

# How many transcripts does it take to reconstruct the Splice Graph?

Paul Jenkins

September 16, 2005

## Abstract

Alternative splicing has emerged as an important biological process which increases the number of transcripts obtainable from a gene, and has been observed in up to 74% of human genes. Given a sample of transcripts, the alternative splicing graph (ASG) can then be constructed—a mathematical object minimally explaining these transcripts. A natural question that arises is: how large is this graph relative to the true ASG of the gene? In this work we propose a probabilistic model of transcript generation from a gene idealized as a real interval  $[0, L]$ . The growth of the ASG with sampled transcripts was investigated for different probabilities and different example genes. The applicability of different models was assessed for five sample genes and these preliminary results support the idea that splicing regulation can vary widely both between and within genes. Finally, using graph theory a polynomial-time algorithm is provided to calculate the minimal number of transcripts required to recover a given ASG.

## Introduction

In recent years the phenomenon of alternative splicing has been studied in greater depth from a bioinformatics approach, thanks to the emergence of new tools such as expressed sequence tag (EST) analysis and microarray studies. It has been able to account for a number of important unresolved issues. For example, why can one not elucidate the relative complexity of two organisms by comparing the number of their genes? Even allowing for our anthropocentric assumptions of superiority, it seems surprising that gene databases such as Ensembl record 24,194 human genes compared to say 24,405 in the frog *X. tropicalis* (Ensembl July 2005). One answer is that alternative splicing can vastly increase the transcriptome of a particular genome, enabling one gene to code for numerous proteins. Some studies estimate the number of human genes undergoing alternative splicing to be as high as 74% (Johnson *et al.* 2003). Leipzig *et al.* (2004) estimate 5% of human genes each to provide more than 100 putative transcripts. It is now thought that alternative pre-mRNA splicing is a central mode of genetic regulation in higher eukaryotes.

Splicing is a mechanism that takes place after the initial transcription of DNA into a complementary RNA strand (known as precursor messenger RNA, or pre-mRNA), but before its translation into protein. The splicing process modifies pre-mRNA by selecting regions to be discarded—known as *introns*—and retaining the rest. Those regions going on to be translated are termed *exons*. The resultant strand of ligated exons is the mature mRNA. Thus, pre-mRNA can be seen as a sequence of (possibly disparate) exons, flanked on each side by an intron. ‘Alternative splicing’ refers to the observation that the splicing process can be performed in a number (possibly thousands) of different ways. Most exons are *constitutive*, that is, always retained in the mRNA, yet others have been seen to be omitted in certain situations. Exons which may be omitted in the splicing process are called *cassette exons* (sometimes the term is reserved for those which are also flanked on each side by constitutive exons). Individual exons may also undergo alternative splicing if they contain alternative 5’ or 3’ splice sites. The exon’s length may be altered at its 3’ or 5’ end respectively (terminology is reversed because an alternative 5’ splice site refers to an alternative 5’ end for the next intron). The mRNAs themselves may have alternative 5’ or 3’ ends, with alternative selection of the 5’-most or 3’-most exons. Another type of alternative splicing event is

