and polyadenylation (poly-A) sites. A number of tools exist that are designed to predict genes based on genomic sequences (see, e.g., [221, 24, 199, 75, 202]). These use a variety of different methods including neural networks [221], linear discriminants [199] and Hidden Markov Models [24, 75]. There are also numerous methods for TSS prediction (see, e.g., [97, 171, 201, 241]) and poly-A site prediction (e.g., [136, 207, 123, 220]) with approaches that include neural networks [171], support vector machines [201] and hybrid approaches [97, 123]. These tools provide predictions for gene and transcript boundaries that SpliceGrapherXT could use to provide additional context for interpreting evidence

from RNA-Seq data. Currently SpliceGrapherXT is able to extend genes on either end based on RNA-Seq evidence, but the outer boundaries for exons at the 5' and 3' ends may not be accurate. Incorporating evidence from a TSS or polyA site predictor might improve these predictions as well.

8.1.2 Improving IR Predictions

One of the key findings in our original work was the difficulty in predicting intron retention based on RNA-Seq data [179]. A common problem is that read coverage can vary by orders of magnitude across a gene. Because SpliceGrapherXT currently relies on continuous read coverage across an intron, it will not predict IR when there are short gaps in coverage. Thus we may improve SpliceGrapherXT's ability to predict IR if we can identify gaps in short read coverage that constitute noise.

The challenge is illustrated in Figure 8.1, where SpliceGrapherXT fails to predict an IR event when using only RNA-Seq alignments. EST alignments provide compelling evidence for an IR event, and the RNA-Seq coverage across the same intron appears to support it as well. However, three small gaps in read coverage prevent SpliceGrapherXT from predicting the event. We would like SpliceGrapherXT to distinguish between read coverage gaps that could be merged to form exons, and disjoint clusters of reads that result from noise in the data. It should be possible to develop a classifier that can leverage characteristics of read coverage and splice sites to make these predictions.

Currently we identify gaps in read coverage whenever the coverage drops to 0. This serves as a proxy for identifying when read coverage is indistinguishable from noise. However, a more rigorous approach could give us a better idea of where clusters should be delineated or split apart, and when we may want to merge them. We could develop a classifier that uses features we associate with noise or with legitimate IR events. Gap size is an obvious feature to include if we wish to close gaps by merging adjacent clusters. SpliceGrapherXT's splice site classifiers may also provide evidence for such a model. When splice sites are predicted between two short-read clusters, a gap between them may be due to splice junctions not recapitulated in the spliced alignments. Conversely, the absence of predicted splice sites in the vicinity of a gap could be evidence that the gap was an artifact of noisy data. Additional features may include relevant statistics such as the expression levels of flanking read clusters; the variability in read depth across the gene; and the gene's average read depth.

8.1.3 Additional Comparisons

Throughout this work we have compared SpliceGrapherXT with Cufflinks, IsoLasso and IsoLasso/CEM. Recently the iReckon method was proposed that exhibits higher precision and sensitivity than IsoLasso or Cufflinks for transcript prediction [140]. In simulation and experiments, iReckon yielded 50% higher recall and twice the precision of Cufflinks, IsoLasso or SLIDE in predicting left-out transcripts in a human data set. On real data iReckon yielded recall that was just 10% higher than Cufflinks, but with more than double the precision. The protocol for these experiments was different from our own, so it will be instructive to run iReckon using our experiments.

CLIIQ [119] is another method recently proposed for predicting isoforms and their abundances. This method takes advantage of multiple RNA-Seq samples using an integer linear programming formulation for identifying and quantifying the most parsimonious set of isoforms. In experiments on simulated human data, CLIIQ yields up to 22% higher recall than Cufflinks or IsoLasso, and up to 16% higher precision. Although these differences are not as striking as those for iReckon or SpliceGrapherXT, the experimental protocol was again different from our own. Hence we would like to see how well CLIIQ would perform using our protocols.

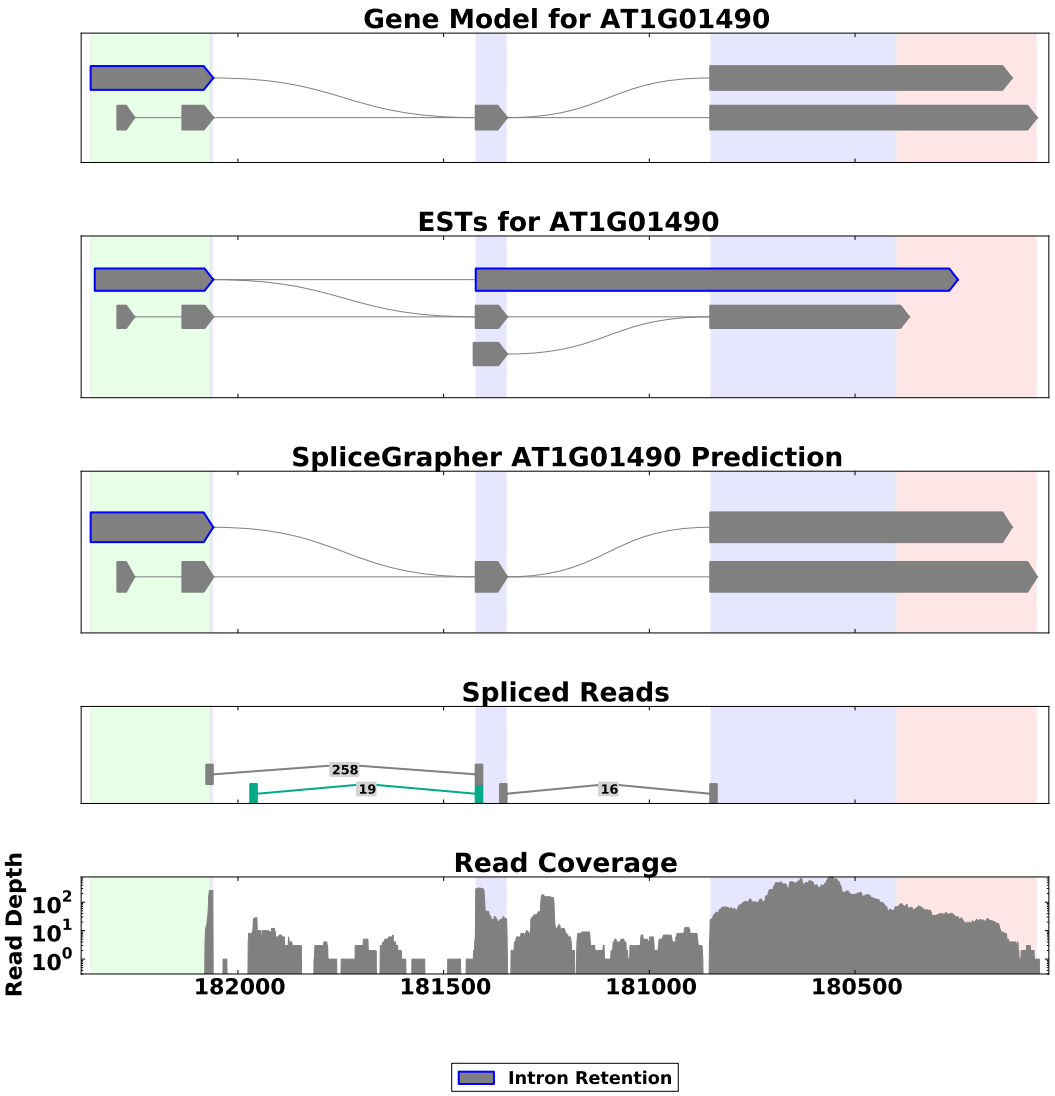

Figure 8.1: An example of a gene where short gaps in read coverage currently prevent SpliceGrapherXT from predicting an IR event. Here we used SpliceGrapherXT to predict AS for this gene based only on RNA-Seq data and compare its prediction with evidence from ESTs. The short-read coverage across the intron on the right provides strong evidence for an IR event, but SpliceGrapherXT is unable to predict the event due to gaps in the coverage.

References

[1] Hoo, September 2011. http://trinityrnaseq.sourceforge.net.

[2] National Center for Biotechnology Information, April 2013. http://www.ncbi.nlm.nih.gov.

[3] P. Akiva, A. Toporik, S. Edelheit, Y. Peretz, A. Diber, R. Shemesh, A. Novik, and R. Sorek. Transcription-mediated gene fusion in the human genome. *Genome Research*, 16(1):30, 2006.

[4] M. Albaqami, M. Hamilton, M.F. Rogers, S.G. Palusa, D. Xing, G.S. Ali, A. Ben-Hur, and A.S.N. Reddy. Global analysis of gene expression and alternative splicing in a splicing regulator SR45 mutant: Role of SR45 in thermotolerance. *under review*.

[5] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J.D. Watson. *Molecular Biology of the Cell*. Garland Science, March 1994.

[6] Gul Shad Ali, Saiprasad G Palusa, Maxim Golovkin, Jayendra Prasad, James L Manley, and Anireddy SN Reddy. Regulation of plant developmental processes by a novel splicing factor. *PLoS One*, 2(5):e471, 2007.

[7] A. Andreadis. Misregulation of tau alternative splicing in neurodegeneration and dementia. *Alternative Splicing and Disease*, pages 89–107, 2006.

[8] P. Arensburger, R.H. Hice, J.A. Wright, N.L. Craig, and P.W. Atkinson. The mosquito Aedes aegypti has a large genome size and high transposable element load but contains a low proportion of transposon-specific piRNAs. *BMC Genomics*, 12(1):606, 2011.

[9] Moritz Aschoff, Agnes Hotz-Wagenblatt, Karl-Heinz Glatting, Matthias Fischer, Roland Eils, and Rainer König. SplicingCompass: differential splicing detection using RNA-Seq data. *Bioinformatics*, 2013.

[10] K.F. Au, H. Jiang, L. Lin, Y. Xing, and W.H. Wong. Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Research*, 38(14):4570, 2010.

[11] Jong-Min Baek, Paul Han, Alberto Iandolino, and Douglas R Cook. Characterization and comparison of intron structure and alternative splicing between Medicago truncatula, Populus trichocarpa, Arabidopsis and rice. *Plant molecular biology*, 67(5):499–510, 2008.

[12] Hua Bao, Yuanyan Xiong, Hui Guo, Renchao Zhou, Xuemei Lu, Zhen Yang, Yang Zhong, and Suhua Shi. MapNext: a software tool for spliced and unspliced alignments and SNP detection of short sequence reads. *BMC genomics*, 10(Suppl 3):S13, 2009.

[13] A Barta, M Kalyna, and ZJ Lorković. Plant sr proteins and their functions. In *Nuclear pre-mRNA Processing in Plants*, pages 83–102. Springer, 2008.

[14] A. Ben-Hur, C.S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch. Support vector machines and kernels for computational biology. *PLoS Computational Biology*, 4(10), 2008.

[15] Asa Ben-Hur. Pyml—a python machine learning package, 2013.

[16] Dennis A Benson, Ilene Karsch-Mizrachi, Karen Clark, David J Lipman, James Ostell, and Eric W Sayers. GenBank. *Nucleic acids research*, 40(D1):D48–D53, 2012.

[17] Elsa Bernard, Laurent Jacob, Julien Mairal, and Jean-Philippe Vert. Flipflop: Fast lasso-based isoform prediction as a flow problem. *under review*, 2013.

[18] Apurva Bhargava, Ivory Clabaugh, Jenn P To, Bridey B Maxwell, Yi-Hsuan Chiang, G Eric Schaller, Ann Loraine, and Joseph J Kieber. Identification of Cytokinin-Responsive Genes Using Microarray Meta-Analysis and RNA-Seq in Arabidopsis. *Plant Physiology*, 162(1):272–294, 2013.

[19] I. Birol, S.D. Jackman, C.B. Nielsen, J.Q. Qian, R. Varhol, G. Stazyk, R.D. Morin, Y. Zhao, M. Hirst, J.E. Schein, et al. De novo transcriptome assembly with ABySS. *Bioinformatics*, 25(21):2872, 2009.

[20] D.L. Black. Mechanisms of alternative pre-messenger RNA splicing. *Annual Review of Biochemistry*, 72:291–336, 2003.

[21] Brunak, Sören and Engelbrecht, Jacob and Knudsen, Steen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, 220(1):49–65, 1991.

[22] V.M. Bruno, Z. Wang, S.L. Marjani, G.M. Euskirchen, J. Martin, G. Sherlock, and M. Snyder. Comprehensive annotation of the transcriptome of the human fungal pathogen Candida albicans using RNA-seq. *Genome Research*, 20(10):1451, 2010.

[23] D.W. Bryant, R. Shen, H.D. Priest, W.K. Wong, and T.C. Mockler. Supersplat–spliced RNA-seq alignment. *Bioinformatics*, 26(12):1500, 2010.

[24] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, 268(1):78–94, 1997.

[25] M Burset, IA Seledtsov, and VV Solovyev. Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic Acids Research*, 28(21):4364–4375, 2000.

[26] Luca Cartegni and Adrian R Krainer. Disruption of an sf2/asf-dependent exonic splicing enhancer in smn2 causes spinal muscular atrophy in the absence of smn1. *Nature genetics*, 30(4):377–384, 2002.

[27] Peter Chang, Joseph Dunham, Sergey Nuzhdin, and Michelle Arbeitman. Somatic sex-specific transcriptome differences in Drosophila revealed by whole transcriptome sequencing. *BMC genomics*, 12(1):364, 2011.

[28] Yao-Fu Chang, J Saadi Imam, and Miles F Wilkinson. The nonsense-mediated decay RNA surveillance pathway. *Annu. Rev. Biochem.*, 76:51–74, 2007.

[29] Jarrod A. Chapman, Isaac Ho, Sirisha Sunkara, Shujun Luo, Gary P. Schroth, and Daniel S. Rokhsar. Meraculous: De Novo Genome Assembly with Short Paired-End Reads. *PLoS ONE*, 6(8):e23501, 08 2011.

[30] Mo Chen and James L Manley. Mechanisms of alternative splicing regulation: insights from molecular and genomics approaches. *Nature Reviews Molecular Cell Biology*, 10(11):741–754, 2009.

[31] Edward B Chuong, MA Karim Rumi, Michael J Soares, and Julie C Baker. Endogenous retroviruses function as species-specific enhancer elements in the placenta. *Nature genetics*, 2013.

[32] N. Cloonan, A.R.R. Forrest, G. Kolle, B.B.A. Gardiner, G.J. Faulkner, M.K. Brown, D.F. Taylor, A.L. Steptoe, S. Wani, G. Bethel, et al. Stem cell transcriptome profiling via massive-scale mRNA sequencing. *Nature Methods*, 5(7):613–619, 2008.

[33] Monica Concha, Xia Wang, Subing Cao, Melody Baddoo, Claire Fewell, Zhen Lin, William Hulme, Dale Hedges, Jane McBride, and Erik K Flemington. Identification of new viral genes and transcript isoforms during epstein-barr virus reactivation using RNA-seq. *Journal of virology*, 86(3):1458–1467, 2012.

[34] Joseph D Coolon, Kraig R Stevenson, C Joel McManus, Brenton R Graveley, and Patricia J Wittkopp. Genomic Imprinting Absent in *Drosophila melanogaster* Adult Females. *Cell Reports*, 2012.

[35] Christine M Costello, Nancy Mah, Robert Häsler, Philip Rosenstiel, Georg H Waetzig, Andreas Hahn, Tim Lu, Yesim Gurbuz, Susanna Nikolaus, Mario Albrecht, et al. Dissection of the inflammatory bowel disease transcriptome using genome-wide cdna microarrays. *PLoS medicine*, 2(8):e199, 2005.

[36] F.H.C. Crick. The biological replication of macromolecules. In *Symposia of the Society for Experimental Biology*, volume 12, pages 138–163, 1958.

[37] Rebecca M Davidson, Malali Gowda, Gaurav Moghe, Haining Lin, Brieanne Vaillancourt, Shin-Han Shiu, Ning Jiang, and C Robin Buell. Comparative transcriptomics of three Poaceae species reveals patterns of gene expression evolution. *The Plant Journal*, 71(3):492–502, 2012.

[38] Irene S Day, Maxim Golovkin, Saiprasad G Palusa, Alicia Link, Gul S Ali, Julie Thomas, Dale N Richardson, and Anireddy SN Reddy. Interactions of SR45, an SR-like protein, with spliceosomal proteins and an intronic sequence: insights into regulated splicing. *The Plant Journal*, 71(6):936–947, 2012.

[39] F. De Bona, S. Ossowski, K. Schneeberger, and G. Rätsch. Optimal spliced alignments of short sequence reads. *BMC Bioinformatics*, 9(Suppl 10):O7, 2008.

[40] Degroeve, Sven and Saeys, Yvan and De Baets, Bernard and Rouzé, Pierre and Van de Peer, Yves. SpliceMachine: predicting splice sites from high-dimensional local context representations. *Bioinformatics*, 21(8):1332–1338, 2005.

[41] M.T. Dimon, K. Sorber, and J.L. DeRisi. HMMSplicer: a tool for efficient and sensitive discovery of known and novel splice junctions in RNA-Seq data. *PloS one*, 5(11):e13875, 2010.

[42] J.C. Dohm, C. Lottaz, T. Borodina, and H. Himmelbauer. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Research*, 36(16):e105, 2008.

[43] Tomislav Domazet-Lošo and Diethard Tautz. A phylogenetically based transcriptome age index mirrors ontogenetic divergence patterns. *Nature*, 468(7325):815–818, 2010.

[44] Nilgun Donmez and Michael Brudno. Hapsembler: an assembler for highly polymorphic genomes. In *Proceedings of the 15th Annual international conference on Research in computational molecular biology*, RECOMB'11, pages 38–52, Berlin, Heidelberg, 2011. Springer-Verlag.

[45] R. Dorn, G. Reuter, and A. Loewendorf. Transgene analysis proves mRNA trans-splicing at the complex mod (mdg4) locus in Drosophila. *Proceedings of the National Academy of Sciences*, 98(17):9724, 2001.

[46] Philipp Drewe, Oliver Stegle, Lisa Hartmann, André Kahles, Regina Bohnert, Andreas Wachter, Karsten Borgwardt, and Gunnar Rätsch. Accurate detection of differential rna processing. *Nucleic Acids Research*, April 2013.

[47] Rachel A Drysdale, Madeline A Crosby, et al. FlyBase: genes and gene models. *Nucleic acids research*, 33(suppl 1):D390–D395, 2005.

[48] Denis Duboule. Temporal colinearity and the phylotypic progression: a basis for the stability of a vertebrate Bauplan and the evolution of morphologies through heterochrony. *Development*, 1994(Supplement):135–142, 1994.

[49] K. Eilbeck, C. Mungall, S. Lewis, and M. Ashburner. The Sequence Ontology Project, January 2009. http://www.sequenceontology.org/gff3.shtml.

[50] EMBL. EMBL European Bioinformatics Institute, April 2013. http://embl.ebi.ac.uk.

[51] ENSEMBL. FTP Download, June 2011. http://uswest.ensembl.org/info/data/ftp/index.html.

[52] ENSEMBL. PSL File Format, April 2013. http://uswest.ensembl.org/info/website/upload/psl.html.

[53] P. Bork et al. Sircah, January 2009. http://www.bork.embl.de/Sircah.

[54] J. Feng, W. Li, and T. Jiang. Inference of Isoforms from Short Sequence Reads. *Journal of Computational Biology*, 18(3):305–321, 2011.

[55] S.A. Filichkin, H.D. Priest, S.A. Givan, R. Shen, D.W. Bryant, S.E. Fox, W.K. Wong, and T.C. Mockler. Genome-wide mapping of alternative splicing in Arabidopsis thaliana. *Genome Research*, 20(1):45, 2010.

[56] Paul Flicek et al. Ensembl 2011. *Nucleic Acids Research*, 39(suppl 1):D800–D806, 2011.

[57] M. Frenkel-Morgenstern, I. Ezkurdia, V. Lacroix, J. Prilusky, A. del Pozo, M. Tress, R. Guigo, and A. Valencia. Chimeras taking shape: potential functions of proteins encoded by chimeric RNA transcripts. *Genome Research (accepted)*, May 2012.

[58] M. Frenkel-Morgenstern, A. Gorohovski, V. Lacroix, M. F. Rogers, K. Ibanez, C. Boullosa, E. Andres Leon, A. Ben-Hur, and A Valencia. ChiTaRS: a database of human, mouse and fruit fly Chimeric Transcripts and RNA-Sequencing data. *Nucleic Acids Research*, 41(D1), 2013.

[59] Chikara Furusawa and Kunihiko Kaneko. Zipfs law in gene expression. *Physical review letters*, 90(8):088102, 2003.

[60] Peter Glaus, Antti Honkela, and Magnus Rattray. Identifying differentially expressed transcripts from RNA-seq data with biological variation. *Bioinformatics*, 28(13):1721–1728, 2012.

[61] Sante Gnerre, Eric Lander, Kerstin Lindblad-Toh, and David Jaffe. Assisted assembly: how to improve a de novo genome assembly by using related species. *Genome Biology*, 10(8):R88, 2009.

[62] David González-Ballester, David Casero, Shawn Cokus, Matteo Pellegrini, Sabeeha S Merchant, and Arthur R Grossman. RNA-seq analysis of sulfur-deprived Chlamydomonas cells reveals aspects of acclimation critical for cell survival. *The Plant Cell Online*, 22(6):2058–2084, 2010.

[63] M.G. Grabherr, B.J. Haas, M. Yassour, J.Z. Levin, D.A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, et al. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature Biotechnology*, 2011.

[64] B.R. Graveley, A.N. Brooks, J.W. Carlson, M.O. Duff, J.M. Landolin, L. Yang, C.G. Artieri, M.J. van Baren, N. Boley, B.W. Booth, et al. The developmental transcriptome of Drosophila melanogaster. *Nature*, 471(7339):473–479, 2010.

[65] Brenton R. Graveley. Sorting out the complexity of sr protein functions. *RNA*, 6(9):1197–1211, 2000.

[66] Thasso Griebel, Benedikt Zacher, Paolo Ribeca, Emanuele Raineri, Vincent Lacroix, Roderic Guigó, and Michael Sammeth. Modelling and simulating generic RNA-Seq experiments with the flux simulator. *Nucleic Acids Research*, 2012.

[67] M. Griffith, O.L. Griffith, J. Mwenifumbo, R. Goya, A.S. Morrissy, R.D. Morin, R. Corbett, M.J. Tang, Y.C. Hou, T.J. Pugh, et al. Alternative expression analysis by RNA sequencing. *Nature Methods*, 7(10):843–847, 2010.

[68] M. Guttman, M. Garber, J.Z. Levin, J. Donaghey, J. Robinson, X. Adiconis, L. Fan, M.J. Koziol, A. Gnirke, C. Nusbaum, et al. Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincRNAs. *Nature Biotechnology*, 28(5):503–510, 2010.

[69] B.J. Haas, A.L. Delcher, S.M. Mount, J.R. Wortman, R.K. Smith Jr, L.I. Hannick, R. Maiti, C.M. Ronning, D.B. Rusch, C.D. Town, et al. Improving the Arabidopsis genome annotation using maximal transcript alignment assemblies. *Nucleic Acids Research*, 31(19):5654, 2003.

[70] B.J. Haas, N. Volfovsky, C.D. Town, M. Troukhan, N. Alexandrov, K.A. Feldmann, R.B. Flavell, O. White, and S.L. Salzberg. Full-length messenger RNA sequences greatly improve genome. *Genome Biology*, 3(6):1–0029, 2002.

[71] Martina Hallegger, Miriam Llorian, and Christopher W. J. Smith. Alternative splicing: global insights. *FEBS Journal*, 277(4):856–866, 2010.

[72] Kasper D. Hansen, Steven E. Brenner, and Sandrine Dudoit. Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Research*, 38(12):e131, 2010.

[73] E.D. Harrington and P. Bork. Sircah: a tool for the detection and visualization of alternative transcripts. *Bioinformatics*, 24(17):1959, 2008.

[74] T. Hashimoto, M.J.L. de Hoon, S.M. Grimmond, C.O. Daub, Y. Hayashizaki, and G.J. Faulkner. Probabilistic resolution of multi-mapping reads in massively parallel sequencing data using MuMRescueLite. *Bioinformatics*, 25(19):2613, 2009.

[75] David Kulp David Haussler and Martin G Reese Frank H Eeckman. A generalized hidden markov model for the recognition of human genes in dna. In *Proc. Int. Conf. on Intelligent Systems for Molecular Biology, St. Louis*, pages 134–142, 1996.

[76] Hebsgaard, Stefan M and Korning, Peter G and Tolstrup, Niels and Engelbrecht, Jacob and Rouzé, Pierre and Brunak, Søren. Splice site prediction in Arabidopsis thaliana pre-mRNA by combining local and global sequence information. *Nucleic Acids Research*, 24(17):3439–3452, 1996.

[77] Jodi D Hoffman, Stephanie E Hallam, Vickie L Venne, Elaine Lyon, and Kenneth Ward. Implications of a novel cryptic splice site in the BRCA1 gene. *American journal of medical genetics*, 80(2):140–144, 1998.

[78] T. Horiuchi, E. Giniger, and T. Aigaki. Alternative trans-splicing of constant and variable exons of a Drosophila axon guidance gene, lola. *Genes & Development*, 17(20):2496, 2003.

[79] Songbo Huang, Jinbo Zhang, Ruiqiang Li, Wenqian Zhang, Zengquan He, Tak-Wah Lam, Zhiyu Peng, and Siu-Ming Yiu. SOAPsplice: genome-wide ab initio detection of splice junctions from RNA-Seq data. *Frontiers in Genetics*, 2(46), 2011.

[80] Yan Huang, Yin Hu, Corbin D Jones, James N MacLeod, Derek Y Chiang, Yufeng Liu, Jan F Prins, and Jinze Liu. A robust method for transcript quantification with RNA-seq data. *Journal of Computational Biology*, 20(3):167–187, 2013.

[81] K. Iida, M. Seki, T. Sakurai, M. Satou, K. Akiyama, T. Toyoda, A. Konagaya, and K. Shinozaki. Genome-wide analysis of alternative pre-mRNA splicing in Arabidopsis thaliana based on full-length cDNA sequences. *Nucleic Acids Research*, 32(17):5096, 2004.

[82] M.K. Iyer, A.M. Chinnaiyan, and C.A. Maher. ChimeraScan: a tool for identifying chimeric transcription in sequencing data. *Bioinformatics*, 27(20):2903–2904, 2011.

[83] G. Jean, A. Kahles, V.T. Sreedharan, F.D. Bona, and G. Rätsch. RNA-Seq Read Alignments with PALMapper. *Current Protocols in Bioinformatics*, 32:11.6.1–11.6.37, 2010.

[84] Jacob K Jensen, Alex Schultink, Kenneth Keegstra, Curtis G Wilkerson, and Markus Pauly. RNA-Seq analysis of developing nasturtium seeds (Tropaeolum majus): identification and characterization of an additional galactosyltransferase involved in xyloglucan biosynthesis. *Molecular Plant*, 5(5):984–992, 2012.

[85] Byeong-Soo Jeong, ATM Golam Bari, Mst Rokeya Reaz, and Ho-Jin Choi. Survey on Nucleotide Encoding Techniques and SVM Kernel Design for Human Splice Site Prediction. *Interdisciplinary Bio Central*, 4(1):14, 2012.

[86] Daniel C. Jones, Walter L. Ruzzo, Xinxia Peng, and Michael G. Katze. A new approach to bias correction in RNA-Seq. *Bioinformatics*, 2012.

[87] Alex T Kalinka, Karolina M Varga, Dave T Gerrard, Stephan Preibisch, David L Corcoran, Julia Jarrells, Uwe Ohler, Casey M Bergman, and Pavel Tomancak. Gene expression divergence recapitulates the developmental hourglass model. *Nature*, 468(7325):811–814, 2010.

[88] Z. Kan, E.C. Rouchka, and W.R. Gish. Gene structure prediction and alternative splicing analysis using genomically aligned ESTs. *Genome Research*, 11(5):889, 2001.

[89] Donna Karolchik, Angela S Hinrichs, Terrence S Furey, Krishna M Roskin, Charles W Sugnet, David Haussler, and W James Kent. The UCSC Table Browser data retrieval tool. *Nucleic acids research*, 32(suppl 1):D493–D496, 2004.

[90] Y. Katz, E.T. Wang, E.M. Airoldi, and C.B. Burge. Analysis and design of RNA sequencing experiments for identifying isoform regulation. *Nature Methods*, 7(12):1009–1015, 2010.

[91] T. Kawasaki, S. Okumura, N. Kishimoto, H. Shimada, K. Higo, and N. Ichikawa. RNA maturation of the rice SPK gene may involve trans-splicing. *The Plant Journal*, 18(6):625–632, 1999.

[92] W.J. Kent. BLAT–the BLAST-like alignment tool. *Genome Research*, 12(4):656, 2002.

[93] H. Keren, G. Lev-Maor, and G. Ast. Alternative splicing and evolution: diversification, exon definition and function. *Nature Reviews Genetics*, 11(5):345–355, 2010.

[94] N. Kim, S. Shin, K.H. Cho, and S. Lee. ChimerDB-database of chimeric sequences in the GenBank. *Genomics & Informatics*, 2(2):61–66, 2004.

[95] M. Kinsella, O. Harismendy, M. Nakano, K.A. Frazer, and V. Bafna. Sensitive gene fusion detection using ambiguously mapping RNA-Seq read pairs. *Bioinformatics*, 27(8):1068, 2011.

[96] Juliane D. Klein, Stephan Ossowski, Korbinian Schneeberger, Detlef Weigel, and Daniel H. Huson. LOCAS – A Low Coverage Assembly Tool for Resequencing Projects. *PLoS ONE*, 6(8):e23455, 08 2011.

[97] S. Knudsen. Promoter2.0: for the recognition of PolII promoter sequences. *Bioinformatics*, 15(5):356–361, 1999.

[98] P.J. Kourlas, M.P. Strout, B. Becknell, M.L. Veronese, C.M. Croce, K.S. Theil, R. Krahe, T. Ruutu, S. Knuutila, C.D. Bloomfield, et al. Identification of a gene at 11q23 encoding a guanine nucleotide exchange factor: evidence for its fusion with MLL in acute myeloid leukemia. *Proceedings of the National Academy of Sciences*, 97(5):2145, 2000.

[99] A. Labadorf, A. Link, M.F. Rogers, J. Thomas, A.S.N. Reddy, and A. Ben-Hur. Genome-wide analysis of alternative splicing in Chlamydomonas reinhardtii. *BMC Genomics*, 11(1):114, 2010.

[100] M. Labrador, F. Mongelard, P. Plata-Rengifo, E.M. Baxter, V.G. Corces, and T.I. Gerasimova. Protein encoding by both DNA strands. *Nature*, 409(6823):1000, 2001.

[101] V. Lacroix, M. Sammeth, R. Guigo, and A. Bergeron. Exact transcriptome reconstruction from short sequence reads. In *Proceedings of the 8th International Workshop on Algorithms in Bioinformatics*, page 63. Springer, 2008.

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

[103] L. LeGault and C. Dewey. Learning Probabilistic Splice Graphs from RNA-Seq data. *under review*, 2013.

[104] B.P. Lewis, R.E. Green, and S.E. Brenner. Evidence for the widespread coupling of alternative splicing and nonsense-mediated mRNA decay in humans. *Proceedings of the National Academy of Sciences*, 100(1):189, 2003.

[105] David D Lewis and William A Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. Springer, 1994.

[106] B. Li, V. Ruotti, R.M. Stewart, J.A. Thomson, and C.N. Dewey. RNA-Seq gene expression estimation with read mapping uncertainty. *Bioinformatics*, 26(4):493, 2010.

[107] H. Li and R. Durbin. Fast and Accurate Short Read Alignment with Burrows-Wheeler Transform. *Bioinformatics*, 2009.

[108] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, and R. Durbin. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078, 2009.

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

[110] H. Li, J. Wang, G. Mor, and J. Sklar. A neoplastic gene fusion mimics trans-splicing of RNAs in normal human cells. *Science*, 321(5894):1357, 2008.

[111] Jingyi Jessica Li, Ci-Ren Jiang, James B. Brown, Haiyan Huang, and Peter J. Bickel. Sparse linear modeling of next-generation mRNA sequencing (RNA-Seq) data for isoform discovery and abundance estimation. *Proceedings of the National Academy of Sciences*, 108(50):19867–19872, 2011.

[112] Jun Li, Hui Jiang, and Wing Wong. Modeling non-uniformity in short-read rates in RNA-Seq data. *Genome Biology*, 11:1–11, 2010. 10.1186/gb-2010-11-5-r50.

[113] Wei Li, Jianxing Feng, and Tao Jiang. IsoLasso: A LASSO Regression Approach to RNA-Seq Based Transcriptome Assembly. In V. Bafna and S. Sahinalp, editors, *Research in Computational Molecular Biology*, volume 6577 of *Lecture Notes in Computer Science*, pages 168–188. Springer Berlin / Heidelberg, 2011.

[114] Wei Li and Tao Jiang. Transcriptome assembly and isoform expression level estimation from biased rna-seq reads. *Bioinformatics*, 28(22):2914–2921, 2012.

[115] X. Li, L. Zhao, H. Jiang, and W. Wang. Short homologous sequences are strongly associated with the generation of chimeric RNAs in eukaryotes. *Journal of Molecular Evolution*, 68(1):56–65, 2009.

[116] Yang Li, Jeremy Chien, David I. Smith, and Jian Ma. FusionHunter: identifying fusion transcripts in cancer using paired-end RNA-seq. *Bioinformatics*, 27(12):1708–1710, 2011.

[117] Z. Li, Z. Zhang, P. Yan, S. Huang, Z. Fei, and K. Lin. RNA-Seq improves annotation of protein-coding genes in the cucumber genome. *BMC Genomics*, 12(1):540, 2011.

[118] Alexander Liede, Beth Y Karlan, and Steven A Narod. Cancer risks for male carriers of germline mutations in brca1 or brca2: a review of the literature. *Journal of Clinical Oncology*, 22(4):735–742, 2004.

[119] Yen-Yi Lin, Phuong Dao, Faraz Hach, Marzieh Bakhshi, Fan Mo, Anna Lapuk, Colin Collins, and S Cenk Sahinalp. CLIIQ: Accurate comparative detection and quantification of expressed isoforms in a population. In *Algorithms in Bioinformatics*, pages 178–189. Springer, 2012.

[120] Zhen Lin, Adriane Puetter, Joseph Coco, Guorong Xu, Michael J Strong, Xia Wang, Claire Fewell, Melody Baddoo, Christopher Taylor, and Erik K Flemington. Detection of murine leukemia virus in the Epstein-Barr virus-positive human B-cell line JY, using a computational RNA-Seq-based exogenous agent detection pipeline, PARSES. *Journal of Virology*, 86(6):2970–2977, 2012.

[121] David J Lipman and William R Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, 1985.

[122] R. Lister, M. Pelizzola, Y.S. Kida, R.D. Hawkins, J.R. Nery, G. Hon, J. Antosiewicz-Bourget, R. OMalley, R. Castanon, S. Klugman, et al. Hotspots of aberrant epigenomic reprogramming in human induced pluripotent stem cells. *Nature*, 471(7336):68–73, 2011.

[123] Huiqing Liu, Hao Han, Jinyan Li, and Limsoon Wong. An in-silico method for prediction of polyadenylation signals in human sequences. *Genome Informatics Series*, pages 84–93, 2003.

[124] J Long and J Caceres. The SR protein family of splicing factors: master regulators of gene expression. *Journal of Biochemistry*, 417:15–27, 2009.

[125] Zdravko J Lorković and Andrea Barta. Genome analysis: Rna recognition motif (rrm) and k homology (kh) domain rna-binding proteins from the flowering plant arabidopsis thaliana. *Nucleic acids research*, 30(3):623–635, 2002.

[126] Susan E Lott, Jacqueline E Villalta, Gary P Schroth, Shujun Luo, Leath A Tonkin, and Michael B Eisen. Noncanonical compensation of zygotic x transcription in early drosophila melanogaster development revealed through single-embryo rna-seq. *PLoS biology*, 9(2):e1000590, 2011.

[127] S.K. Lou, B. Ni, L.Y. Lo, S.K.W. Tsui, T.F. Chan, and K.S. Leung. ABMapper: a suffix array-based tool for multi-location searching and splice-junction mapping. *Bioinformatics*, 27(3):421, 2011.

[128] D.V. Lu, R.H. Brown, M. Arumugam, and M.R. Brent. Pairagon: a highly accurate, HMM-based cDNA-to-genome aligner. *Bioinformatics*, 25(13):1587, 2009.

[129] T. Lu, G. Lu, D. Fan, C. Zhu, W. Li, Q. Zhao, Q. Feng, Y. Zhao, Y. Guo, W. Li, et al. Function annotation of the rice transcriptome at single-nucleotide resolution by RNA-seq. *Genome Research*, 20(9):1238, 2010.

[130] C.A. Maher, N. Palanisamy, J.C. Brenner, X. Cao, S. Kalyana-Sundaram, S. Luo, I. Khrebtukova, T.R. Barrette, C. Grasso, J. Yu, et al. Chimeric transcript discovery by paired-end transcriptome sequencing. *Proceedings of the National Academy of Sciences*, 106(30):12353, 2009.

[131] Christopher A Maher, Chandan Kumar-Sinha, Xuhong Cao, Shanker Kalyana-Sundaram, Bo Han, Xiaojun Jing, Lee Sam, Terrence Barrette, Nallasivam Palanisamy, and Arul M Chinnaiyan. Transcriptome sequencing to detect gene fusions in cancer. *Nature*, 458(7234):97–101, 2009.

[132] Serghei Mangul, Adrian Caciula, Sahar Al Seesi, Dumitru Brinza, Abdul Rouf Banday, and Rahul Kanadia. An integer programming approach to novel transcript reconstruction from paired-end RNA-Seq reads. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 369–376. ACM, 2012.

[133] E.R. Mardis. Next-Generation Sequencing Platforms. *Annual Review of Analytical Chemistry*, 6(1), 2013.

[134] J.C. Marioni, C.E. Mason, S.M. Mane, M. Stephens, and Y. Gilad. RNA-seq: an assessment of technical reproducibility and comparison with gene expression arrays. *Genome Research*, 18(9):1509, 2008.

[135] Yamile Marquez, John WS Brown, Craig Simpson, Andrea Barta, and Maria Kalyna. Transcriptome survey reveals increased complexity of the alternative splicing landscape in Arabidopsis. *Genome Research*, 22(6):1184–1195, 2012.

[136] Sherri Matis, Ying Xu, Manesh Shah, Xiaojun Guan, J Ralph Einstein, Richard Mural, and Edward Uberbacher. Detection of RNA polymerase II promoters and polyadenylation sites in human DNA sequence. *Computers & chemistry*, 20(1):135–140, 1996.

[137] M.G. Mayer and L.M. Floeter-Winter. Pre-mRNA trans-splicing: from kinetoplastids to mammals, an easy language for life diversity. *Memórias do Instituto Oswaldo Cruz*, 100(5):501–513, 2005.

[138] Davis J McCarthy, Yunshun Chen, and Gordon K Smyth. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Research*, 40(10):4288–4297, 2012.

[139] C.J. McManus, M.O. Duff, J. Eipper-Mains, and B.R. Graveley. Global analysis of trans-splicing in Drosophila. *Proceedings of the National Academy of Sciences*, 107(29):12975–12979, 2010.

[140] Aziz M Mezlini, Eric JM Smith, Marc Fiume, Orion Buske, Gleb L Savich, Sohrab Shah, Sam Aparicio, Derek Y Chiang, Anna Goldenberg, and Michael Brudno. iReckon: Simultaneous isoform discovery and abundance estimation from RNA-seq data. *Genome research*, 23(3):519–529, 2013.

[141] F. Mitelman, B. Johansson, and F. Mertens. The impact of translocations and gene fusions on cancer causation. *Nature Reviews Cancer*, 7(4):233–245, 2007.

[142] E. Mizrachi, C.A. Hefer, M. Ranik, F. Joubert, and A.A. Myburg. De novo assembled expressed gene catalog of a fast-growing Eucalyptus tree produced by Illumina mRNA-Seq. *BMC Genomics*, 11(1):681, 2010.

[143] T.C. Mockler and J.R. Ecker. Applications of DNA tiling arrays for whole-genome analysis. *Genomics*, 85(1):1–15, 2005.

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

[145] Nayler, Oliver and Strätling, Wolf and Bourquin, Jean-Pierre and Stagljar, Igor and Lindemann, Lothar and Jasper, Heinrich and Hartmann, Annette M and Fackelmayer, Frank O and Ullrich, Axel and Stamm, Stefan. SAF-B protein couples transcription and pre-mRNA splicing to SAR/MAR elements. *Nucleic Acids Research*, 26(15):3542–3549, 1998.