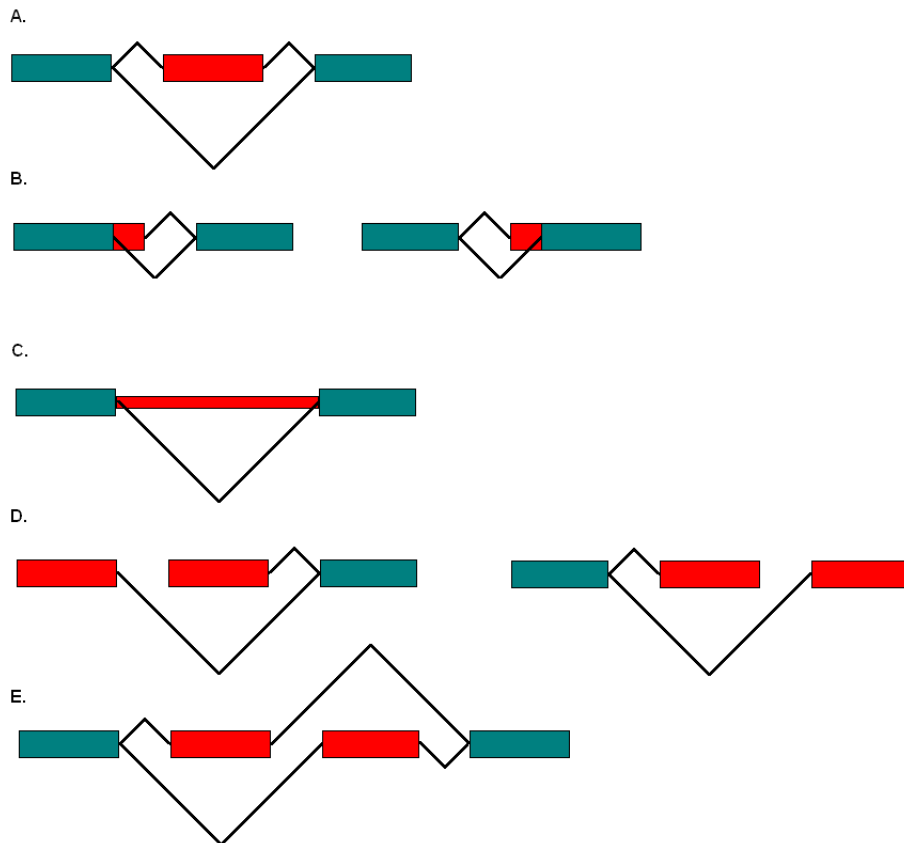


Figure 1: **Basic patterns of alternative splicing** (Black 2003, Modrek & Lee 2002). Exons are represented as rectangles. Constitutive exons are shown in green, regions which may be spliced out in red. Black lines represent paths of translation, from left to right. **A. Cassette exon.** It can either be included in the mature mRNA or not. **B. Alternative 5' and alternative 3' splice sites.** An exon may be lengthened or shortened. **C. Retained intron.** Introns are usually excised from the final mRNA. In some circumstances they are not. **D. Alternative promoter and alternative polyadenylation sites.** Translation may begin or end at different positions along transcript, governed by use of promoters and poly(A) sequences respectively. **E. Mutually exclusive exons.** More complicated patterns are possible, such as only one of two mutually exclusive exons being selected. Other 'logic-gate' filters may also be possible. Often, these relationships are controlled by exon lengths. If one exon is included it can cause a frame-shift, thereby precluding other downstream exons.

a *retained intron*, in which an intron flanked by exons is also included in the final mRNA. The 'building blocks' of alternative splicing events are shown in figure 1. The percentage of human genes observed to exhibit different splicing behaviours are 30.3% for cassette exons, 17.1% for competing 5' splice sites, 17.8% for competing 3' splice sites and 31.0% for retained introns (Leipzig *et al.* 2004). Some or all of these patterns may be observed from translations of a gene's mRNA, leading to potentially complex overall splicing patterns (figure 2).

In fact the distribution of splicing 'complexity' varies widely: from the  $\sim 30\%$  of genes not exhibiting alternative splicing, to the 0.4% with more than 5000 possible transcript reconstructions. The distribution for the human genome is shown in figure 3. However, in many cases only a very small proportion of potential transcripts has been observed. Consequently, curation of the transcripts associated with genes is often poor. Of the 89 genes offering more than 5000 possible transcripts, none reveal more than 10 as recorded by Ensembl, though projects are underway to improve the annotation (such as HAVANA). Individual alternative splices within genes can be better characterized by EST and microarray data. ESTs are single-pass reads from complementary-DNA (cDNA), effectively providing snapshots of a segment of a transcript in a given set of conditions; microarrays can measure expression levels in a large sample

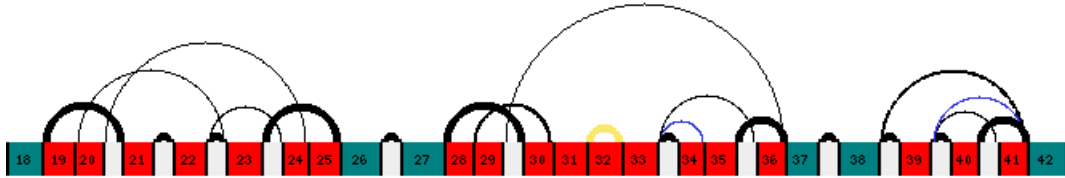


Figure 2: An example of more complicated splicing patterns: exon fragments 18–42 of the human gene PPFIA3 (not to scale). Splicing events are represented by curved edges. Intron retention event shown in yellow. Competing 3' splice site shown in blue. More complicated and nested relationships are also visible. Edge thicknesses are proportional to EST support for that splicing event. The ASG of this gene offers more than 5000 putative transcripts, though only 4 are recorded by Ensembl (July 2005). Image derived using the Alternative Splicing Gallery (Leipzig *et al.* 2004).

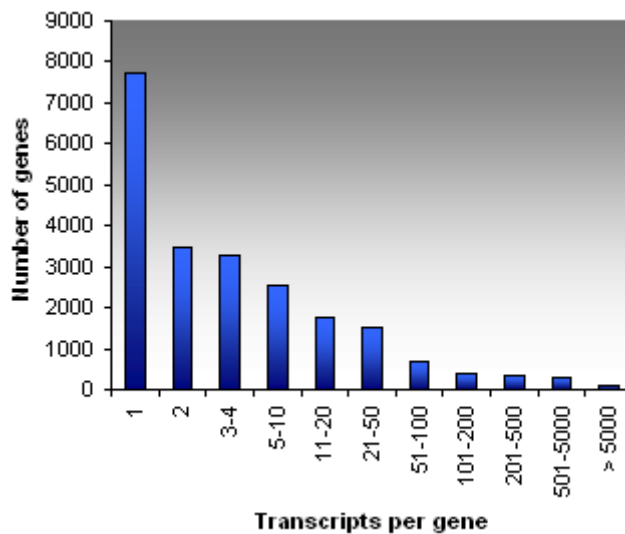


Figure 3: Distribution of the number of all possible transcript reconstructions per gene, for 22127 genes of *Homo sapiens* (data taken from Leipzig *et al.* 2004). Note that 89 of those genes offer more than 5000 putative transcripts.

of genes simultaneously, and have been used effectively to measure exon skipping events in alternatively splicing genes (e.g. Johnson *et al.* 2003, Pan *et al.* 2004). Only 12.8% of alternative splicing relationships have been observed in full length transcripts (Lee *et al.* 2003), implying that for many genes the detailed nature of the transcriptome is yet to be fully characterized, and hence that measuring reconstruction of the alternative splice graph is a worthwhile undertaking.

The junctions between introns and exons are marked by special signature sequences known as *splice sites*, and acted upon by the *spliceosome*, a large macromolecular complex responsible for carrying out the splicing procedure. It assembles onto each intron from a set of five small nuclear ribonucleoproteins (snRNPs) and other accessory proteins. The details of the splicing process are intricate (see Black 2003 for further details), and are governed by the presence or absence of many other proteins as well as neighbouring spliceosomes and the splice site sequence. Regulation in different cell types enables one gene to be used in different ways and at different times. Because many alternatively spliced products are themselves important regulators, alternative splicing patterns can influence critical development decisions such as sex and death. One of the best known examples is the sexual identity of *Drosophila melanogaster* (Black 2003, Lopez 1998). The regulatory gene at the top of the sex determination pathway is the RNA binding protein *Sex lethal* (Sxl), which is expressed specifically in female fruitflies and represses splicing patterns that would lead to male development. A downstream target of Sxl includes transcripts from the *Transformer* (Tra) gene, which is itself a splicing regulator. Alternative splicing decisions can have therefore far-reaching consequences. A second example is to be seen in the CD44 gene, which contains

an array of 10 cassette exons labelled v1–v10. Combinatorial splicing of just this region of the gene confers  $2^{10} = 1024$  protein isoforms, though only a small fraction of these have been observed (Roberts & Smith 2002). Crucially, certain isoforms are observed only in certain circumstances. Some are seen only in tumour cells; inclusion of exon v6 enables metastasis to be induced (Gunthert *et al.* 1991). CD44 is well-studied due to its association with cancer, however the regulatory features of alternative splicing in many other genes are less understood.