[146] NCBI. GenBank Home, April 2013. http://www.ncbi.nlm.nih.gov/genbank.

[147] Hadas Ner-Gaon, Ronit Halachmi, Sigal Savaldi-Goldstein, Eitan Rubin, Ron Ophir, and Robert Fluhr. Intron retention is a major phenomenon in alternative splicing in Arabidopsis. *The Plant Journal*, 39(6):877–885, 2004.

[148] M. Nicolae, S. Mangul, I. Măndoiu, and A. Zelikovsky. Estimation of alternative splicing isoform frequencies from RNA-Seq data. *Algorithms in Bioinformatics*, pages 202–214, 2010.

[149] Chad E Niederhuth, O Rahul Patharkar, and John C Walker. Transcriptional profiling of the Arabidopsis abscission mutant hae hsl2 by RNA-Seq. *BMC genomics*, 14(1):37, 2013.

[150] Timothy W Nilsen. The spliceosome: the most complex macromolecular machine in the cell? *Bioessays*, 25(12):1147–1149, 2003.

[151] T.W. Nilsen and B.R. Graveley. Expansion of the eukaryotic proteome by alternative splicing. *Nature*, 463(7280):457–463, 2010.

[152] Alicia Oshlack and Matthew J Wakefield. Transcript length bias in RNA-seq data confounds systems biology. *Biology Direct*, 4(1):14, 2009.

[153] Q. Pan, O. Shai, L.J. Lee, B.J. Frey, and B.J. Blencowe. Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature Genetics*, 2008.

[154] G. Parra, A. Reymond, N. Dabbouseh, E.T. Dermitzakis, R. Castelo, T.M. Thomson, S.E. Antonarakis, and R. Guigó. Tandem chimerism as a means to increase protein complexity in the human genome. *Genome Research*, 16(1):37, 2006.

[155] Till K Pellny, Alison Lovegrove, Jackie Freeman, Paola Tosi, Christopher G Love, J Paul Knox, Peter R Shewry, and Rowan AC Mitchell. Cell walls of developing wheat starchy endosperm: comparison of composition and RNA-Seq transcriptome. *Plant Physiology*, 158(2):612–627, 2012.

[156] Mihaela Pertea, Xiaoying Lin, and Steven L Salzberg. GeneSplicer: a new computational method for splice site prediction. *Nucleic Acids Research*, 29(5):1185–1190, 2001.

[157] Douglas H Phanstiel, Justin Brumbaugh, Craig D Wenger, Shulan Tian, Mitchell D Probasco, Derek J Bailey, Danielle L Swaney, Mark A Tervo, Jennifer M Bolin, Victor Ruotti, et al. Proteomic and phosphoproteomic comparison of human ES and iPS cells. *Nature methods*, 8(10):821–827, 2011.

[158] M.A. Pierotti, M. Santoro, R.B. Jenkins, G. Sozzi, I. Bongarzone, M. Grieco, N. Monzini, M. Miozzo, M.A. Herrmann, and A. Fusco. Characterization of an inversion on the long arm of chromosome 10 juxtaposing D10S170 and RET and creating the oncogenic sequence RET/PTC. *Proceedings of the National Academy of Sciences*, 89(5):1616, 1992.

[159] Anouk Pijpe, Nadine Andrieu, Douglas F Easton, Ausrele Kesminiene, Elisabeth Cardis, Catherine Noguès, Marion Gauthier-Villars, Christine Lasset, Jean-Pierre Fricker, Susan Peock, et al. Exposure to diagnostic radiation and risk of breast cancer among carriers of BRCA1/2 mutations: retrospective cohort study (GENE-RAD-RISK). *BMJ: British Medical Journal*, 345, 2012.

[160] H.D. Priest, D.W. Bryant, S.A. Givan, C. Sullivan, and T.C. Mockler. TAU-an algorithm for reference guided assembly of transcription units from RNA-seq data. *Submitted to Genome Biology*, 2010.

[161] Marcel Quint, Hajk-Georg Drost, Alexander Gabel, Kristian Karsten Ullrich, Markus Bönn, and Ivo Grosse. A transcriptomic hourglass in plant embryogenesis. *Nature*, 490:98–101, October 2012.

[162] Rudolf A Raff. *The shape of life: genes, development, and the evolution of animal form*. University of Chicago Press, Chicago, 1996.

[163] G. Rätsch and S. Sonnenburg. Accurate splice site detection for Caenorhabditis elegans. In Bernhard Schölkopf, Koji Tsuda, and Jean-Phillipe Vert, editors, *Kernel Methods in Computational Biology*, pages 277–298. MIT Press, 2004.

[164] G. Rätsch, S. Sonnenburg, and B. Schölkopf. RASE: recognition of alternatively spliced exons in C. elegans. *Bioinformatics*, 21(Suppl 1):i369–i377, 2005.

[165] A.S.N. Reddy. Plant serine/arginine-rich proteins and their role in pre-mRNA splicing. *Trends in plant science*, 9(11):541–547, 2004.

[166] A.S.N. Reddy. Alternative splicing of pre-messenger RNAs in plants in the genomic era. *Annual Review of Plant Biology*, 58:267–294, 2007.

[167] A.S.N. Reddy. Extensive coupling of alternative splicing of pre-mRNAs of serine/ arginine (SR) genes with nonsense-mediated decay. *New Phytologist*, page 1, 2009.

[168] A.S.N. Reddy, I.S. Day, J. Göhring, and A. Barta. Localization and dynamics of plant splicing regulators. *Plant Physiology*, 2011.

[169] A.S.N. Reddy, M.F. Rogers, D. N. Richardson, M. Hamilton, and A. Ben-Hur. Deciphering the plant splicing code: experimental and computational approaches for predicting alternative splicing and splicing regulatory elements. *Frontiers in Plant Genetics and Genomics*, 3(0), 2012.

[170] Martin G Reese, Frank H Eeckman, David Kulp, and David Haussler. Improved splice site detection in Genie. *Journal of Computational Biology*, 4(3):311–323, 1997.

[171] M.G. Reese. Application of a time-delay neural network to promoter annotation in the Drosophila melanogaster genome. *Computers & Chemistry*, 26(1):51–56, 2001.

[172] D.N. Richardson, M.F. Rogers, A. Labadorf, A. Ben-Hur, H. Guo, A.H. Paterson, and A.S.N. Reddy. Comparative Analysis of Serine/Arginine-Rich Proteins across 27 Eukaryotes: Insights into Sub-Family Classification and Extent of Alternative Splicing. *PLoS ONE*, 6(9):e24542, 09 2011.

[173] A. Roberts, H. Pimentel, C. Trapnell, and L. Pachter. Identification of novel transcripts in annotated genomes using RNA-Seq. *Bioinformatics*, 27(17):2325–2329, 2011.

[174] Gordon Robertson, Jacqueline Schein, Readman Chiu, Richard Corbett, Matthew Field, Shaun D Jackman, Karen Mungall, Sam Lee, Hisanaga Mark Okada, Jenny Q Qian, et al. De novo assembly and analysis of RNA-seq data. *Nature methods*, 7(11):909–912, 2010.

[175] H.M. Robertson, J.A. Navik, K.K.O. Walden, and H.W. Honegger. The bursicon gene in mosquitoes: an unusual example of mRNA trans-splicing. *Genetics*, 176(2):1351, 2007.

[176] Xavier Roca, Adrian R Krainer, and Ian C Eperon. Pick one, but be quick: 5 splice sites and the problems of too many choices. *Genes & Development*, 27(2):129–144, 2013.

[177] M. F. Rogers, C. Boucher, and A. Ben-Hur. SpliceGrapherXT: from splice graphs to transcripts using RNA-Seq. *under review*.

[178] M.F. Rogers. SpliceGrapher, April 2011. http://splicegrapher.sourceforge.net.

[179] M.F. Rogers, J. Thomas, A.S.N. Reddy, and A. Ben-Hur. SpliceGrapher: Detecting patterns of alternative splicing from RNA-seq data in the context of gene models and EST data. *Genome Biology*, 13(R4), 2012.

[180] Antonello Romani, Emanuela Guerra, Marco Trerotola, and Saverio Alberti. Detection and analysis of spliced chimeric mRNAs in sequence databanks. *Nucleic Acids Research*, 31(4):e17, 2003.

[181] Roy Ronen, Christina Boucher, Hamidreza Chitsaz, and Pavel Pevzner. Sequel: improving the accuracy of genome assemblies. *Bioinformatics*, 28(12):i188–i196, 2012.

[182] Tanja Dorothe Rösel, Lee-Hsueh Hung, Jan Medenbach, Katrin Donde, Stefan Starke, Vladimir Benes, Gunnar Rätsch, and Albrecht Bindereif. RNA-Seq analysis in mutant zebrafish reveals role of U1C protein in alternative splicing regulation. *The EMBO journal*, 30(10):1965–1976, 2011.

[183] Julien Roux and Marc Robinson-Rechavi. Age-dependent gain of alternative splice forms and biased duplication explain the relation between splicing and duplication. *Genome Research*, 21(3):357–363, 2011.

[184] Jesse W Rowley, Andrew J Oler, Neal D Tolley, Benjamin N Hunter, Elizabeth N Low, David A Nix, Christian C Yost, Guy A Zimmerman, and Andrew S Weyrich. Genome-wide RNA-seq analysis of human and mouse platelet transcriptomes. *Blood*, 118(14):e101–e111, 2011.

[185] Barbara Ruskin, Phillip D Zamore, and Michael R Green. A factor, u2af, is required for u2 snrnp binding and splicing complex assembly. *Cell*, 52(2):207–219, 1988.

[186] Marta Sanchez-Carbayo, Nicholas D Socci, Juan Jose Lozano, Wentian Li, Elizabeth Charytonowicz, Thomas J Belbin, Michael B Prystowsky, Angel R Ortiz, Geoffrey Childs, and Carlos Cordon-Cardo. Gene discovery in bladder cancer progression using cDNA microarrays. *The American journal of pathology*, 163(2):505–516, 2003.

[187] JR Sanford, D Longman, and JF Caceres. Multiple roles of the SR protein family in splicing regulation. In *Regulation of Alternative Splicing*, pages 33–58. Springer, 2003.

[188] F. Sanger, S. Nicklen, and A.R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463, 1977.

[189] E.W. Sayers, T. Barrett, D.A. Benson, E. Bolton, S.H. Bryant, K. Canese, V. Chetvernin, D.M. Church, M. DiCuccio, S. Federhen, et al. Database resources of the national center for biotechnology information. *Nucleic Acids Research*, 39(suppl 1):D38, 2011.

[190] A. Sboner, L. Habegger, D. Pflueger, S. Terry, D.Z. Chen, J.S. Rozowsky, A.K. Tewari, N. Kitabayashi, B.J. Moss, M.S. Chee, et al. FusionSeq: a modular framework for finding gene fusions by analyzing paired-end RNA-sequencing data. *Genome Biology*, 11(10):R104, 2010.

[191] B Schoölkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel Methods in Computational Biology*. The MIT Press, 2004.

[192] U. Schulze, B. Hepp, C.S. Ong, and G. Rätsch. PALMA: mRNA to genome alignments using large margin algorithms. *Bioinformatics*, 23(15):1892, 2007.

[193] Schraga Schwartz, Ram Oren, and Gil Ast. Detection and removal of biases in the analysis of next-generation sequencing reads. *PloS one*, 6(1):e16685, 2011.

[194] M. Seki, M. Narusaka, A. Kamiya, J. Ishida, M. Satou, T. Sakurai, M. Nakajima, A. Enju, K. Akiyama, Y. Oono, et al. Functional annotation of a full-length Arabidopsis cDNA collection. *Science*, 296(5565):141, 2002.

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

[196] J.T. Simpson, K. Wong, S.D. Jackman, J.E. Schein, S.J.M. Jones, and İ. Birol. ABySS: a parallel assembler for short read sequence data. *Genome Research*, 19(6):1117, 2009.

[197] N.N. Singh and R.N. Singh. Alternative splicing in spinal muscular atrophy underscores the role of an intron definition model. *RNA Biology*, 8(4):600–606, 2011.

[198] D. Solnick. Trans splicing of mRNA precursors. *Cell*, 42(1):157–164, 1985.

[199] Victor V Solovyev, Asaf A Salamov, and Charles B Lawrence. Predicting internal exons by oligonucleotide composition and discriminant analysis of spliceable open reading frames. *Nucleic Acids Research*, 22(24):5156–5163, 1994.

[200] Sören Sonnenburg, Gabriele Schweikert, Petra Philips, Jonas Behr, and Gunnar Rätsch. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8(Suppl 10):S7, 2007.

[201] Sören Sonnenburg, Alexander Zien, and Gunnar Rätsch. ARTS: accurate recognition of transcription starts in human. *Bioinformatics*, 22(14):e472–e480, 2006.

[202] Mario Stanke and Stephan Waack. Gene prediction with a hidden Markov model and a new intron submodel. *Bioinformatics*, 19(suppl 2):ii215–ii225, 2003.

[203] J.E. Sulston and H.R. Horvitz. Post-embryonic cell lineages of the nematode, Caenorhabditis elegans. *Developmental Biology*, 56(1):110–156, 1977.

[204] M. Sultan, M.H. Schulz, H. Richard, A. Magen, A. Klingenhoff, M. Scherf, M. Seifert, T. Borodina, A. Soldatov, D. Parkhomchuk, et al. A global view of gene activity and alternative splicing by deep sequencing of the human transcriptome. *Science*, 321(5891):956–959, 2008.

[205] Ying-Fei Sun, Xiao-Dan Fan, and Yan-Da Li. Identifying splicing sites in eukaryotic RNA: support vector machine approach. *Computers in Biology and Medicine*, 33(1):17–29, 2003.

[206] D. Swarbreck, C. Wilks, P. Lamesch, T.Z. Berardini, M. Garcia-Hernandez, H. Foerster, D. Li, T. Meyer, R. Muller, L. Ploetz, et al. The Arabidopsis Information Resource (TAIR): gene structure and function annotation. *Nucleic Acids Research*, 36(suppl 1):D1009, 2008.

[207] Jack E Tabaska and Michael Q Zhang. Detection of polyadenylation signals in human DNA sequences. *Gene*, 231(1):77–86, 1999.

[208] TAIR. TAIR Breaking News, November 2010. http://www.arabidopsis.org/doc/news/breaking_news/140.

[209] F. Tang, C. Barbacioru, Y. Wang, E. Nordman, C. Lee, N. Xu, X. Wang, J. Bodeau, B.B. Tuch, A. Siddiqui, et al. mRNA-Seq whole-transcriptome analysis of a single cell. *Nature Methods*, 6(5):377–382, 2009.

[210] S. Tang and A. Riva. PASTA: splice junction identification from RNA-Sequencing data. *BMC Bioinformatics*, 14(16), April 2013.

[211] Zhaohua Tang, Norbert F Käufer, and Ren-Jang Lin. Interactions between two fission yeast serine/arginine-rich proteins and their modulation by phosphorylation. *Biochemical Journal*, 368(Pt 2):527, 2002.

[212] J. Tazi, N. Bakkour, and S. Stamm. Alternative splicing and disease. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1792(1):14–26, 2009.

[213] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[214] Alexandru I Tomescu, Anna Kuosmanen, Romeo Rizzi, and Veli Mäkinen. A novel min-cost flow method for estimating transcript expression with rna-seq. *BMC Bioinformatics*, 14(Suppl 5):S15, 2013.

[215] J.M. Toung, M. Morley, M. Li, and V.G. Cheung. RNA-sequence analysis of human B-cells. *Genome Research*, 21(6):991–998, 2011.

[216] C. Trapnell, L. Pachter, and S.L. Salzberg. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 2009.

[217] C. Trapnell and S.L. Salzberg. How to map billions of short reads onto genomes. *Nature Biotechnology*, 27(5):455–457, 2009.

[218] C. Trapnell, B.A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M.J. van Baren, S.L. Salzberg, B.J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 2010.

[219] N.A. Twine, K. Janitz, M.R. Wilkins, and M. Janitz. Whole transcriptome sequencing reveals gene expression and splicing differences in brain regions affected by Alzheimer's disease. *PLoS ONE*, 6(1):e16266, 2011.

[220] George Tzanis, Ioannis Kavakiotis, and Ioannis Vlahavas. PolyA-iEP: A data mining method for the effective prediction of polyadenylation sites. *Expert Systems with Applications*, 38(10):12398–12408, 2011.

[221] Edward C Uberbacher and Richard J Mural. Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach. *Proceedings of the National Academy of Sciences*, 88(24):11261–11265, 1991.

[222] UCSC. UCSC Genome Bioinformatics, April 2013. http://genome.ucsc.edu.

[223] Jernej Ule, Giovanni Stefani, Aldo Mele, Matteo Ruggiu, Xuning Wang, Bahar Taneri, Terry Gaasterland, Benjamin J Blencowe, and Robert B Darnell. An RNA map predicting Nova-dependent splicing regulation. *Nature*, 444(7119):580–586, 2006.

[224] UniProt. Uniprot release 2013-03, March 2013. http://www.uniprot.org/news/2013/03/06/release.

[225] C.J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2 edition, March 1979.

[226] J.P. Venables. Aberrant and alternative splicing in cancer. *Cancer Research*, 64(21):7647–7654, 2004.

[227] B.B. Wang and V. Brendel. Genomewide comparative analysis of alternative splicing in plants. *Proceedings of the National Academy of Sciences*, 103(18):7175, 2006.

[228] E.T. Wang, R. Sandberg, S. Luo, I. Khrebtukova, L. Zhang, C. Mayr, S.F. Kingsmore, G.P. Schroth, and C.B. Burge. Alternative isoform regulation in human tissue transcriptomes. *Nature*, 456(7221):470–476, 2008.

[229] Furu Wang, Qiaoqiao Fang, Zhen Ge, Ningle Yu, Sanxiao Xu, and Xiangyong Fan. Common BRCA1 and BRCA2 mutations in breast cancer families: a meta-analysis from systematic review. *Molecular Biology Reports*, 39(3):2109–2118, 2012.

[230] K. Wang, D. Singh, Z. Zeng, S.J. Coleman, Y. Huang, G.L. Savich, X. He, P. Mieczkowski, S.A. Grimm, C.M. Perou, et al. MapSplice: Accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Research*, 2010.

[231] Lei Wang, Lindsay Duke, Peter S Zhang, Ralph B Arlinghaus, W Fraser Symmans, Aysegul Sahin, Richard Mendez, and Jia Le Dai. Alternative splicing disrupts a nuclear localization signal in spleen tyrosine kinase that is required for invasion suppression in breast cancer. *Cancer Research*, 63(15):4724–4730, 2003.

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

[233] Judson A Ward, Lalit Ponnala, and Courtney A Weber. Strategies for transcriptome analysis in nonmodel plants. *American Journal of Botany*, 99(2):267–276, 2012.

[234] Maria Warnefors and Adam Eyre-Walker. The accumulation of gene regulation through time. *Genome Biology and Evolution*, 3:667, 2011.

[235] Robert F. Weaver. *Molecular Biology*. McGraw Hill, Boston, MA, third edition, 1999.

[236] J. Wu, M. Akerman, S. Sun, W.R. McCombie, A.R. Krainer, and M.Q. Zhang. SpliceTrap: a method to quantify alternative splicing under single cellular conditions. *Bioinformatics*, 2011.

[237] Jia Qian Wu, Lukas Habegger, Parinya Noisa, Anna Szekely, Caihong Qiu, Stephen Hutchison, Debasish Raha, Michael Egholm, Haifan Lin, Sherman Weissman, et al. Dynamic transcriptomes during neural differentiation of human embryonic stem cells revealed by short, long, and paired-end sequencing. *Proceedings of the National Academy of Sciences*, 107(11):5254–5259, 2010.

[238] T.D. Wu and C.K. Watanabe. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*, 21(9):1859, 2005.

[239] Z. Xia, J. Wen, C Chang, and X Zhou. NSMAP: A Method for Spliced Isoforms Identification and Quantification from RNA-Seq. *BMC Bioinformatics*, 12(162), May 2011.

[240] Rui Xiao, Peng Tang, Bo Yang, Jie Huang, Yu Zhou, Changwei Shao, Hairi Li, Hui Sun, Yi Zhang, and Xiang-Dong Fu. Nuclear Matrix Factor hnRNP U/SAF-A Exerts a Global Control of Alternative Splicing by Regulating U2 snRNP Maturation. *Molecular Cell*, 45(5):656–668, March 2012.