Traditionally the transcriptome of a gene has been represented by an exhaustive list of its splice variants. However, as the prevalence yet abstruse nature of alternative splicing has become apparent, a need for more concise notation has emerged. Heber *et al.* (2002) introduced the idea of the *alternative splice graph* (ASG). This enables the set of possible transcripts to be represented in a single graph, avoiding the error-prone nature of case-by-case transcript reconstruction. Denote the ASG by  $G = (V, E)$ . It is defined as follows. Let  $\{s_1, \dots, s_n\}$  be the set of RNA transcripts of the gene of interest, with each  $s_k$  corresponding to a sequence of genomic positions  $V_i$  ( $V_i \neq V_j$  for  $i \neq j$ ). Define  $V := \bigcup_i V_i$ , the set of all transcribed positions, and  $E := \{(v, w) : v \text{ and } w \text{ form consecutive positions in at least one transcript } s_i\}$ . Hence the ASG  $G$  is a directed graph and a putative transcript is any path in  $G$ . The graph is always acyclic, since the exons present in any spliced transcript are retained in the correct 5' to 3' linear order (Black 2005, Ibrahim *et al.* 2005). Edges can only traverse towards a downstream genomic position. Henceforth we assume a transcript begins and ends at one of a gene's alternative promoter and polyadenylation sites respectively, so that paths always proceed from a source to a sink. Even with this restriction, note that the reconstruction of the ASG immediately throws up more potential transcripts than the number required to construct the graph (figure 4). Studies of the ASG can be viewed either as 'liberal'—seeking to discover all potential transcripts, or 'conservative'—seeking to explain a given ASG as parsimoniously as possible.) A final modification made to  $G$  is the collapsing of strings of vertices with *indegree* = *outdegree* = 1 into a single vertex. So each exon fragment (i.e. portion of an exon bounded by two splice sites) is represented by a single vertex rather than a string of vertices for each nucleotide therein. This enables the ASG to be illustrated in a similar manner to those shown in figure 2—simply imagine vertices to be the numbered blocks, and the directed edges to be the arcs traversing from left to right plus an edge connecting each consecutive vertex (the latter not drawn). The ASG is a convenient, compact representation of all the splicing events associated with a particular gene. Additionally, a particular advantage for our purposes is that it provides a natural measure of the splicing complexity of the gene: the number of edges in the ASG. By collapsing each exon fragment into a single vertex, the only edges that matter are those associated with splicing events. Long exons do not unduly skew this measure because of their numerous internal edges.

Alternative splicing appears to underlie many important biological processes—development, regulation, disease. As a concise representation of the splicing capability of a gene, the ASG lends itself to much further investigation. It would be interesting to model the growth of the ASG as more and more transcripts are sampled, as a measure of how fast knowledge of the gene's potential can grow. How many transcripts do we need before the full ASG can be reconstructed, either theoretically or through the experimental sampling of transcripts? Do different genes exhibit different patterns in the growth of their ASG? By introducing a simple mathematical representation of a gene, it is hoped that we can predict the answers to some of these questions.

## A Stochastic Model

### Model 1

We are interested in the probabilities of observing different transcripts from a given gene. For simplicity we will not consider the cases—such as those of CD44 and Sxl outlined above—in which certain alternative splicing events control or are controlled by external events. Tissue-specific control appears to be restricted to a relatively small number of specialised genes. 667 out of 30793 alternative splice relationships (2.2%) have been observed with high confidence to be tissue-specific (Lee *et al.* 2003), though this fraction may increase with further research. For now however we will think of the probability of selecting a particular transcript to be taken across say, many samples of identical cells at the same time with all

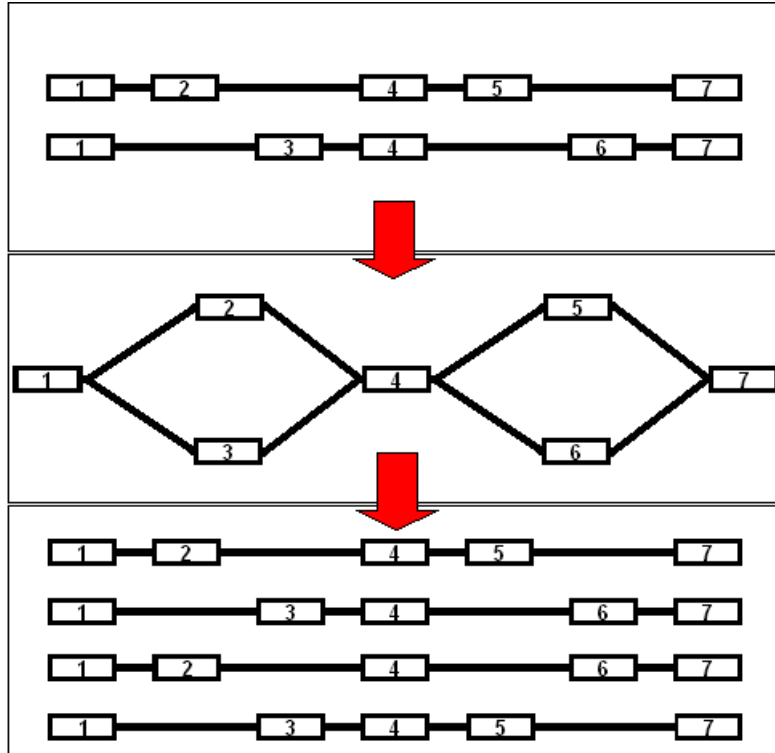


Figure 4: Expansion of the transcriptome. Two transcripts  $\{(1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 7), (1 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7)\}$  (**top**) over seven vertices  $V = \{1, 2, \dots, 7\}$  produce an ASG (**middle**—all edges proceed from left to right) with the additional potential transcripts  $\{(1 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 7), (1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7)\}$  (**bottom**).

other things being equal. Nor will we concern ourselves with having to deal with sequencing errors: transcripts are always assumed to be spliced at exact locations. Let us propose the following simple model for the generation of transcripts from a gene. We may approximate the discrete structure of a gene by an interval of the real line  $[0, L]$ . Within the gene is a (fixed) set  $S$  of *splice sites*, which we define as any point on the line such that there exists a mature transcript from the gene with the sequence on one side of the splice site excluded and the sequence on the other side included.  $S$  therefore contains all junctions between any potential exon and intron, including junctions within exons that may serve as alternative 5' or 3' splice sites. Based on the true underlying ASG there exist pairwise probabilities for each pair of splice sites  $(s_1, s_2)$  representing the probability that if an mRNA contains the exon ending at  $s_1$  then it jumps forward to  $s_2$ . These probabilities can be captured as a  $|S| \times |S|$  probability matrix  $P$ , with entries defined by  $p_{ij} =$  probability of a transcript jumping from splice site  $i$  to splice site  $j$ , given that it contains  $i$ . Since exons are always retained in a transcript in the correct linear order,  $P$  is strictly upper triangular. The sum of any row of  $P$  represents the overall probability of a splicing event occurring at this site. The difference between this value and 1 is the probability that no splicing event occurs here, i.e. a transcript encompasses the neighbourhoods either side of the splice site. Transcript generation can thus be seen as a walk along the line  $[0, L]$ , jumping forwards (or not) with well-defined probabilities at each splice site.

To account for the features of figure 1D, we can define sets  $I, T \subseteq S$  of initiation splice sites and terminal splice sites, respectively. For each walk to generate a transcript, rather than beginning at 0 we proceed randomly from one of the  $i \in I$ . With each  $i \in I$  associate a probability  $p_i^{start}$  of initiating the transcript from that site (so  $\sum_{i \in I} p_i^{start} = 1$ ). Similarly, for each walk that reaches a splice site  $t \in T$ , transcript generation is then terminated. Thus, the complete model is captured by the collection  $(S, P, I, \mathbf{p}^{start}, T)$  (figure 5).

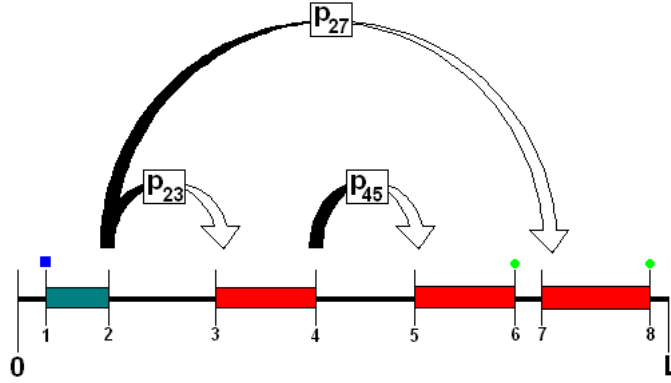


Figure 5: **Model 1.** A simple example of transcript generation. Constitutive exon shown in green. Exons which may be spliced out shown in red. In this simple example we might label the splice sites as  $S = \{1, 2, 3, 4, 5, 6, 7, 8\}$ . Then  $P$  is the zero matrix other than  $p_{23}, p_{27}$  and  $p_{45}$ .  $I = \{1\}$  (marked by a blue square),  $p_1^{start} = 1$ ,  $T = \{6, 8\}$  (marked by a green circle). Transcripts are generated by ‘walking’ from  $1 \in I$  until we reach one of the  $t \in T$ , jumping forward at each splice site with the relevant probabilities in  $P$ .