[241] Xudong Xie, Shuanhu Wu, Kin-Man Lam, and Hong Yan. PromoterExplorer: an effective promoter identification method based on the AdaBoost algorithm. *Bioinformatics*, 22(22):2722–2728, 2006.

[242] Augix Guohua Xu, Liu He, Zhongshan Li, Ying Xu, Mingfeng Li, Xing Fu, Zheng Yan, Yuan Yuan, Corinna Menzel, Na Li, et al. Intergenic and repeat transcription in human, chimpanzee and macaque brains measured by RNA-Seq. *PLoS Computational Biology*, 6(7):e1000843, 2010.

[243] Guorong Xu, Claire Fewell, Christopher Taylor, Nan Deng, Dale Hedges, Xia Wang, Kun Zhang, Michelle Lacey, Haitao Zhang, Qinyan Yin, et al. Transcriptome and targetome analysis in mir155 expressing cells using rna-seq. *Rna*, 16(8):1610–1622, 2010.

[244] Y. Yang and C.E. Walsh. Spliceosome-mediated RNA trans-splicing. *Molecular Therapy*, 12(6):1006–1012, 2005.

[245] Z. Yang, YU HongTshi, LU Heng, YAO Kai, C. HanThua, and Z. RongTjia. Interchromosomal trans-splicing of DMRT1 gene on chicken chromosome Z. *Zoological Research*, 27(2):175–180, 2006.

[246] M. Yassour, T. Kaplan, H.B. Fraser, J.Z. Levin, J. Pfiffner, X. Adiconis, G. Schroth, S. Luo, I. Khrebtukova, A. Gnirke, et al. Ab initio construction of a eukaryotic transcriptome by massively parallel mRNA sequencing. *Proceedings of the National Academy of Sciences*, 106(9):3264, 2009.

[247] Alan M Zahler, William S Lane, John A Stolk, and Mark B Roth. SR proteins: a conserved family of pre-mRNA splicing factors. *Genes & development*, 6(5):837–847, 1992.

[248] S. Zenoni, A. Ferrarini, E. Giacomelli, L. Xumerle, M. Fasoli, G. Malerba, D. Bellin, M. Pezzotti, and M. Delledonne. Characterization of transcriptional complexity during berry development in Vitis vinifera using RNA-Seq. *Plant Physiology*, 152(4):1787, 2010.

[249] Lirong Zhang and Liaofu Luo. Splice site prediction with quadratic discriminant analysis using diversity measure. *Nucleic Acids Research*, 31(21):6214–6220, 2003.

[250] X.H.F. Zhang and L.A. Chasin. Comparison of multiple vertebrate genomes reveals the birth and evolution of human exons. *Proceedings of the National Academy of Sciences*, 103(36):13427, 2006.

[251] Zhi Ming Zheng, Martijn Huynen, and Carl C Baker. A pyrimidine-rich exonic splicing suppressor binds multiple rna splicing factors and inhibits spliceosome assembly. *Proceedings of the National Academy of Sciences*, 95(24):14088–14093, 1998.

[252] H. Zhu, H.M. Tucker, K.E. Grear, J.F. Simpson, A.K. Manning, L.A. Cupples, and S. Estus. A common polymorphism decreases low-density lipoprotein receptor exon 12 splicing efficiency and associates with increased cholesterol. *Human Molecular Genetics*, 16(14):1765–1772, 2007.

Appendix A

Original SpliceGrapher Inference Rules

In our original SpliceGrapher implementation we used a set of inference rules designed to predict specific types of AS events from RNA-Seq evidence. Each of the data sources used by SpliceGrapher—genome annotations, short-read data, and EST alignments—requires a distinct interpretation for splice graph construction. For the following discussion of the procedures used to interpret each type of data we borrow terminology from [73] and refer to exons that have explicit acceptor and donor splice sites as *bounded* and those with an undefined acceptor or donor splice site as *unbounded*. When one graph element (an exon or an intron) falls completely within the genomic coordinates of another graph element, we say that it is *contained* within the other element.

A.1 Alternative Splicing Inference from RNA-Seq

Because of the relatively short length of RNA-Seq reads, they cannot be unambiguously interpreted as splice-graphs and AS events (see Figure A.1). SpliceGrapher's original approach was to use as much data as possible to make confident predictions, and to annotate AS events as *unresolved* if the evidence did not clearly support a specific isoform. This original version of SpliceGrapher applies inference rules in the order presented in the following sections.

A.1.1 Intron Retention

Intron retention (IR) is arguably the most challenging form of AS to infer from RNA-Seq data: the best evidence for novel IR events comes from ungapped alignments across annotated introns, but sequencing and alignment artifacts such as unprocessed pre-mRNA or gaps in read coverage make it difficult to discriminate between IR events and noise [179, 140]. SpliceGrapher infers IR events from RNA-Seq evidence in two ways that exploit information from the gene models. When short-read coverage remains above a desired threshold across an intron's full length, it is evidence that the intron was retained in some transcripts (see Figure 3.2). In this case the intron is excised in the constitutive form represented by the gene model. In an alternative scenario shown in Figure A.2, the intron is retained in the known constitutive form. We detect this form of IR when a known exon has a novel splice junction within it.

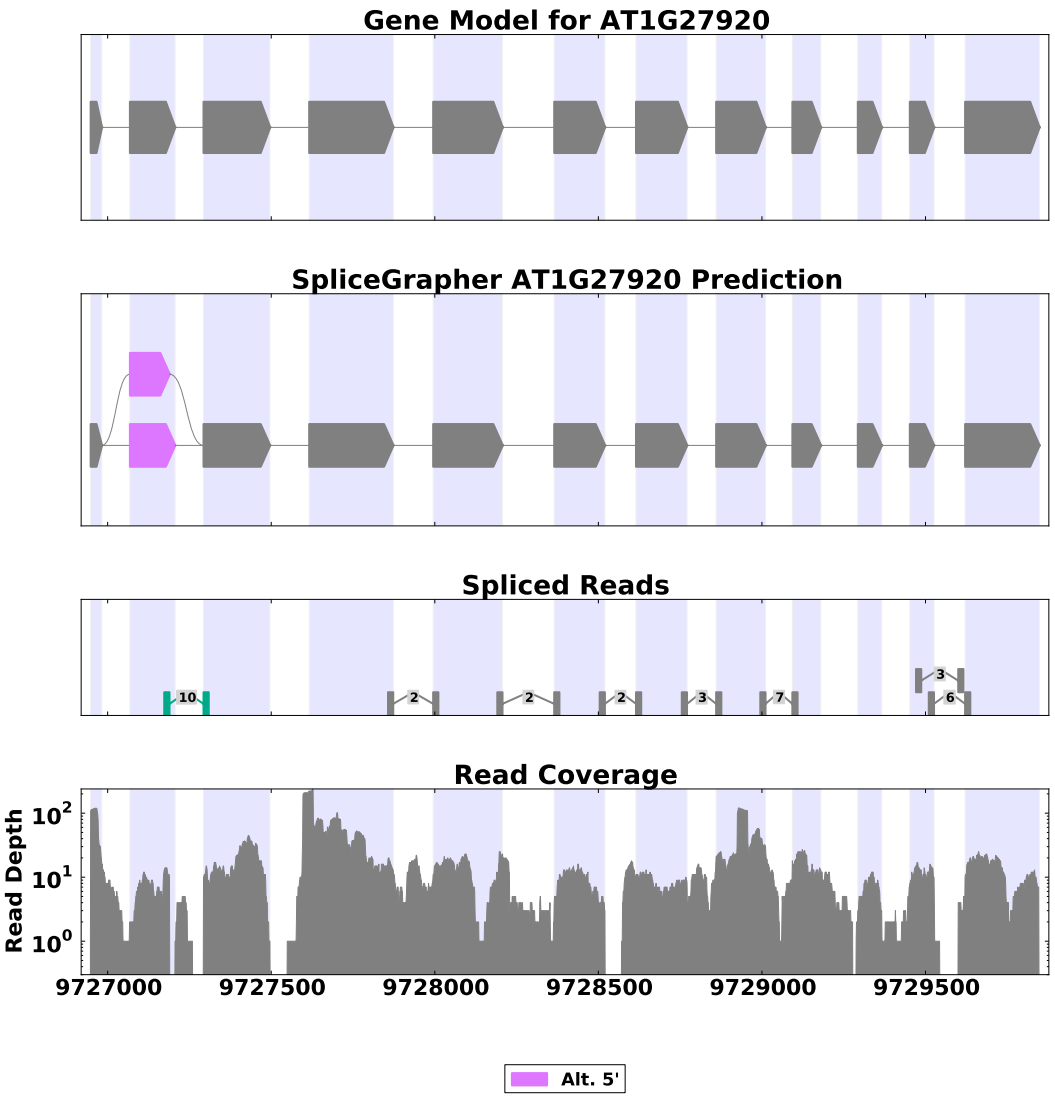

Figure A.1: This figure demonstrates ambiguities that arise in RNA-Seq data that make isoform prediction challenging. Because there is read coverage across several introns, SpliceGrapher is not able to determine whether this is a result of a single intron retention event, or several independent events.

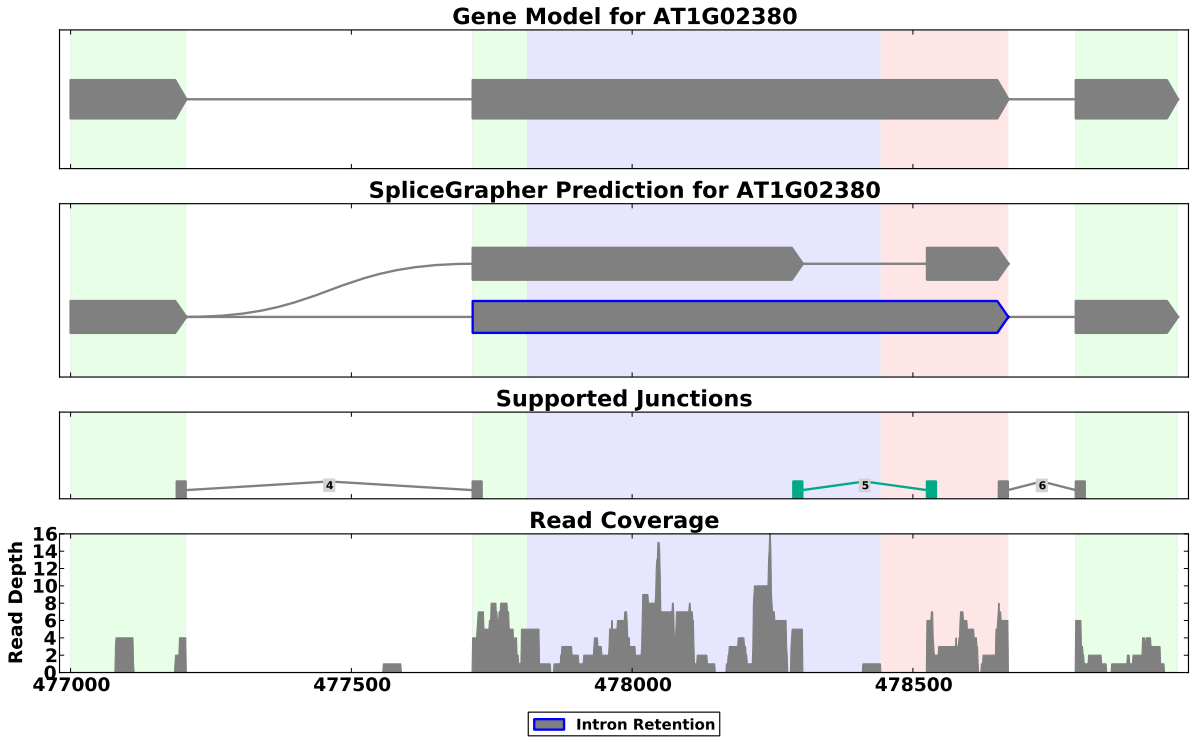

Figure A.2: *A. thaliana* gene AT1G02380 provides an example where the known splice form is the one with the intron retained (as opposed to the example shown in Figure 3.2 where the novel splice form has the intron as retained). This scenario of intron retention requires different evidence, namely the novel splice junction, shown in green in the figure. The boundaries of the exon from the gene model were used to infer the boundaries of the exons that flank the new intron.

When SpliceGrapher infers a novel IR event, it must identify unique exon boundaries for three exons: the longer exon in which the intron is retained, and the two shorter exons that flank the intron when it is excised. Usually the gene model provides good evidence for these boundaries, but in some cases it may not be possible to resolve them unambiguously.

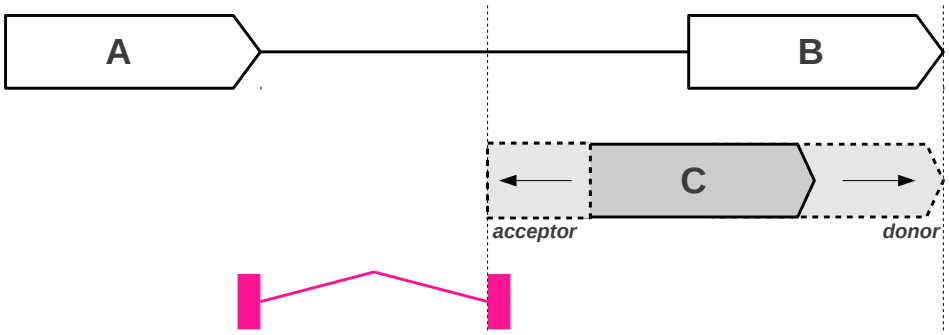

Figure A.3: The method SpliceGrapher uses to resolve an alternate acceptor site when a short-read exon (C) extends beyond another exon (B). If the exon C is not directly flanked by splice junctions (known or predicted from RNA-Seq data), SpliceGrapher will attempt to extend the exon to the nearest splice junctions, upstream and downstream. If such an extension is not supported by the data, the exon is considered unresolved.

When short-read coverage remains high across an intron's full length, SpliceGrapher will create a short-read exon that spans the intron. Its next task is to determine the exon's correct boundaries. SpliceGrapher first finds all exons in the graph that overlap the short-read exon. If one upstream exon overlaps its 5' end and one downstream exon overlaps its 3' end, SpliceGrapher creates a new exon whose boundaries are the acceptor site from the upstream exon and the donor site from the downstream exon. If more than one exon overlaps either end of the short-read exon, it is still possible to infer the new exon's boundaries provided all overlapping upstream exons share the same acceptor site and overlapping downstream exons share the same donor site (Figure A.3). If the boundary at either end is ambiguous, SpliceGrapher creates an *unresolved* IR event. When a junction is used to infer an IR event through the scenario of a splice junction within an exon, SpliceGrapher must identify the boundaries for two new exons, which is performed in a manner analogous to the single exon case.

In some cases read coverage may remain high across two or more introns in succession, making it impossible to determine which of several possible splice forms is correct (see Figure A.1). In these cases, SpliceGrapher annotates the corresponding short-read exon as *unresolved*.

A.1.2 Alternative 3' and 5' Events

When an intron is excised at more than one splice site, changing the boundaries of one of its flanking exons, we have evidence of an alternate 3' or 5' event. SpliceGrapher uses two forms of evidence to infer alternative 3'/5' events. When a short-read exon overlaps an existing exon but extends beyond its 3' or 5' end, it provides evidence for an alternate donor or acceptor site. In addition, when a novel splice junction appears between two exons it provides evidence for a novel intron (Figure A.3). SpliceGrapher requires both forms of evidence to infer a novel alternative splice site. Below we describe the procedure for inferring an alternate acceptor (3' site). The procedure for an alternate donor (5' site) is analogous.

When a short-read exon overlaps an existing exon and extends into its upstream intron, it is evidence that the exon boundaries changed in some transcripts. To identify an acceptor site for the new exon, SpliceGrapher looks for junctions that have acceptor sites within the same intron, upstream of the short-read exon (Figure A.3). SpliceGrapher then uses the acceptor site nearest the short-read exon as its acceptor site. If it finds no acceptor sites within the intron, SpliceGrapher annotates the short-read exon as *unresolved*.

If SpliceGrapher can resolve a new exon's acceptor site, it must resolve its donor site as well. The procedure is the same as that for identifying retained intron boundaries in the previous section. If one downstream exon overlaps the new exon's 3' end, SpliceGrapher uses the downstream exon's donor site as the new exon's donor site. If more than one downstream exon overlaps the new exon's 3' end, SpliceGrapher can still resolve its donor site provided all overlapping exons share the same donor site (see Figure A.3). If the overlapping exons have different donor sites, SpliceGrapher cannot resolve the new exon's donor, and it annotates the exon as *unresolved*.

A.1.3 Exon Skipping

An exon skipping event occurs when an exon is excised from some transcripts but included in others. SpliceGrapher infers skipped exons in two different ways. In the first scenario the exon is included in the known constitutive form represented in the gene model. If a novel splice junction spans the existing exon, it is evidence that the exon was skipped in some transcripts (see top panel of Figure A.4). If the novel junction's acceptor and donor sites match those of established exons, SpliceGrapher adds the new intron to the graph and annotates the skipped exon.

An alternate scenario is when the exon is skipped in the constitutive form (bottom panel of Figure A.4). In this case, if a short-read exon falls within an intron and is flanked by two novel junctions, it is evidence

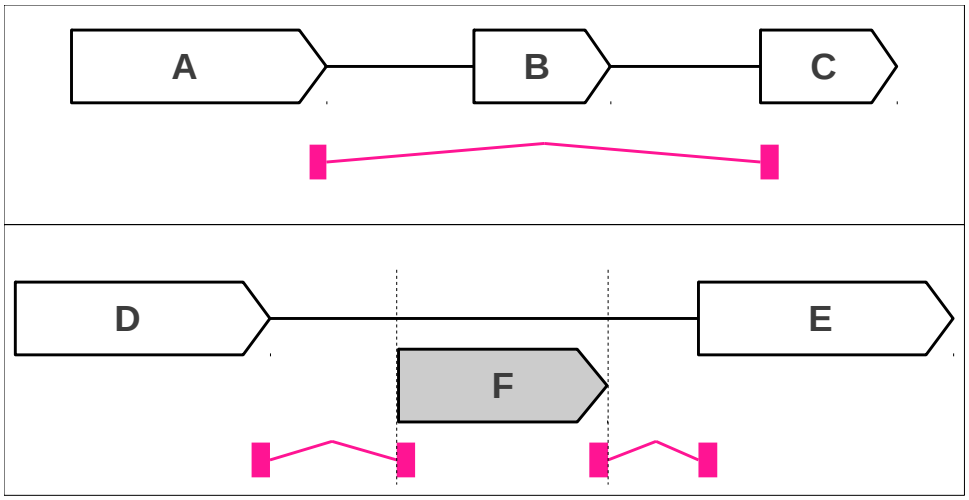

Figure A.4: Two scenarios in which SpliceGrapher will predict a skipped exon. In the top panel, a novel splice junction between exons A and C provides evidence that exon B is skipped in some transcripts. In the lower panel a novel exon, F, is contained within an intron in the graph. The new exon is flanked by novel splice junctions within the same intron, so SpliceGrapher uses the acceptor site from the upstream junction to resolve the exon's 5' boundary and the donor site from the downstream junction to resolve its 3' boundary. If SpliceGrapher is unable to resolve either boundary, the exon is unresolved.

for a novel exon that is skipped in some transcripts. These clues may not provide enough evidence to resolve the event, so SpliceGrapher tries to associate the upstream junction's donor site and the downstream junction's acceptor site with exons in the graph. If this first step is successful, SpliceGrapher uses the upstream junction's acceptor site as the new exon's acceptor site and the downstream junction's donor site as the new exon's donor site. If it is unable to resolve the junctions, SpliceGrapher annotates the short-read exon as *unresolved*.

Appendix B

Counting AS Events

An important consequence of predicting splice graphs is the ability to annotate novel AS events and to generate statistics from them. Genome-wide AS surveys can provide insights into how organisms develop and how they respond to different conditions [99, 227, 11, 55, 81, 147]. To generate statistics we count the AS events within each splice graph and sum over the graphs predicted for a data set.