Pseudocode for the generation of transcripts might look like this:

```

do current splice site  $\leftarrow$  sample from  $I$  with probability distribution  $\mathbf{p}^{start}$ 
previous splice site  $\leftarrow$  current splice site
jump  $\leftarrow$  0
transcript  $\leftarrow$   $\emptyset$ 

while current splice site  $\notin T$ 
  if jump == 0
    then transcript  $\leftarrow$  transcript  $\cup$  [previous splice site, current splice site]

  do with probability  $\sum_k P(\text{current splice site}, k)$ :
    jump  $\leftarrow$  1
    next splice site  $\leftarrow$  scaled sample from  $P(\text{current splice site}, .)$ 
  else
    next splice site  $\leftarrow$  min{splice sites: splice site > current splice site}
    jump  $\leftarrow$  0

  previous splice site  $\leftarrow$  current splice site
  current splice site  $\leftarrow$  next splice site

```

## Model 2

The above model is useful if we’d like to model a gene with possibly complex relationships between splice sites (as in figure 2). However, we might like to test whether an observed splicing graph is really part of some more general relationship between splice sites. A more flexible model might be this. Assume that exons present in all the transcript samples seen so far—those candidates that may be constitutive—are indeed constitutive (see below). Between these exons, suppose that any two splice sites might somehow be linked. With each splice site  $x \in S$ , associate two probabilities  $p_x^{in}, p_x^{out}$ , the probabilities of jumping ‘into’ and ‘out of’ transcription. Conceptually we can imagine travelling along the real line from 0 to  $L$ , and as we reach each splice site jumping ‘in’ with probability  $p^{in}$  if we are ‘out’, or jumping ‘out’ with probability  $p^{out}$  if we are ‘in’. As a splice site is reached while ‘in’ we are in effect specifying some fixed probability of a splicing event occurring (which could be based on the

properties of that splice site), compounded with a geometric distribution for the choice of the next splice site, given that a splicing event has occurred. A geometric distribution implies the general preference for nearby exons to be concatenated together in a transcript, a vastly simplifying assumption of in reality a far more complicated process (Kornblihtt *et al.* 2004). By weighting the parameter of the distribution at different splice sites, additional knowledge about them can be incorporated (recall that exon inclusion also depends on factors such as splice site sequence and other regulatory regions). In this model we can view a transcript as a walk along the gene, flipping between these states which we loosely call ‘in’ and ‘out’ with certain probabilities at each splice site. By concatenating all those subintervals of  $[0, L]$  for which we are ‘in’, this provides a simple method for generating mature transcripts. Note that only two probabilities are used at each splice site rather than up to  $n$  for each site in model 1, where  $n = |S|$ . Model 2 seeks to explain the splicing events we observe with far fewer parameters. Compare the  $O(n^2)$  parameters of model 1 with  $O(n)$  of model 2.

So we might characterize basic splicing events as in figure 1A–C as follows (assuming each block is surrounded by introns):

A constitutive exon between splice sites $x$ and $y$ :	<table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"></td><td style="padding: 2px 5px;"><math>x</math></td><td style="padding: 2px 5px;"><math>y</math></td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{in}</math></td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{out}</math></td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> </table>		$x$	$y$	$p^{in}$	1	0	$p^{out}$	0	1						
	$x$	$y$														
$p^{in}$	1	0														
$p^{out}$	0	1														
A cassette exon between $x$ and $y$ with probability $p$ of selection:	<table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"></td><td style="padding: 2px 5px;"><math>x</math></td><td style="padding: 2px 5px;"><math>y</math></td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{in}</math></td><td style="padding: 2px 5px;"><math>p</math></td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{out}</math></td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> </table>		$x$	$y$	$p^{in}$	$p$	0	$p^{out}$	0	1						
	$x$	$y$														
$p^{in}$	$p$	0														
$p^{out}$	0	1														
An exon occupying $[x, z]$ with an alternative 3’ splice site at $y$ :	<table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"></td><td style="padding: 2px 5px;"><math>x</math></td><td style="padding: 2px 5px;"><math>y</math></td><td style="padding: 2px 5px;"><math>z</math></td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{in}</math></td><td style="padding: 2px 5px;"><math>p</math></td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{out}</math></td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> </table>		$x$	$y$	$z$	$p^{in}$	$p$	1	0	$p^{out}$	0	0	1			
	$x$	$y$	$z$													
$p^{in}$	$p$	1	0													
$p^{out}$	0	0	1													
An exon occupying $[x, z]$ with an alternative 5’ splice site at $y$ :	<table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"></td><td style="padding: 2px 5px;"><math>x</math></td><td style="padding: 2px 5px;"><math>y</math></td><td style="padding: 2px 5px;"><math>z</math></td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{in}</math></td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{out}</math></td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;"><math>p</math></td><td style="padding: 2px 5px;">1</td></tr> </table>		$x$	$y$	$z$	$p^{in}$	1	0	0	$p^{out}$	0	$p$	1			
	$x$	$y$	$z$													
$p^{in}$	1	0	0													
$p^{out}$	0	$p$	1													
Exons $[w, x]$ and $[y, z]$ with an intervening retained intron:	<table style="border-collapse: collapse; margin: auto;"> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"></td><td style="padding: 2px 5px;"><math>w</math></td><td style="padding: 2px 5px;"><math>x</math></td><td style="padding: 2px 5px;"><math>y</math></td><td style="padding: 2px 5px;"><math>z</math></td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{in}</math></td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td><td style="padding: 2px 5px;">0</td></tr> <tr><td style="border-right: 1px solid black; padding: 2px 5px;"><math>p^{out}</math></td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;"><math>p</math></td><td style="padding: 2px 5px;">0</td><td style="padding: 2px 5px;">1</td></tr> </table>		$w$	$x$	$y$	$z$	$p^{in}$	1	0	1	0	$p^{out}$	0	$p$	0	1
	$w$	$x$	$y$	$z$												
$p^{in}$	1	0	1	0												
$p^{out}$	0	$p$	0	1												

$I, T \subseteq S$  and  $\mathbf{p}^{\text{start}}$  are defined as in model 1. Thus, the complete model is captured by the collection  $(S, \mathbf{p}^{\text{in}}, \mathbf{p}^{\text{out}}, I, \mathbf{p}^{\text{start}}, T)$  (figure 6).

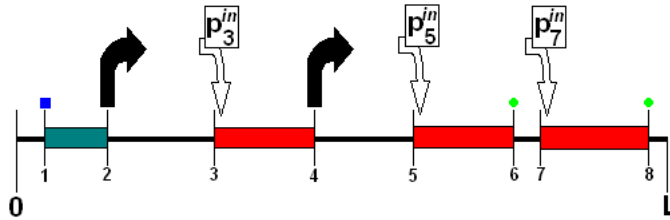


Figure 6: **Model 2.** A simple example of transcript generation. The same gene as in figure 5 with the same values in  $S, I, T, \mathbf{p}^{\text{start}}$ , but with a proposed more general ASG. In this example,  $\mathbf{p}^{\text{in}} = (1, 0, p_3^{\text{in}}, 0, p_5^{\text{in}}, 0, p_7^{\text{in}}, 0)$  and  $\mathbf{p}^{\text{out}} = (0, 1, 0, 1, 0, 1, 0, 1)$ . Transcripts are generated by ‘walking’ from some  $i \in I$  until we reach some  $t \in T$ , jumping in or out at each splice site with the relevant probabilities in  $\mathbf{p}^{\text{in}}, \mathbf{p}^{\text{out}}$ . In this example it permits additional possible edges in the ASG, such as  $(2, 5)$ , provided the unspecified probabilities  $p_3^{\text{in}}, p_5^{\text{in}}$  and  $p_7^{\text{in}}$  are strictly between 0 and 1.