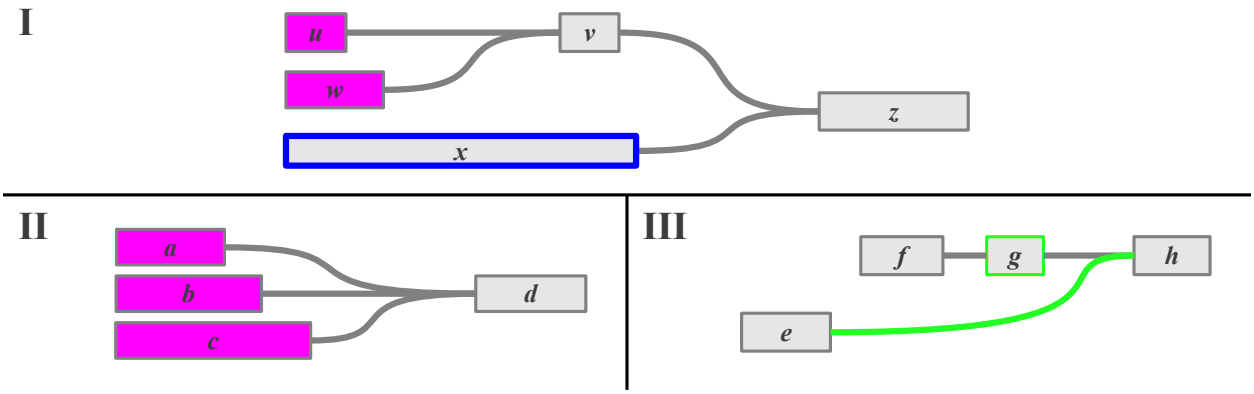

Figure B.1: Examples showing how SpliceGrapher counts different AS events. (I) Exon x contains the introns from u to v and w to v. This is counted as a single IR event (plus an alternative 5' event) because only one exon (x) retains the introns. (II) Exons a, b and c overlap one another and have three distinct donor sites. Together these are counted as two alternative 5' events. (III) The intron between exons e and h skips exon g, which is counted as an ES event. Exon f is an alternative transcription start site that we do not count as AS.

B.1 Intron retention

Intron retention occurs when the splicing machinery refrains from excising an intron, resulting in a single long exon comprised of the intron and its flanking exons. Thus whenever an exon contains one or more introns in a graph, we count it as an IR event. If two or more exons contain the same introns, they are counted as separate IR events. Example I in Figure B.1 is an example of a graph that has a single IR event containing two distinct introns.

B.2 Alternative 3' and 5' events

An alternative 5' event occurs when an intron is excised at two or more sites at its 5' end. These splice sites correspond to donor sites for different exons in a graph. Thus to count alternative 5' events we search a graph for overlapping exons that use different donor sites. We consider one of these sites as a baseline representing the prevalent form, and the remaining sites as alternatives. The total count is then simply the number of distinct donor sites in the set minus 1. Example II in Figure B.1 shows a graph with two alternative 5' events. One exon may be selected arbitrarily as a baseline while the other exons with distinct donor sites are counted as alternative 5' events. We use an analogous approach to count alternative 3' events using exon acceptor sites.

B.3 Exon skipping

An ES event occurs when a cell's splicing machinery excises an exon from a transcript along with its flanking introns. Our evidence for an ES event in a splice graph is an intron that contains one or more exons (Figure B.1, example III). If multiple distinct introns span the same exons, we count each one as a separate ES event.

When counting ES events we make a distinction between alternatively-spliced exons and alternate initial or terminal exons. Initial and terminal exons are determined by a cell's transcription process rather than its splicing process, so we do not count them as ES events. Example III in Figure B.1 illustrates this distinction. The grey exons e and f in this graph represent alternative transcription start sites; either one will be selected during transcription rather than during splicing. Exon g is the only one removed during the splicing process, so it is the only exon counted as an ES event.

Appendix C

File formats

SpliceGrapher uses different file formats to import data of different types. Below we describe in some detail how these files are imported and converted into splice graphs or evidence for predicting splice graphs.

C.1 Gene Models

Gene models typically are stored as General Feature Format (GFF3) files or as Gene Transfer Format (GTF) files. Both formats use tab-delimited files with eight or more columns. While these are both documented standards, the standards are not always followed. For example, the second column in the GTF format is allocated for information about the source of the annotations, such as the name of the software package used. However, many of the ENSEMBL GTF files use this column to describe the kind of gene being annotated (e.g., "protein coding"). Thus SpliceGrapher's interpretation of these files may incorporate some knowledge of where they originated. In this discussion we refer to exons that have explicit acceptor and donor splice sites as *bounded* and those with an undefined acceptor or donor splice site as *unbounded* (see [73, 179]).

C.1.1 GFF3 Format

The GFF3 format [49] is a nine-column format, where each line in the file constitutes a record that describes a genomic feature such as a chromosome, gene or exon. Each record has a type (third column) that denotes its genomic feature such as *chromosome*, *gene*, *exon* or *CDS* (protein coding sequence), as well as positions (fourth and fifth columns) that specify its genomic location (Figure C.1). Usually the information in a file is organized in a hierarchical order: a *chromosome* record precedes any *gene* records associated with the chromosome, a *gene* record precedes any *exons* records associated with the gene, and so on. However, this may not not always be the case, so SpliceGrapher loads the information without regard for order and attempts to validate gene models once they are loaded. Another issue is the fact that not all GFF3 files use the same record types. For example, untranslated regions (UTRs) may appear as *UTR* records in one file and as *five_prime_UTR* or *three_prime_UTR* in another.

When SpliceGrapher accepts gene model annotations in GFF3 format it constructs graphs using the

Chr1	TAIR10	chromosome	1	30427671	.	.	.	ID=Chr1;Name=Chr1
Chr1	TAIR10	gene	3631	5899	.	+	.	ID=AT1G01010;Note=protein_coding_gene
Chr1	TAIR10	mRNA	3631	5899	.	+	.	ID=AT1G01010.1;Parent=AT1G01010
Chr1	TAIR10	protein	3760	5630	.	+	.	ID=AT1G01010.1-Protein;Name=AT1G01010.1
Chr1	TAIR10	exon	3631	3913	.	+	.	Parent=AT1G01010.1
Chr1	TAIR10	five_prime_UTR	3631	3759	.	+	.	Parent=AT1G01010.1
Chr1	TAIR10	CDS	3760	3913	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein
Chr1	TAIR10	exon	3996	4276	.	+	.	Parent=AT1G01010.1
Chr1	TAIR10	CDS	3996	4276	.	+	2	Parent=AT1G01010.1,AT1G01010.1-Protein
Chr1	TAIR10	exon	4486	4605	.	+	.	Parent=AT1G01010.1
Chr1	TAIR10	CDS	4486	4605	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein
Chr1	TAIR10	exon	4706	5095	.	+	.	Parent=AT1G01010.1
Chr1	TAIR10	CDS	4706	5095	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein
Chr1	TAIR10	exon	5174	5326	.	+	.	Parent=AT1G01010.1
Chr1	TAIR10	CDS	5174	5326	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein
Chr1	TAIR10	exon	5439	5899	.	+	.	Parent=AT1G01010.1
Chr1	TAIR10	CDS	5439	5630	.	+	0	Parent=AT1G01010.1,AT1G01010.1-Protein
Chr1	TAIR10	three_prime_UTR	5631	5899	.	+	.	Parent=AT1G01010.1

Figure C.1: Example of the GFF3 file format for a single gene from the *A. thaliana* TAIR10 annotations. Columns from left to right are: chromosome, annotation source, record type, start location, end location, alignment score, strand, reading frame and attributes. A period is a placeholder for missing values. Attributes may be specific to the record type, the application that generated the annotations, or the species. Some records have been truncated to fit the page.

sequence coordinates found in *UTR*, *CDS* and *exon* records. It interprets *exon* records as bounded exons and incorporates them directly into a splice graph. It infers an intron whenever the corresponding exons are adjacent in a transcript. When an exon appears in multiple transcripts, a single exon node is created along with edges that link it to the exons that follow or precede it in the corresponding transcripts.

In some cases gene models may yield a 0-length intron—for example, when the models describe coding sequence (CDS) and untranslated regions (UTR) in mature mRNA. In these cases an exon that contains a start or end codon will be annotated in the models as two exons—one associated with the UTR and one with the CDS—with a 0-length gap between them. SpliceGrapher merges these adjacent exons into a single exon and marks the starting location of the codon.

C.1.2 GTF Format

The GTF format uses the same columns as GFF3, but the attributes column use a different syntax. In addition, genetic information is stored not as a hierarchy, but as a series of exons. Hierarchical information such as the transcript or gene associated with an exon is included in the exon's attributes. If an exon takes part in more than one transcript, there will be multiple records for the exon. When SpliceGrapher loads a GTF file, it stores exon or CDS records on the fly, creating structures for chromosomes, genes and transcripts as it encounters them.

1	protein_coding	exon	3631	3913	.	+	.	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	CDS	3760	3913	.	+	0	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	start_codon	3760	3762	.	+	0	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	exon	3996	4276	.	+	.	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	CDS	3996	4276	.	+	2	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	exon	4486	4605	.	+	.	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	CDS	4486	4605	.	+	0	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	exon	4706	5095	.	+	.	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	CDS	4706	5095	.	+	0	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	exon	5174	5326	.	+	.	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	CDS	5174	5326	.	+	0	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	exon	5439	5899	.	+	.	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	CDS	5439	5627	.	+	0	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”
1	protein_coding	stop_codon	5628	5630	.	+	0	gene_id ”AT1G01010”; transcript_id ”AT1G01010.1”

Figure C.2: Example of the GTF file format for a gene from the *A. thaliana* annotations. The columns are nearly identical to the GFF format, but the second column may have gene type information instead of the annotation source. Note too the difference in the way attributes are listed in the last column. Records have been truncated to fit the page.

C.2 EST Alignments

Alignment tools such as BLAT [92] and GMAP [238] can provide alignment information in the Pattern Space Layout (PSL) format. The PSL format is a 21-column tab-delimited format that contains a wealth of information about a gapped EST alignment. Each line in the file is a record that includes information such as the start and end points for each aligned segment (called *blocks*), the number of matching and mismatching bases, and the number of insertions and deletions. A full description of the format is provided on the ENSEMBL website [52].

When SpliceGrapher loads a PSL file, it converts each record into a sequence of exons, but exons at the 3’ or 5’ end of an EST are considered unbounded, as the ESTs may originate from anywhere in a transcript. Unbounded exons are merged with other exons in the graph. An exon that is unbounded at its 3’ end is merged with another exon if they share the same acceptor site and both exons are unbounded at the 3’ end. Alternately, if one of the two exons is bounded, SpliceGrapher may merge them if the bounded exon contains the unbounded exon. Analogous rules apply to exons unbounded at the 5’ end. SpliceGrapher infers an intron between exons whenever they are adjacent in an EST.

C.3 RNA-Seq Alignments

Most alignment tools use the Sequence Alignment/Map (SAM) format (Figure C.5) or its cousin the Binary Alignment/Map (BAM) format to record alignment information. These formats have detailed

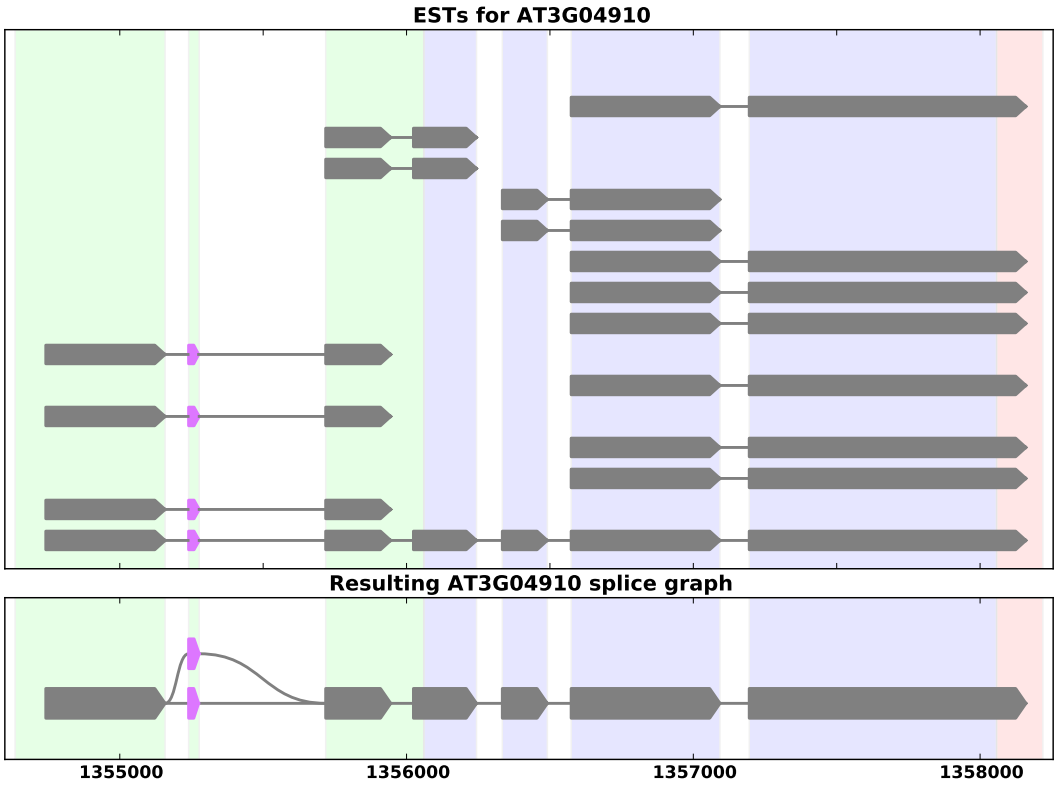

Figure C.3: Example showing how ESTs are converted to a splice graph. Exons at either end of an EST are considered unbounded at their extreme ends (the 5' exon is unbounded at its 5' end and the 3' exon is unbounded at its 3' end). These are merged with exons from other ESTs when their bounded ends match. The resulting splice graph is often far less complex than the collection of ESTs.

58	0	0	0	0	0	0	0	+	E5.1	59	1	59	3	230	66	24	1	58,	1,	66,
65	1	0	0	0	0	0	0	+	E3.1	84	0	66	3	230	70	36	1	66,	0,	70,
90	0	0	0	0	0	0	0	+	E9.1	90	0	90	3	230	28	18	1	90,	0,	28,
249	4	0	0	0	0	0	0	+	A8.1	253	0	253	3	230	26	79	1	253,	0,	26,
107	0	0	0	0	0	1	82	+	E5.1	107	0	107	3	230	62	51	2	29,78,	0,29,	62,73,
94	0	0	0	0	0	1	1	+	E9.1	94	0	94	3	230	98	93	2	25,69,	0,25,	98,24,

Figure C.4: Example of the PSL file format with several EST alignments.

specifications, but these may not be followed in the same way by all alignment tools. For example, in the tuxedo suite (Bowtie/Tophat/Cufflinks) Tophat [216] includes information in its alignments that Cufflinks requires to make its predictions. Thus Cufflinks may not make good predictions with alignments from another tool.

C.3.1 SAM Format

```
FC:296  73   chr1  13780  255  20M  *  0  0  AGCCTGGGTGAGGAAGCTGT  VVVVVVVVVVVVVVVVRQQQQ  NM:i:2  NH:i:1
FC:14   137  chr1  13024  255  20M  *  0  0  CGGCCCGGATGGGACAGGAC  VVVVVVVVVVVVVVVVQQQQQ  NM:i:1  NH:i:1
FC:34   137  chr1  13025  255  20M  *  0  0  GACCCGGATGGGACAGGAGT  VVVVVVVVVVVVVVVVQQQQQ  NM:i:1  NH:i:1
FC:8    137  chr1  13029  255  20M  *  0  0  CGGAGATGCCAGGAGTTGGG  QQQQQVVVVIVVVVVVVVVV  NM:i:1  NH:i:1
```

Figure C.5: Example of the SAM file format for alignments in *H. sapiens*. Records have been truncated to fit the page. Columns are: read id, set of bit-encoded flags, chromosome, alignment start position, alignment score, CIGAR string, mate pair identifier, mate pair alignment position, inferred insert size, read sequence, read quality and optional tags.

A record in the standard SAM format uses a minimum of 11 tab-delimited columns containing the alignment information for a single read (Figure C.5). From left to right, these are the unique read identifier; a bit-encoded set of flags describing alignment characteristics such as strand; the name of the location where it aligned (usually chromosome); the alignment start position; alignment score; a Concise Idiosyncratic Gapped Alignment Report (CIGAR) string (described below); unique identifier for the read's mate (paired-end reads); the mate's alignment position; the insert size inferred for two mates; the actual read sequence; the base-by-base read quality, and optional coded tags. When SpliceGrapher loads SAM data, it tracks the location and size of each read in the file. For ungapped reads it is straightforward, but spliced reads include elaborate CIGAR strings that describe the matches and gaps in the alignment.

C.3.2 CIGAR Strings

CIGAR string	Interpretation
76M	76-nt match
37M2600N38M	37-nt match / 2,600-nt gap / 38-nt match
11M780N58M1911N9M	11-nt match / 780-nt gap / 58-nt match / 1,911-nt gap / 9-nt match

Figure C.6: Some examples of typical CIGAR strings taken from a set of SAM alignments for *H. sapiens*. The first example shows a typical ungapped alignment; the second example shows a typical spliced alignment, and the last example is a read that crosses two junctions.

The CIGAR string is the primary source of gapped alignment information for a read. The format uses letter codes to convey matches (M), gaps (N), insertions (I), deletions (D) and other information about a spliced alignment. A string is simply a concatenation of sizes and codes; for example, "35M" means a

35-character match between a read and a reference sequence. Some typical examples are given in Figure C.6. The first example shows a typical ungapped alignment where 76 characters from the read matched the reference sequence. The second example shows a typical spliced alignment with a single intron gap and anchors of 37nt and 38nt on either side. The last example is from a read that crosses two junctions: a 780-nt intron followed by a 1,911-nt intron.

C.4 Splice Graphs

```
4  SG  graph   7252  7384  .  -  .  ID=G180;Name=G180;Note=protein_coding_gene
4  SG  parent  7497  7584  .  -  .  ID=G180_7;AltForm=IR;Isoforms=G180.2;disposition=known
4  SG  parent  7476  7584  .  -  .  ID=G180_8;Isoforms=G180.1;disposition=known
4  SG  parent  7468  7487  .  -  .  ID=pred_123;putative_children=G180_4;disposition=unresolved
4  SG  parent  7468  7534  .  -  .  ID=pred_121;putative_children=G180_4;disposition=unresolved
4  SG  child   7290  7307  .  -  .  ID=G180_2;Parent=G180_3;Isoforms=G180.1,G180.2;disposition=known
4  SG  child   7369  7377  .  -  .  ID=G180_4;Parent=G180_5;AltForm=A3;Isoforms=G180.1,G180.2;disposition=known
4  SG  child   7467  7487  .  -  .  ID=G180_6;Parent=G180_8;Isoforms=G180.1;disposition=known
4  SG  child   7252  7288  .  -  .  ID=G180_1;Parent=G180_2;Isoforms=G180.1,G180.2;disposition=known
4  SG  child   7369  7370  .  -  .  ID=pred_118;Parent=G180_5;AltForm=A3;disposition=predicted
4  SG  child   7428  7447  .  -  .  ID=G180_5;Parent=G180_7,G180_6;Isoforms=G180.1,G180.2;disposition=known
4  SG  child   7348  7353  .  -  .  ID=G180_3;Parent=G180_4,pred_118;Isoforms=G180.1,G180.2;disposition=known
```

Figure C.7: Example of SpliceGrapherXT GFF output format. The columns are the same as for GFF3 gene models, but the hierarchical structure is more compact. After the requisite `graph` record, every node (exon) is listed exactly once along with annotations with the identifiers for the isoforms in which it participates along with the AS events associated with the node. Child nodes include a `Parent` annotation that lists the parent nodes connected to the child.

When SpliceGrapherXT predicts a splice graph, it stores the graph in a file that uses the GFF format (Figure C.7). These files are similar to the gene model files described in section C.1.1, but with the following modifications. Every file begins with a `graph` record, similar to a gene, that records the graph's chromosome, start and ane locations, strand and name. Additional information may be copied from the gene models. The remaining records describe `parent` and `child` nodes in the graph. In addition to the usual chromosome, strand and position information, transcript and AS attributes are included in the attribute list in the last column. These include the isoforms in which a node participates and the AS events associated with it. Parent-child relationships are annotated for child nodes using the *Parent* attribute in the last column. Finally, every node has a `disposition` that tells whether the associated exon originated from the gene models (`known`), was predicted by SpliceGrapherXT or is still unresolved.

Appendix D

Splice Site Classifier Performance for 107 Species

We downloaded the gene models and reference genome sequences for 107 species from the ENSEMBL website and used SpliceGrapherXT to create classifiers for GT donor sites and AG acceptor sites for each species (Table D.1). These classifiers are all highly accurate, with ROC scores ranging from .90 to .99. In general GT donor sites are easier to classify as indicated by an average ROC score of .97 and lower variance across species (Figure D.1, right). For AG acceptor sites the average score is .95 and the distribution across the species varies considerably more.

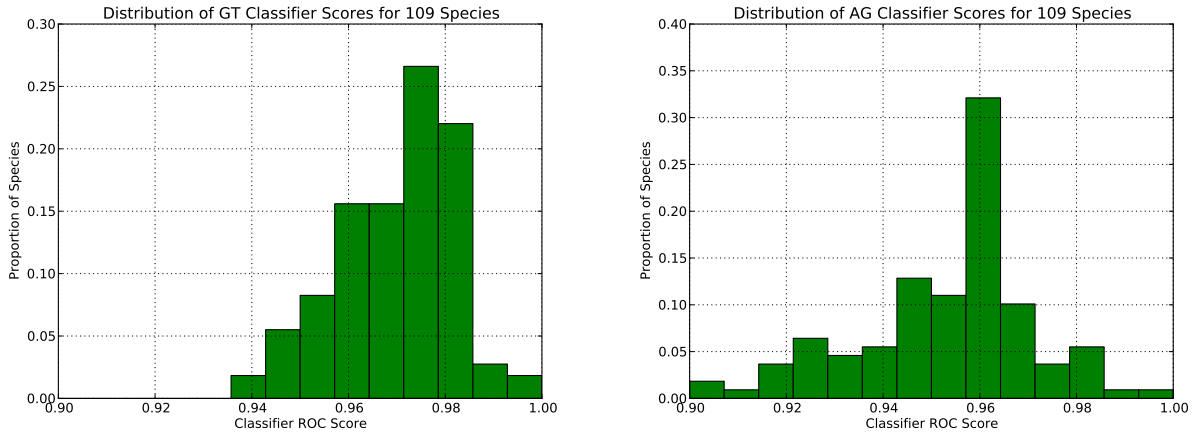

Figure D.1: Distributions of classifier scores across 107 species. Both GT and AG classifiers average ROC scores better than 95%, demonstrating that SpliceGrapher's splice-site classifiers can be used to filter alignments in nearly any species that has enough gene model annotations to generate a training set.

Table D.1: SpliceGrapher classifier performance for 107 different species. Classifiers were built using SpliceGrapherXT pipeline and ROC scores computed via 5-fold cross-validation.

Species	GT	AG	Species	GT	AG	Species	GT	AG
A. pisum	0.96	0.94	*D. mojavensis*	0.97	0.96	*O. princeps*	0.98	0.96
A. aegypti	0.96	0.95	*D. persimilis*	0.96	0.96	*O. anatinus*	0.96	0.94
A. queenslandica	0.99	0.99	*D. pseudoobs.*	0.98	0.96	*O. cuniculus*	0.97	0.96
A. carolinensis	0.98	0.96	*D. sechellia*	0.97	0.96	*O. sativa*	0.95	0.94
A. gambiae	0.96	0.95	*D. simulans*	0.97	0.96	*O. latipes*	0.96	0.95
A. mellifera	0.98	0.96	*D. virilis*	0.98	0.97	*O. garnettii*	0.97	0.96
A. thaliana	0.97	0.95	*D. willistoni*	0.97	0.96	*P. troglodytes*	0.98	0.96
A. gossypii	0.99	0.96	*D. yakuba*	0.98	0.97	*P. marinus*	0.95	0.93
A. clavatus	0.98	0.93	*E. telfairi*	0.98	0.96	*P. patens*	0.98	0.95
A. flavus	0.96	0.92	*E. histolytica*	0.99	0.96	*P. berghei*	0.94	0.99
A. fumigatus	0.97	0.90	*E. caballus*	0.95	0.95	*P. chabaudi*	0.96	0.98
A. nidulans	0.96	0.91	*E. europaeus*	0.98	0.96	*P. falciparum*	0.96	0.98
A. terreus	0.97	0.93	*F. catus*	0.98	0.96	*P. pacificus*	0.94	0.98
B. mori	0.98	0.98	*F. oxysporum*	0.96	0.92	*P. capensis*	0.98	0.96
B. taurus	0.98	0.96	*G. morhua*	0.98	0.96	*P. vampyrus*	0.98	0.97
B. fuckeliana	0.98	0.95	*G. gallus*	0.97	0.95	*P. ultimum*	0.95	0.95
B. distachyon	0.98	0.96	*G. aculeatus*	0.96	0.95	*S. cerevisiae*	0.99	0.92
C. brenneri	0.97	0.98	*G. moniliformis*	0.97	0.93	*S. harrisii*	0.97	0.95
C. briggsae	0.95	0.97	*G. zeae*	0.98	0.94	*S. mansoni*	0.96	0.94
C. elegans	0.97	0.98	*G. gorilla*	0.97	0.95	*S. pombe*	0.99	0.95
C. remanei	0.97	0.98	*H. melpomene*	0.96	0.96	*S. sclerotiorum*	0.95	0.90
C. familiaris	0.98	0.95	*H. sapiens*	0.98	0.96	*S. araneus*	0.98	0.96
C. porcellus	0.97	0.96	*I. scapularis*	0.94	0.93	*S. bicolor*	0.97	0.96
C. hoffmanni	0.98	0.96	*L. chalumnae*	0.97	0.94	*S. scrofa*	0.97	0.96
C. intestinalis	0.98	0.96	*L. africana*	0.96	0.95	*T. rubripes*	0.96	0.92
C. quinquefas.	0.96	0.97	*M. mulatta*	0.95	0.93	*T. syrichta*	0.99	0.97
D. plexippus	0.97	0.96	*M. eugenii*	0.97	0.96	*T. nigroviridis*	0.96	0.95
D. rerio	0.98	0.96	*M. oryzae*	0.97	0.93	*T. gondii*	0.98	0.98
D. pulex	0.98	0.95	*M. gallopavo*	0.97	0.94	*T. castaneum*	0.96	0.96
D. novemcinctus	0.97	0.96	*M. murinus*	0.97	0.96	*T. adhaerens*	0.96	0.93
D. discoideum	0.99	0.99	*M. musculus*	0.98	0.96	*T. belangeri*	0.97	0.96
D. ordii	0.98	0.97	*M. lucifugus*	0.97	0.97	*T. truncatus*	0.98	0.97
D. ananassae	0.98	0.96	*N. vectensis*	0.97	0.96	*V. pacos*	0.98	0.96
D. erecta	0.98	0.97	*N. fischeri*	0.98	0.93	*X. tropicalis*	0.96	0.94
D. grimshawi	0.98	0.96	*N. crassa*	0.96	0.94	*Z. mays*	0.96	0.94
D. melanogaster	0.98	0.96	*N. leucogenys*	0.96	0.94			

Appendix E

The SpliceGrapherXT Software

E.1 Overview

SpliceGrapher predicts alternative splicing patterns and produces splice graphs that capture in a single structure the ways a gene's exons may be assembled. It enhances gene model annotations using evidence from next-generation sequencing and EST alignments. With *SpliceGrapherXT* (version 0.2.2), we introduce the ability to convert splice graph predictions into transcript predictions using IsoLasso.

E.1.1 Downloading and Installation

- You may download *SpliceGrapher* from `http://SpliceGrapher.sourceforge.net`
- Requires PyML version 0.7.9 or higher for classifying splice sites.
- Requires `matplotlib` version 1.1.0 or higher for the extensive graphics tools.
- The optional IsoLasso pipeline requires `IsoLasso` version 2.6.1 or higher, along with the 'gtfToGenePred' and 'genePredToBed' programs from UCSC (`http://hgdownload.cse.ucsc.edu/admin/exe`).

Currently Unix/Linux and Mac OS-X are supported. A *setup.py* script is provided so installation is standard for python:

```
python setup.py build
python setup.py install
```

To check that SpliceGrapher is installed correctly, run the python interpreter and type

```
>> import SpliceGrapher
>> SpliceGrapher.__version__
'0.2.2'
```

To test your installation more thoroughly, use the test script in the *SpliceGrapher* `examples` sub-directory:

```
cd examples
run_tests.sh
```

E.1.2 Brief Tutorial

Before getting into SpliceGrapher's nuts and bolts, we begin with a few examples to demonstrate its key features. In the directory where you installed SpliceGrapher you will find a sub-directory called `tutorial` that contains all the data you will need for the following examples. Be sure that SpliceGrapher's scripts are in your path.

Note that all of the examples use the *SpliceGrapher.cfg* file in the `tutorial` directory. This contains the paths to the default reference genome (*a_thaliana.fa.gz*) and the default gene models (*a_thaliana.gff3.gz*).

E.1.2.1 Example 1: Create Classifiers

In this example you will create accurate models for canonical (GT-AG) and semi-canonical (GC-AG) splice sites for the plant *Arabidopsis thaliana*. To reduce classifier training time you may also want to set the number of examples to something relatively small such as `-n 400` (the default is 2000). Create the classifiers by running the *build_classifiers.py* script as follows:

```
build_classifiers.py -d gt,gc -n 400
```

The program shows you the SpliceGrapher scripts it uses for each step in the process. Below is output from a typical run:

```
Creating SVM models for splice sites
  09:45:10 generate_splice_site_data.py -d gt -n 200 -o gt_don_training.fa
  -r splice_site_report.txt
  09:45:20 generate_splice_site_data.py -d gt -n 200 -o gt_don_neg.fa -N
  09:45:29 cat gt_don_neg.fa >> gt_don_training.fa
  09:45:29 rm gt_don_neg.fa
  09:45:51 generate_splice_site_data.py -d gc -n 200 -o gc_don_training.fa
  09:46:01 generate_splice_site_data.py -d gc -n 200 -o gc_don_neg.fa -N
  09:46:10 cat gc_don_neg.fa >> gc_don_training.fa
  09:46:10 rm gc_don_neg.fa
  09:46:31 generate_splice_site_data.py -a -d ag -n 200 -o ag_acc_training.fa
  09:46:40 generate_splice_site_data.py -a -d ag -n 200 -o ag_acc_neg.fa -N
  09:46:49 cat ag_acc_neg.fa >> ag_acc_training.fa
  09:46:49 rm ag_acc_neg.fa
  09:47:12 zip classifiers.zip ??_???.cfg ??_???.svm ??_???.fa
  09:47:12 rm ??_tmp_e*_i*.fa
Finished.
```

When the script finishes, you may check classifier performance by looking for the ROC scores in their corresponding `.cfg` files:

```
grep roc *.cfg
```

Scores typically will be above 0.90:

```
ag_acc.cfg:roc_score = 0.935625
gc_don.cfg:roc_score = 0.94925
gt_don.cfg:roc_score = 0.969875
```

The script packages these classifiers into the `classifiers.zip` file in the local directory.

E.1.2.2 Example 2: Filter Alignments

This example you may use the classifiers you created in Example 1, or you may simply copy the `Arabidopsis_thaliana.zip` file from `classifiers` sub-directory of your SpliceGrapher distribution. This example uses the following files:

- Alignment output (SAM format): `alignments.sam.gz`
- *A. thaliana* classifiers (ZIP format): `classifiers.zip` (from Example 1 above)
  or `Arabidopsis_thaliana.zip`

To filter the alignment output, run the *sam_filter.py* script:

```
sam_filter.py alignments.sam.gz classifiers.zip -o filtered.sam
```

E.1.2.3 Example 3: Predict a Splice Graph

This example demonstrates SpliceGrapher's prediction modules. In this case we will use it to predict a graph for the gene **AT2G04700** that we selected as an illustrative example. This example uses the following files:

- Alignment output (SAM format): `filtered.sam`
- Plotter configuration file: `AT2G04700_plot.cfg`

To predict a graph for this gene, run the *predict_splicegraph.py* script:

```
predict_splicegraph.py AT2G04700 -d filtered.sam -o AT2G04700.gff
```

There are two ways to view the resulting graph: *view_splicegraphs.py* and *plotter.py*. The first script allows you to view a splice graph on your screen or, if you prefer, save it to a file. *plotter.py* is designed for file output and provides a more extensive set of options. We have provided an example plotter configuration file to get you started:

```
view_splicegraphs.py AT2G04700.gff -L
plotter.py AT2G04700_plot.cfg
```

E.2 The *SpliceGrapher* Environment

Throughout the this guide we describe a variety of tasks that require different input files. By far the most common files are a reference sequence (a FASTA file) and a gene model (either GFF3 or GTF format).

E.2.1 Environment Variables

You may specify default paths for these files by setting the environment variables `SG_FASTA_REF` and `SG_GENE_MODEL`. For example, in C shell this might look like:

```
setenv SG_FASTA_REF "/home/o_sativa/sequences/o_sativa.fasta"
setenv SG_GENE_MODEL "/home/o_sativa/annotations/o_sativa.gff3"
```

SpliceGrapher will use these default paths when they are not provided to a script on the command line or in a configuration file.

E.2.2 SpliceGrapher Configuration File

SpliceGrapher programs and modules also use the configuration file `SpliceGrapher.cfg` to locate the genomic reference and gene model files. Below is an example of a configuration file.

```
[SpliceGrapher]
GENE_MODEL = /home/o_sativa/annotations/o_sativa.gff3
FASTA_REFERENCE = /home/o_sativa/sequences/o_sativa.fasta
```

SpliceGrapher scripts look for `SpliceGrapher.cfg` in your `PATH`. When working with data for different organisms, it is often convenient to place a `SpliceGrapher.cfg` file for each organism in its own

directory. Then one can change *SpliceGrapher*'s defaults simply by changing directories. *SpliceGrapher* scripts will first use any paths specified on the command line, followed by those found in a configuration file and then those given as environment variables.

E.3 Splice Site Classifiers

For RNA-Seq data we are concerned with two kinds of alignments: *ungapped* alignments and *splice junction* alignments. An ungapped alignment is one in which a read aligns completely within an exonic region in the reference sequence. Most short-read alignment algorithms can perform exonic alignments. A splice junction alignment is one in which part of a read aligns to the 3' end of one exon and the remainder aligns to the 5' end of another exon, usually within the same gene. However, many spliced alignment programs use only rudimentary heuristics to identify correct splice sites. *SpliceGrapher's* approach is to filter these junctions using highly accurate splice-site classifiers.

Starting with version 0.2.0, *SpliceGrapher* comes with over 100 pre-built classifiers for a variety of different species. These are provided as `.zip` files located in the `classifiers` directory. You may also build your own classifiers using the *build_classifiers.py* script.

E.3.1 Building Classifiers

To simplify the process of constructing classifiers and to provide an example script you may use to create your own pipeline, *SpliceGrapher* includes the script *build_classifiers.py*. Assuming you have established paths for a FASTA reference sequence a gene model file `SpliceGrapher.cfg` file, you can generate a complete set of classifiers for filtering spliced alignments with one command:

```
build_classifiers.py
```

The script performs the following steps:

1. Generates training data for each splice site dimer.
2. Selects optimal parameters for each classifier.
3. Stores the data for each classifier in a `.zip` file

The script accepts the following options:

Option	Value	Description
–commands		Show but don't run commands
-a	dimer list	Acceptor site dimers to predict
-d	dimer list	Donor site dimers to predict
-f	file path	Fasta reference file
-m	file path	GFF3 gene model annotation file
-n	examples	Total number of examples to use for training
-l	file path	Optional output log file

By default the script builds classifiers only for canonical GT donor sites and AG acceptor sites. To build a classifier for semi-canonical GC donor sites as well, simply provide a list to the script:

```
build_classifiers.py -d gt,gc
```

In each step of this pipeline, the main script calls other *SpliceGrapher* scripts to generate data and train classifiers. By using the `--commands` option, you can see the commands that would be generated without actually running them. This is instructive when learning how to use each of the scripts individually. Each of the scripts used in *build_classifiers.py* is described in the following sections. (For more information on how these classifiers are constructed, please see [14].)

For the splice site prediction modules *SpliceGrapher* uses support vector machines (SVM) implemented in the `http://pyml.sourceforge.net`PyML package. To create a set of splice-site classifiers, we suggest taking the following steps:

1. Identify most common splice-site dimers for the species
2. Generate positive, negative training data sets
3. Find optimal classifier parameters for each splice site

Each of these procedures is described in the following sections.

E.3.1.1 Identifying Splice-Site Dimers

The workhorse script *generate_splice_site_data.py* can identify frequently-occurring splice-site dimers and generate positive and negative example sequences for any given dimer. The script can produce a report of all splice site dimers and splice junctions found in a set of gene models. The most frequently occurring splice site dimers are then candidates for constructing classifiers for predicting novel splice sites in a genome. The command is simply:

```
generate_splice_site_data.py -r
```

Below is an example of a report, truncated for readability:

```
Breakdown of donor sites for 122535 introns:
 GT:  121165 (98.88%)
 GC:    1248 ( 1.02%)
 AT:      84 ( 0.07%)
 GA:      13 ( 0.01%)
 TT:       8 ( 0.01%)
Breakdown of acceptor sites for 122535 introns:
 AG:  122419 (99.91%)
 AC:      76 ( 0.06%)
 AT:       9 ( 0.01%)
Breakdown of splice junctions:
 GT-AG:  121132 (98.8550%)
 GC-AG:    1248 ( 1.0185%)
 AT-AC:      73 ( 0.0596%)
 GA-AG:      13 ( 0.0106%)
 TT-AG:       8 ( 0.0065%)
```

Given a list of splice-site dimers and their frequencies, one can then choose an optimal set that balances the potential number of successful predictions with the overhead of training a classifier for each dimer.

E.3.1.2 Generating Training Data

The *generate_splice_site_data.py* script is also used to generate positive and negative examples for training SVMs. Positive examples are simply examples of the given dimer found at splice sites in the gene model. To generate negative examples *SpliceGrapher* uses the procedure outlined in [164]: it looks for all examples of the given dimer within each gene and uses as negative examples those that don't appear in splice sites.

To generate positive examples for a given donor-site dimer such as `GT`, use the following command:

```
generate_splice_site_data.py -o gt_positive.fa -d GT
```

To generate examples for acceptor sites, use the `-a` option:

```
generate_splice_site_data.py -o ag_positive.fa -d AG -a
```

As with other scripts, the default gene model and FASTA reference are provided in `SpliceGrapher.cfg` variables, and output is written to `stdout` by default.

To generate negative examples, simply add the `-N` option to the command:

```
generate_splice_site_data.py -o gt_negative.fa -d GT -N
```

By default, output splice site sequences will include 100nt on either side of a splice-site dimer, but will not include the dimer itself. To include splice-site dimers in output sequences, use the `-D` option. To change the window size (for example, 30nt on either side), use the `-W` option:

```
generate_splice_site_data.py -o gt_positive.fa -d GT -W 30
```

Note that the window size will reduce the number of possible SVM parameter combinations but may also reduce SVM performance if the windows are too small. Positive examples for any dimer tend to be relatively rare, while negative examples are plentiful. In fact, there may orders of magnitude more negative examples than positive, which would be overkill for training a good classifier. To set a limit on the number of negative examples generated, use the `-n` option. When a limit is set, the script will perform the following random selection procedure a fixed number of times.

First it selects a gene at random from the genome. Once a gene has been selected, it samples positions at random from within the gene until it finds the appropriate dimer at a position other than a known splice site. Once the appropriate number of negative examples have been randomly selected, the script halts. Thus to generate a sample of 10,000 negative `GT` examples, use the command:

```
generate_splice_site_data.py -o gt_negative.fa -d GT -N -n 10000
```

E.3.1.3 Optimizing Classifiers

SpliceGrapher uses support-vector machines (SVMs) to classify splice sites. More specifically it uses SVMs with a weighted-degree kernel that has proved especially successful at discriminating real splice sites from spurious ones. While these models are well-suited for predicting novel splice sites, there are a number of parameters that that can influence SVM performance for a specific kind of splice site. Briefly, these are:

- intronic and exonic sequence length on either side of a site
- range of k-mer sizes to use
- size of shifts to allow when assessing k-mers
- number of mismatches to allow
- SVM regularization parameter, **C**

For an overview and tutorial on SVMs with weighted-degree kernels, see [14].

Option	Value	Description
-a		Interpret splice sites as acceptors
-e	EXONSIZE	List of exon sizes to try (e.g., '20,25,30')
-C	CLIST	List of regularization constants for SVM (e.g., '0.1,1.0,10.0')
-i	INTRONSIZE	List of intron sizes to try (e.g., '20,25,30')
-l	LOGFILE	Optional log file for tracking performance
-k	KFOLDS	Number of folds to run cross-validation
-m	MINK	Set of minimum k values for kernel (e.g, '1,2,3')
-M	MAXK	Set of maximum k values for kernel (e.g., '3,4,5')
-N		Turn normalization OFF
-p		Apply a mismatch profile
-P	PREFIX	Optional prefix for output files
-S	SHIFT	List of maximum shift values

To simplify this process, *SpliceGrapher* includes the script *select_model_parameters.py*. This script trains a weighted-degree kernel SVM on labeled FASTA sequences generated by *generate_splice_site_data.py* and runs cross-validation for a variety of parameter settings. As the program proceeds, it saves the best model and parameter configuration. The basic format of this command is

```
select_model_parameters.py FASTA-training-data dimer
```

For example:

```
select_model_parameters.py gt_training.fa gt
```

This runs cross-validation on a model with just the default parameters (exon lengths of 8nt, 12nt and 16nt; intron lengths of 15nt, 20nt and 25nt; minimum k-mer size 1; maximum k-mer 1, 2 or 3; no mismatches; shift sizes 0 and 1, and C = $\{0.01,0.1,1,10,100\}$). By default, cosine normalization is applied to kernel vectors, but this may be turned off using the `-N` flag.

The script accepts a list of values to try for every SVM parameter. For example, to try several shift values, one might use `'-S 0,1,2,3'`. Note, however, that the number of parameter combinations grows geometrically with each set of values. Since it can take a long time to run cross-validation on an elaborate model, the program accepts a maximum of 100 parameter combinations per run.

E.3.1.4 Intron and Exon Sequence Lengths

When *generate_splice_site_data.py* generates training data, it uses the same window size on either side of a splice site. This permits the model selection program to try different combinations of intron and exon sequence lengths up to this window size.

Note: in theory, one could use windows that are hundreds or thousands of nucleotides long. But as window sizes increase, the time required for cross-validation and for classification can increase dramatically. Experience has shown that classifier performance often does not improve significantly beyond a range of 30-40nt on either side of a splice site. The parameter selection script imposes a hard limit of 50nt on either side of a junction. If you wish to experiment with longer sequences you will need to change the hard-coded limit.

To identify the best intron and exon sequence lengths, run the model selection algorithm with a series of possible lengths for each. For example, if one wishes to find the best model performance for a donor `GT` splice site with exon sequence sizes from 10 to 30 and intron sequence sizes from 10 to 40, the command would then be:

```
select_model_parameters.py gt_training.fa gt -e 10,20,30 -i 10,20,30,40
```

To experiment with parameters, simply include in the same command values for those parameters you wish to try:

```
select_model_parameters.py gt_training.fa gt -e 20,30 -i 30,40 -m 1,2 -M 2,3
```

Note: despite one's best efforts, some parameter settings may cause an SVM to fail to converge. Often this may be corrected by adding training data or by using different parameter values. For this reason we recommend working with only a few parameters at a time, taking care to save the resulting output files.

E.3.2 Classifier Data

SpliceGrapher creates three files for each classifier: a `.svm` file that contains the trained classifier parameters; a `.fa` file that contains the sequences used for training, and a `.cfg` file that contains meta-data

about the classifier including the parameters optimized by *select_model_parameters.py* and its ROC score. To avoid having to track three files for every classifier, *SpliceGrapher* usually stores these files in a single convenient `.zip` archive.

E.3.3 Generating ROC Curves

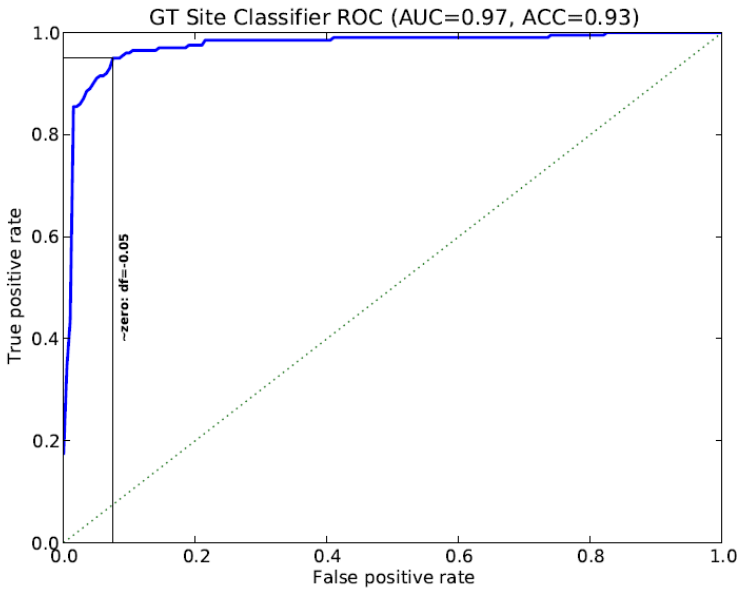

Figure E.1: Example of an ROC curve generated by *generate_roc.py* for a GT classifier.

SpliceGrapher includes a script for generating ROC curves for classifiers. The basic command syntax is *generate_roc.py config-file*. This script reads the files generated by the parameter selection program, performs 5-fold cross- validation, and generates an ROC curve. For example:

```
generate_roc.py gt_don.cfg -o gt_roc.pdf
```

Example output is shown in the figure. It may not always be possible to train a classifier with strong ROC scores. In these cases it may be useful to set a decision function threshold for classifying splice sites. The *generate_roc.py* script can aid in this process. Using the `-D` option, one can see the True Positive rate associated with different decision function values. One can then set an appropriate threshold value in the classifier configuration (`.cfg`) file for classifying splice sites.

E.4 Predicting Splice Graphs

Splice graph prediction is *SpliceGrapher*'s main purpose. This process involves two key aspects: validating the available evidence and making confident predictions from all available evidence.

E.4.1 Filtering Spliced Alignments

A key aspect of splice graph prediction is ensuring the quality of the evidence being used. To this end, *SpliceGrapher* includes scripts for filtering spliced alignments in the SAM format. The first is *sam_filter.py* (which replaces *filter_sam_jct.py* in older versions). It takes as input a SAM file and a set of classifiers and produces a copy of the SAM file with false-positive sites removed:

```
sam_filter.py alignments.sam classifiers.zip -o filtered.sam
```

The script accepts the following options:

Option	Value	Description
-F	SAM file	File for storing false-positive records
-o	OUTPUT	Output file
-r	REPORT	Write classifier scores to file
-z		Apply gzip compression to output

By default, SAM records that include false-positive sites will be discarded. You may save these records to a separate file using the `-F` option. The `-r` option allows you to see the classifier scores for each splice site found in the input file: negative values show which sites were predicted as false-positives.

E.4.1.1 Collating SAM Files

For large volumes of RNA-Seq data, a single SAM file may be unwieldy. For this reason *SpliceGrapher* includes the script *sam_collate.py* that reads a SAM alignment file and writes the output to separate files, one for each chromosome.

```
sam_collate.py *.sam
```

Because SAM files can be large, the script can read gzipped files and with the `-z` option, it can write them as well.

Note: Most SpliceGrapher scripts expect SAM files to be sorted by chromosome and position within the chromosome. A command for sorting a SAM file is:

```
sort -k 3,3 -k 4,4n unsorted.sam > sorted.sam
```

Alternately you may use samtools (samtools.sourceforge.net) for manipulating files prior to collating.

E.4.2 Creating Splice Graphs from Gene Models

Splice graphs are a key data structure for the *SpliceGrapher* package. They can be used to depict a gene model, a collection of EST alignments and of course, they provide the foundation for splice graph predictions. SpliceGrapher files are stored in a GFF format that is specific to these graphs.

The script *gene_model_to_splicegraph.py* accepts a GFF3 gene model annotation file as input and generates splice graphs as output. The output file may contain all the genes in the gene model file, or you may specify an output directory and write a separate file for each gene. For example, to generate splice graphs for all gene models in the `GENE_MODEL` file (from `SpliceGrapher.cfg`), the command would be:

```
gene_model_to_splicegraph.py -o gene_models.gff
```

To generate the same graphs into separate files in a directory `/home/mygraphs`, the command would be:

```
gene_model_to_splicegraph.py -d /home/mygraphs}
```

It is often useful to organize splice graphs by chromosome, since reference sequences are separated that way. For example, instead of having a single top- level directory, you may have a directory for each chromosome, such as `/home/mygraphs/chr1, /home/mygraphs/chr2, ....` In this case, simply use the `-c` option to specify the chromosome for which you want to generate graphs:

```
gene_model_to_splicegraph.py -d /home/mygraphs/chr3 -c chr3
```

Finally, you may use the script to generate splice graphs for a specific set of genes using the `-g` option. For example, if you were interested in generating graphs only for the Arabidopsis genes AT3G01100, AT3G01120, AT3G01460 and AT3G01490 into the local directory you would enter:

```
gene_model_to_splicegraph.py -d . -g AT3G01100,AT3G01120,AT3G01460,AT3G01490
```

Note: although GFF3 gene model annotations are becoming more standardized, not all organisms adhere to the same naming conventions. For example, UTR records may be given as 'three_prime_UTR' and

'five_prime_UTR', or simply 'UTR' in which case the relative gene position must be inferred. Thus you may need to modify the GFF3 annotations prior to converting them to splice graphs if you want access to all gene features. (See the GeneModel module for more information.)

E.4.2.1 Converting GTF Annotations

GTF file functionality has been enhanced in version 0.2.1, allowing users to work more easily with gene models in either GTF or GFF3 formats. We have replaced the *gtf2gff.py* script with *convert_models.py* that can convert gene models in GTF format to GFF3 format and vice versa. To convert from GTF to GFF3, the command is:

```
convert_models.py Homo_sapiens.gtf --gff3=Homo_sapiens.gff3
```

To convert from GFF3 to GTF, the command is:

```
convert_models.py Homo_sapiens.gff3 --gtf=Homo_sapiens.gtf
```

Note: for both GTF and GFF3 files, SpliceGrapher expects curated gene models from sites such as UCSC, ENSEMBL or the TAIR website. In many cases it can accept GTF files output from tools like Cufflinks, but this behavior is not guaranteed, as not all GTF or GFF3 files represent gene models. Also, some tools may not adhere strictly to the GTF or GFF3 standards. For more on these file formats, see `http://uswest.ensembl.org/info/website/upload/gff.html`.

E.4.3 Creating Splice Graphs from ESTs

Popular EST alignment algorithms such as BLAST, BLAT and GMAP can produce output in PSL format. Consequently *SpliceGrapher* includes a script, *ests_to_splicegraph.py* that converts EST alignments in PSL format to a splice graph. The script operates very much like the *gene_model_to_splicegraph.py* script outlined above except that you must specify a PSL file:

```
ests_to_splicegraph.py gmap_alignments.psl -d est_splicegraphs
```

As before, you may specify a single output file or an output directory, and you may request specific gene names.

Alignment programs sometimes yield introns that are too short (for example exons may abut one another) and that may lead to spurious alternative splicing predictions. To address this issue, the script allows

you to specify a minimum allowed intron length (default is 4nt). For example, if the smallest known intron for an organism is 10nt long, you might use:

```
ests_to_splicegraph.py gmap_alignments.psl -d est_splicegraphs -i 10
```

Alignments in the PSL file that infer introns shorter than this will not be included in the splice graphs.

E.4.4 predict_graphs.py

With version 0.2.1 SpliceGrapher allows you to predict the splice graphs for every gene in a file of gene models at once. The *predict_graphs.py* script provides a simple interface for loading RNA-Seq alignments along with gene models to generate predictions for all genes that have read coverage. The predicted graphs will be created in files stored in sub-directories named for each chromosome in the gene models.

The basic format of the command is:

```
predict_graphs.py SAM-file [options]
```

This program accepts the following parameters:

Note: *predict_graphs.py* loads a complete SAM file along with a complete set of gene models, so it can require a lot of memory (40GB or more for predictions across the human genome). We intend to address this memory issue in a future release. For now we recommend splitting the SAM file and gene models by chromosome and running *predict_graphs.py* on each chromosome separately.

E.4.5 predict_splicegraph.py

To predict splice graphs from diverse evidence sources, *SpliceGrapher* uses the script *predict_splicegraph.py*. Given annotations for a particular gene along with additional evidence such as EST alignments, ungapped short read alignments and splice junction alignments, it combines the information and uses inference rules to predict an output graph.

The program starts with a splice graph created from a gene model. To this you may add additional splice graphs (such as those generated from EST alignments) or short read alignment data. The output is a single splice graph that incorporates all of the data sources into a single predicted graph.

The basic format of the command is:

```
predict_splicegraph.py gene-model [options]
```

Option	Value	Description
-d	SAM file	short-read alignments
-J	JMINDEPTH	Minimum coverage required across splice junctions
-M	MINANCHOR	Minimum anchor size required for junction evidence
-s	GRAPHS	Comma-separated list of EST splice graph files
-T	THRESHOLD	Minimum depth threshold for accepting short-read clusters

This program accepts the following parameters:

Use the `-s` option to include additional splice graphs from EST alignments. For example, for the Human gene HES4, you might use:

```
predict_splicegraph.py HES4_model.gff -s HES4_ests.gff
```

To include RNA-Seq reads, specify a SAM alignment file using the `-d` option:

```
predict_splicegraph.py HES4_graph.gff -d chr1.sam.gz
```

The `-M` option provides flexibility beyond the anchor constraints placed on spliced alignments. Even if you specify a minimum anchor size for spliced alignment, the distribution of reads across a junction may not be uniform. This option tells the prediction program to use only junctions supported by at least one read with this many bases on each side:

```
predict_splicegraph.py HES4_graph.gff -d chr1.sam.gz -M 12
```

Finally, the `-T` option provides the prediction program with a threshold for treating short-read clusters as background noise. By default, *SpliceGrapher* accepts all short-read clusters as possible evidence.

E.4.5.1 Finding Splice Forms

SpliceGrapher also has the ability to identify annotated splice forms that are represented in a set of RNA-Seq data. It uses existing gene models to establish a set of known transcripts that are comprised of lists of nodes from a splice graph. By identifying splice junctions and nodes unique to each transcript, it can infer which transcripts are present in a set of aligned reads. The *find_splice_forms.py* script accepts as input a SAM file of RNA-Seq alignments and reports splice forms from annotated gene models where the RNA-Seq data provide sufficient evidence for them.:

```
find_splice_forms.py SAM-file [options]
```

Option	Value	Description
-d	OUTDIR	Output directory (overrides -o)
-j	MINJCT	Minimum junction threshold
-o	OUTPUT	Output file
-O	OVERLAP	Minimum number of bases that a read cluster must overlap a feature
-t	THRESHOLD	Minimum average read coverage for clusters

The script uses the following parameters to assess all genes in a gene model file:

Note that a gene model is absolutely required for making these predictions. The input gene model file may either be given as a command-line option (`-m`) or specified in your `SpliceGrapher.cfg` file.

Each splice form is represented by splice junctions or exons unique to that form. *find_splice_forms.py* loads RNA-Seq alignments and converts them into clusters of overlapping reads. When a cluster overlaps sequence that is unique to a single exon and that exon is unique to a single splice form, the splice form will be included in the final graph. Similarly, when spliced reads fall across a splice junction that is unique to a particular splice form, the splice form will be included in the final graph.

Three parameters control the sensitivity/specificity of this program: minimum average cluster read coverage (`-t`), a minimum number of splice junction reads (`-j`), and the minimum required cluster overlap (`-O`). Read clusters must have the desired minimum coverage before they are used to identify splice forms. Likewise, splice junctions must have the given minimum number of reads before they are used to identify forms. In cases where the RNA-Seq coverage does not uniquely cover any particular splice forms, no output graph will be produced.

You may specify an output file (`-o`), in which case all splice graphs will be written to the same file. The recommended approach is instead to specify an output directory (`-d`) in which case a separate file will be written for each gene for which a graph is produced.

Features That Uniquely Identify Splice Forms

In this context, a *feature* is that part of an exon that is completely unique to a particular splice form. Clusters must overlap unique exon *features* by a minimum amount (the default is 1, but a higher number is recommended for better specificity). For example, an exon that represents a retained intron will typically have as its unique feature that region from the start to the end of the associated intron. Some unique features may be smaller than the minimum overlap, for example, when the minimum is set to 10 and an alternative 3' site is a NAGNAG site whose unique feature is just 3nt long. For this reason, SpliceGrapher uses the smaller of the overlap size or the feature length.

E.5 Viewing Splice Graphs

E.5.1 Simple Viewing Tool

SpliceGrapher includes two scripts for viewing splice graphs. These scripts also serve as examples of how to use the SpliceGrapher viewing modules, many of which have been encapsulated in the `ViewerUtils` module. The simplest way to view one or more splice graphs is to use the *view_splicegraphs.py* script. For example, to view the graph stored in `gene_graph.gff`, simply enter:

```
view_splicegraphs.py gene_graph.gff
```

This produces a splice graph with minimal annotations. The script will accept more than one file on the command line. For example, to plot the files `AT1G01020.gff, AT1G01030.gff, AT1G01040.gff` and `AT1G01060.gff`, one might enter either of the following commands:

```
view_splicegraphs.py AT1G010[2346]0.gff
view_splicegraphs.py AT1G010*.gff
```

This can be powerful if you want to compare graph predictions for the same gene across different data sets (such as for different tissue samples or mutants):

```
view_splicegraphs.py data/var[1-3]/AT1G01020.gff
```

The script provides a selection of options that allow you to make the output graph as elaborate as you like:

Option	Value	Description
-a		(Re)annotate graphs
-m	MODEL	GFF gene model reference
-o	OUTPUT	Output file
-x		Show genomic positions of graph elements
-E	EDGE	Intron edge weight
-H	HEIGHT	Display area height, in inches
-W	WIDTH	Display area width, in inches
-L		Add legend to splice graph plot
-F	FONTSIZE	Font size for plot titles
-X		Use the same genomic position range for each plot
-S		Shrink introns

For example, to generate a more elaborate plot for the comparison above and write the output to a PDF file, one could enter:

```
view_splicegraphs.py data/var[1-3]/AT1G01020.gff -xLt var1,var2,var3 -o cmp.pdf
```

E.5.1.1 Shrinking Introns

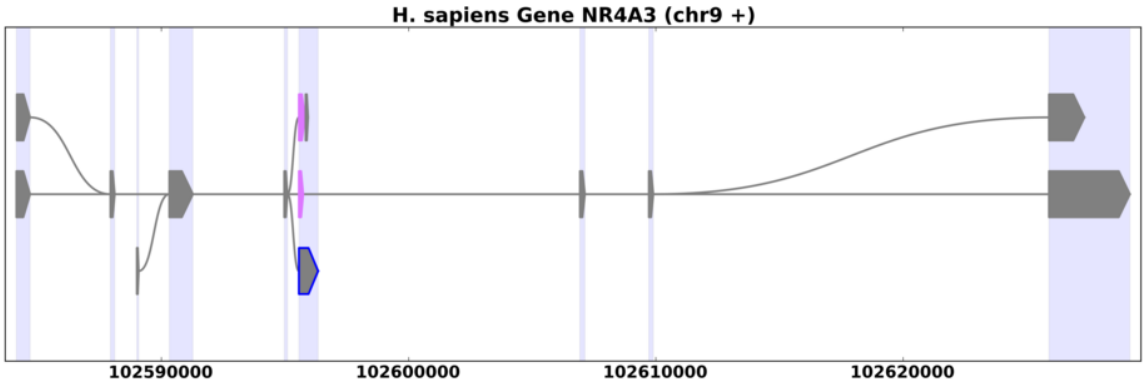

Figure E.2: Example of a splice graph for a human gene with large introns.

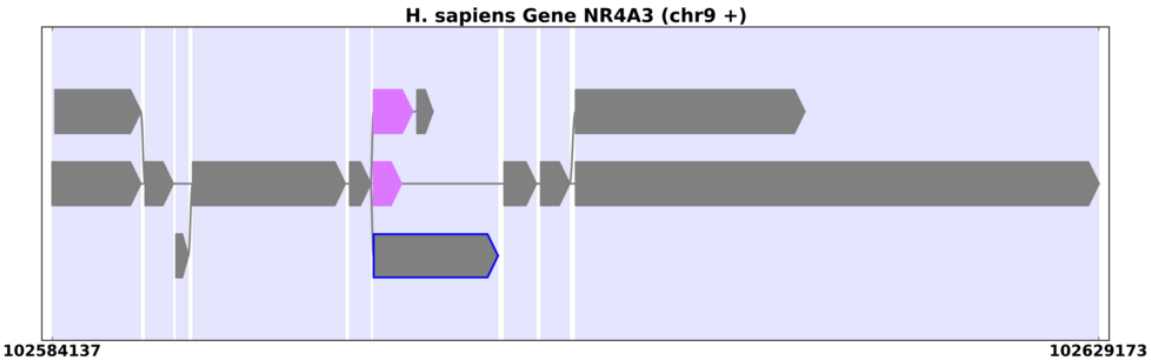

Figure E.3: The same human gene with introns reduced to emphasize exons.

Many organisms, especially mammals, have introns that are much longer than exons. This can make it difficult to see alternative splicing behavior on these plots. Figure 8.1 shows an example of a human gene that has large introns relative to its exons as generated from the command:

```
view_splicegraphs.py -m chr9.gff.gz NR4A3
```

You can change the way these appear in *view_splicegraphs.py* using the `-S` option (Figure 8.2):

```
view_splicegraphs.py -m chr9.gff.gz NR4A3 -S
```

E.5.1.2 Plot Types

SpliceGrapher can produce different kinds of plots for different kinds of data. These are: splice graphs, isoform plots, read coverage plots, splice junction plots and X-Y plots. These are described in more detail below:

E.5.1.3 Splice Graphs

Splice graphs are *SpliceGrapher's* principal plot type. These depict splice graphs that represent all the different ways in which a gene's exons may be combined.

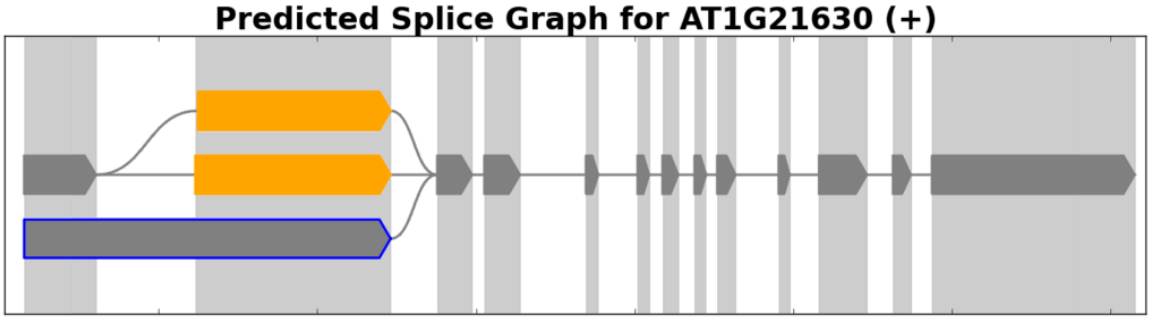

Figure E.4: Example of a splice graph for a gene from the plant *A. thaliana*.

E.5.1.4 Isoforms

Isoform graphs are similar to splice graphs in the way that introns and exons are depicted. However, instead of plotting the graph in its compact representation, isoform graphs show each possible splice form from a graph. These are particularly useful for showing splice forms represented in RNA-Seq data.

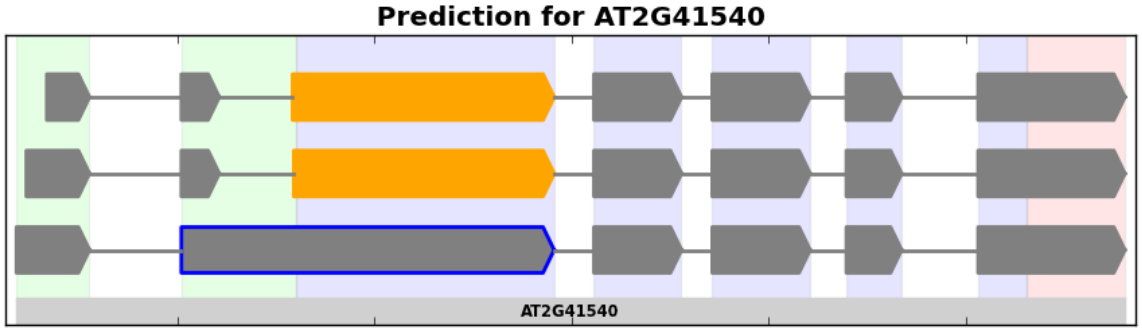

Figure E.5: Example of an isoform graph for a gene from the plant *A. thaliana*.

E.5.1.5 Read Coverage

Read coverage graphs depict the way RNA-Seq reads aligned to a gene region. The height of the graph at any given position represents the number of reads whose alignments overlap at that point.

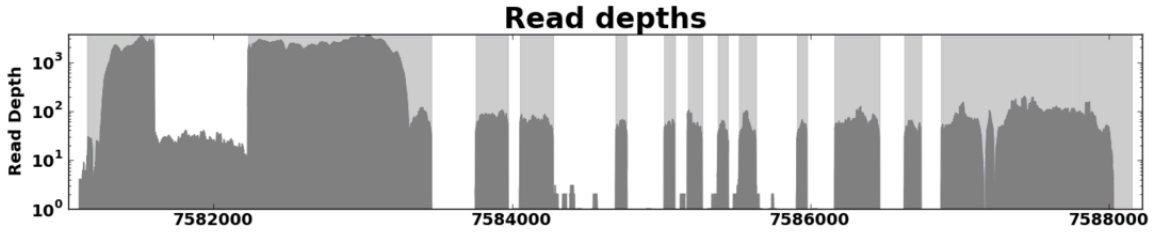

Figure E.6: Example of a read coverage graph for a gene from *A. thaliana*.

E.5.1.6 Splice Junction Coverage

Splice junction coverage graphs depict the way RNA-Seq reads aligned across splice junctions within a gene region. Splice junctions are depicted as '^'-shaped widgets flanked by bars that depict the anchor regions on either side of a junction. Labels for each splice junction show the number of reads that aligned across each junction.

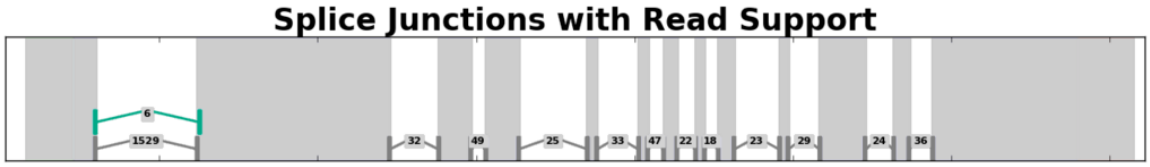

Figure E.7: Example of a splice junction coverage graph for a gene from *A. thaliana*.

E.5.1.7 X-Y Plots

SpliceGrapher also provides a generic X-Y plot that can depict arbitrary values across a gene region. These could be used, for example, to plot values from other forms of genomic activity (such as methylation) to see if there is any correlation between alternative splicing and these other phenomena.

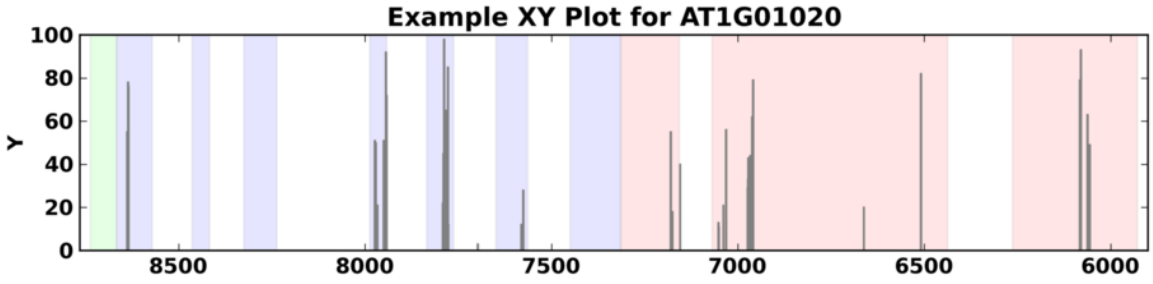

Figure E.8: Example of an X-Y graph for a gene from *A. thaliana* using randomly-generated values between 1 and 100 to simulate activity within the gene.

E.5.2 Multiple Plots

For more elaborate plots, *SpliceGrapher* includes a second script designed to show a predicted splice graph along with its accompanying evidence. It accepts a variety of data formats as input and can produce pages with one to four plots. Each of these plots requires a separate input file.

- Predicted splice graph (*SpliceGrapher* GFF format)
- Original gene model splice graph (*SpliceGrapher* GFF format)
- Short read depths (SAM format)
- (Optional) splice junctions with short read support (SAM format)

E.5.2.1 File Options

The options for this script are organized into file options and display options. File options tell the script what the kinds of input files to expect and how to render the output graphs. You may include any combination of input files and display them in any order (see Display Options below).

File Options	Value	Description
-m	MODEL	Gene models (GTF or GFF3 format)
-o	OUTPUT	Output plot file (infers format from file extension)
-d	DEPTH_FILE	SAM alignments (ungapped and spliced reads)
-s	SPLICE_GRAPH	GFF splice graph file
-G	ORIG_GRAPH	Optional original graph file
-X	XYDATA	Data file of X,Y value pairs for a simple graph

Note: SAM alignments may be specified in two ways. Some pipelines may produce separate files for ungapped alignments and spliced alignments, in which case both the -d and -j options should be used. Other methods (notably *Tophat*) produce a single file that contains both ungapped and spliced alignments, in which case just the -d option should be used.

E.5.2.2 Display Options

The display options are nearly the same as those for *view_splicegraphs.py*:

As an example, suppose we have used *SpliceGrapher* to predict a splice graph for the gene G123. We used *gene_model_to_splicegraph.py* to generate a splice graph for the gene model and stored it in `G123m.gff`, generated splice junction and read depth files in SAM format and used *predict_splicegraph.py*

Display Options	Value	Description
-c		Add read coverage labels to junctions
-x		Add genomic position labels to plots
-E	EDGE	Minimum edge thickness
-J	MINJCT	Minimum junction depth
-L		Add a legend to graph
-H	HEIGHT	Display window height (inches)
-W	WIDTH	Display window width (inches)
-F	FONTSIZE	Font size for plot titles
-S		Shrink introns relative to exons
-D	DISPLAY	Display order string for plots

to generate a splice graph prediction and stored it in `G123p.gff`. To produce a four-panel plot of the predicted graph along with its evidence, we could enter:

```
view_splicegraph_multiplot.py G123 -d chr1.sam -G G123m.gff -s G123p.gff
```

Note that the background for each of the panels includes a color-coded synopsis of the gene model. Thus we may choose to omit the gene model graph from the output and focus instead on the prediction and its evidence. In addition, suppose we wish to see the evidence at the top of the page and the predicted splice graph at the bottom. We may change all of these characteristics in a single step using the `-D` option. It accepts a string that determines the display order and whether or not to display each plot. Each character represents a specific plot: 'O' for the original gene model, 'P' for the predicted splice graph, 'J' for the splice junction plot, and 'R' for the read depth plot. The default display order is 'OPJR', so to modify a graph to omit the original graph and change the order, simply enter the string 'RJP':

```
view_splicegraph_multiplot.py G123 -d chr1.sam -G G123m.gff -s G123p.gff -D RJP
```

E.5.3 High-Quality Plotting

In addition to viewing utilities, *SpliceGrapher* provides plotting utilities that allow you to tune your plotting parameters for a specific purpose (such as figures suitable for a talk or paper) and save your configuration for later use. The program is called simply *plotter.py* and has the following format:

```
plotter.py gene.cfg
```

The plotting utility has a few options that allow you to try different output parameters such as page dimensions, but because the program provides a rich set of options, it is more convenient to store these in a configuration file (Figure E.9).

```
[SinglePlotConfig]
legend         = True
shrink_introns = True
height         = 5.0
width          = 18.0
fontsize       = 16
output_file    = NR4A3.pdf

[GeneModelGraph]
plot_type     = gene
gene_name     = NR4A3
relative_size = 10.0
source_file   = $GENE_MODELS/chr9.gff3.gz
file_format   = gene_model
title_string  = Gene Model for %gene

[SpliceGrapher]
plot_type     = splice_graph
relative_size = 10.0
source_file   = $GRAPHS/NR4A3_pred.gff
title_string  = SpliceGrapher Prediction for %gene

[Reads]
plot_type     = read_depth
relative_size = 8.0
source_file   = $SAM/chr9.sam.gz
title_string  = %gene Read Coverage
```

Figure E.9: This is an example of a plotter configuration file showing the main section (SinglePlotConfig) and three sub-plot sections including gene models (GeneModelGraph), a splice graph (SpliceGrapher) and read coverage (Reads). The parameters in the first section control overall output behavior, while parameters in each section control the behavior of individual sub-plots.

E.5.3.1 Main Section

The only required section is the main `[SinglePlotConfig]` section that provides general parameters for producing a single plot of one or more panels. The remaining sections each identify a distinct panel that will appear on the same plot in the output file. You may specify as many of these panels as you wish, provided each one has a unique name. The main difference between the graph sections and the main section is the parameters that are allowed for each section. *SpliceGrapher* enforces strict typing on these parameters and values.

Parameters for the `[SinglePlotConfig]` section are:

Name	Type	Description
fontsize	int	Font size for title strings
legend	boolean	Include legend on all plots
log_file	string	Write error messages to the given log file instead of stderr
height	float	Page height (in inches)
output_file	string	Path to output file
shrink_introns	boolean	Shrink introns relative to exons
width	float	Page width (in inches)

E.5.3.2 Plot Sections

Parameters for individual graphs are:

Name	Type	Description
acceptors	string	Comma-delimited list of acceptor sites to highlight (junctions only)
background	boolean	Show the gene model in the plot background
clusters	boolean	Show read depths as whole clusters (read_depth only)
donors	string	Comma-delimited list of donor sites to highlight (junctions only)
edge_weight	int	Edge weight to use when drawing intron edges
file_format	string	Type of data file to use (gene_plot only)
gene_name	string	Gene name to use (gene_plot only)
hide	boolean	Use the gene information but do not display the plot (gene_plot only)
highlight	string	Comma-delimited list of regions to highlight (read_depth only)
labels	boolean	Plot-dependent labels: exon ids in splice graphs, read count for junctions
min_coverage	float	Minimum read coverage required for showing junctions
relative_size	float	Relative plot size
plot_type	string	One of: gene/isoforms/junctions/read_depth/splice_graph/xy_plot
source_file	string	Path to file containing data for this plot
title_string	string	Text to use for plot title
x_labels	boolean	Show positions for all features in plot

Note: A gene model section is required for all plots that use shaded bars in the background. To prevent

the gene model itself from plotting, use the `hide` option.

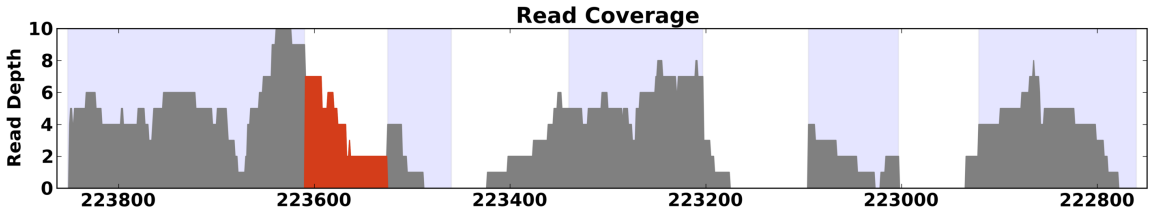

Figure E.10: Example of read depth plot with highlighting turned on to highlight possible intron retention activity.

E.5.3.3 Source Files for Plots

Each plot type is associated with specific file formats:

Plot Type	File Format	Description
gene	gene model GFF	Plots a gene model as a splice graph
isoforms	splice graph GFF	Plots elements in a splice graph as distinct splice forms
junctions	SAM	Depicts all splice junctions from alignments
read_depth	SAM	Depicts read coverage based on alignments
splice_graph	splice graph GFF	Plots a splice graph
xy_plot	CSV	Plots X-Y values

Note: plotting tools expect SAM files to be sorted. If you provide an unsorted SAM file to *plotter.py*, you may see nothing in the splice junctions or read depth sections of your plot.

E.5.3.4 Output Options

Although the configuration file allows you to specify output settings such as page width and height and font size, you may wish to try different settings to get a figure to look the way you want. Thus *plotter.py* provides command-line options to change these settings:

Option	Value	Description
-F	FONTSIZE	Font size (12-point)
-H	HEIGHT	Plotting area width (11.0 inches)
-W	WIDTH	Plotting area width (8.5 inches)
-o	OUTPUT	Output file

By default, a plot will appear on the screen, but if you specify an output file, *plotter.py* will output to the file, using the file extension to determine the file format. Valid file extensions are: emf, eps, pdf, png, ps,

raw, rgba, svg and svgz. These command-line settings will override settings in the configuration file. If no settings are specified in either place, the defaults are used.

E.6 File Formats

Given the variety of data used to generate predictions, *SpliceGrapher* uses several different file formats to process data.

E.6.1 Gene Model Files

E.6.1.1 GFF3 Files

Gene models are often stored using the GFF3 format and may contain a wide variety of different record types, though *SpliceGrapher* uses only a subset of these types:

- *chromosome* records are used to establish boundaries for all genes within a chromosome, but may also be inferred from the genes found within a chromosome.
- *gene* records are required to establish parent/child relationships within a gene using the "ID" attribute in the last column. A gene record is assumed to have one or more children given as exon records.
- *mRNA* records are used to infer exon and intron boundaries as well as parent/child relationships. Each record must have an "ID" attribute and a "Parent" attribute that refers to its gene. An mRNA record is assumed to have one or more children given as UTR or CDS records.
- *exon* records are used to infer exon and intron boundaries as well as parent/child relationships. Each record must have an "ID" attribute and a "Parent" attribute that refers to its gene.
- *CDS* records (CDS, UTR, FIVE_PRIME_UTR, or THREE_PRIME_UTR) are used to infer exon and intron boundaries as well as parent/child relationships. Each record must have an "ID" attribute and a "Parent" attribute that refers to its mRNA parent.

E.6.1.2 GTF Files

The annotations for many organisms (notably those on the ENSEMBL website) use GTF format instead. In version 0.1.0 we introduced a script *gtf2gff.py* that converts GTF files into GFF3 format that SpliceGrapher can understand. With version 0.2.0 *SpliceGrapher* can also load GTF files directly.

E.6.2 Splice Graph Files

SpliceGrapher also uses the GFF format to store splice graphs. The GFF format is a convenient way to express all the parent/child relationships in a graph, along with annotations for alternative splicing analysis and visualization purposes. A *SpliceGrapher* GFF file uses the following record types:

- *graph* records define a splice graph. These records include attributes that include the gene name and a brief description of how the splice graph differs from the gene models. Note that this replaces the old *cluster* name, which has been deprecated.
- *parent* records define exons in a splice graph that act as root nodes in the graph.
- *child* records define all other exons in a graph, including leaf nodes.

Both *parent* and *child* records include attributes that describe unique identifiers for each exon, the splicing forms that include them, and the alternative splicing forms associated with them. Child records also include their parents' unique identifiers. Many of these attributes are used by *SpliceGrapher's* visualization modules.

E.6.3 SAM Files

SAM files are similar to GFF files but are used to store short-read depths after reads have been aligned to a genome (for a complete description, see `http://samtools.sourceforge.net/SAM1.pdf`). *SpliceGrapher* uses short read alignments stored in the SAM format to predict splice graphs and to display read depths and splice junctions on plots.

E.6.4 BED and WIG Files

Some spliced alignment tools use BED and WIG files to store information about short-read alignments and splice junctions that have short-read support. *SpliceGrapher* can read these files to predict splice graphs and to display read depths and splice junctions on plots. For more information on these file formats, see `http://genome.ucsc.edu/FAQ/FAQformat.html`.

E.6.5 PSL Files

Programs that perform EST or cDNA alignments, such as BLAST and GMAP, can produce alignment information in PSL format. *SpliceGrapher* uses PSL files to construct splice graphs from these alignments

that may then be merged with other graphs to make predictions. For details on the PSL file format, see `http://fungi.ensembl.org/info/website/upload/psl.html`.

E.7 Pre-Built Classifiers

Starting with version 0.2.0, *SpliceGrapher* now comes with a set of pre-built classifiers for over 100 different species. Only classifiers with adequate performance are included, so some species will have classifiers for canonical (GT donors/AG acceptors) and semi-canonical (GC donors) sites while others will have classifiers only for canonical sites.