Pseudocode for the generation of transcripts might look like this:

```

do current splice site  $\leftarrow$  sample from  $I$  with probability distribution  $\mathbf{p}^{\text{start}}$ 
transcription state  $\leftarrow$  'in'
transcript  $\leftarrow$   $\emptyset$ 

while  $\neg(\text{previous splice site} \in T \wedge \text{transcription state} == \text{'in'})$ 
  if transcription state == 'in'
    then transcript  $\leftarrow$  transcript  $\cup$  [previous splice site, current splice site]
      with probability  $p_{\text{current splice site}}^{\text{out}}$ , transcription state  $\leftarrow$  'out'
    else with probability  $p_{\text{current splice site}}^{\text{in}}$ , transcription state  $\leftarrow$  'in'

    previous splice site  $\leftarrow$  current splice site
    current splice site  $\leftarrow$  next splice site

```

Why do we apply this model only to regions between exons believed to be constitutive? Consider figure 9. Under model 1 all splicing events are assumed to be given only by all those edges shown. Under model 2 we are asking: could this ASG have been reconstructed from a sample in which the true underlying ASG permits all possible splicing relationships between exon fragments 18–25, and all possible splicing relationships between exon fragments 27–40? Now it becomes clear why we leave in the restriction that exon fragment 26 is indeed constitutive (by setting its  $p^{\text{in}}$  value to 1). If this were not specified then under model 2 edges would frequently be generated between splice sites either side of fragment 26. We do not know that this is physically impossible, but maintaining this restriction does seem a reasonable question to investigate from ASGs such as figure 9. We can see it as an intermediate step to the even more general model which would allow splicing relationships to occur between any two exons. Of course, one could go further and also test this (call it model 3), but for genes with many overlapping splicing relationships such as that in figure 9 it seems intuitively far less likely to have been generated in this way than from model 2. With further thought one could investigate a whole hierarchy of similar models, with parameter spaces of different orders of magnitude, though we shall restrict our investigation into models 1 and 2. Whether an exon is in fact constitutive is also a question that could be addressed statistically.

## Minimal number of transcripts required

We are now in a position to investigate the growth of the reconstructed ASG as the number of sampled transcripts increases. A useful result would be to compare this with the minimal possible number of transcripts required to provide the full ASG. In graph theoretic terms, what we require is the minimal number of paths required to provide an edge covering of a directed acyclic graph. (An *edge covering* of a graph  $G = (V, E)$  is a set of paths  $\{P_1, \dots, P_k\}$  such that  $\forall (u, v) \in E \exists j \in \{1, \dots, k\}$  such that  $(u, v) \in P_j$ .) Without loss of generality we may assume each path traverses from a source to a sink, since in any such minimal collection each path can be extended appropriately. A polynomial time algorithm for similarly-defined graphs has been provided previously by Li *et al.* (1989), but since it was used for logic-gate circuits with weighted edges it is more involved than is necessary for the equivalent algorithm on an ASG. Here we present a simpler polytime method for determining the size of a minimal edge covering of a directed acyclic graph  $G$ :

1. Construct any edge covering of  $G$ . This can easily be achieved in polytime, as follows:
  - (a) Store all the edges of  $G$  in a list.
  - (b) Start a new path by picking an edge in the list.
  - (c) Extend this path to a source and a sink.
  - (d) Remove all the edges in this path from the list.
  - (e) Repeat until the list of uncovered edges is empty.

We could easily improve the efficiency at each step by preferentially choosing uncovered edges, though the efficiency here is not important (provided it remains in polynomial time). All that matters is that we construct some edge covering. Set the variable  $n$  to be the size of this covering.

2. Define the *weight* of an edge with respect to this covering to be the number of paths in the covering containing this edge. Calculate the weight of each edge.
3. Construct a path from a source to a sink which traverses only forward edges with weight  $\geq 2$  or over any backward edge (where forwards and backwards refer to the direction we traverse the edge compared to the orientation of the edge itself) ( $\dagger$ ). Since this definition of a ‘path’ differs from our previous use of the word (which traverses only forwards edges), let’s use the alternative notation *passage*. A passage can easily be constructed using something like a breadth-first search (such as that given in Cormen *et al.* (1990) P470, modified appropriately. See appendix for this code in MATLAB). If no passage exists subject to the weight restrictions ( $\dagger$ ), the answer is  $n$  and we are done.
4. If we have constructed such a passage, for each edge traversed forwards in the passage decrement its weight by 1, and for each edge traversed backwards in the passage increment its weight by 1. Decrement  $n$  by 1. Repeat the previous step.

To prove that this algorithm works, we need some definitions:

**Definition:** A *topological cut* of a directed graph  $G = (V, E)$  is a partition  $(S, T)$  of  $V$  such that  $\forall (u, v) \in E, u \in T \Rightarrow v \in T$ .

**Definition:** The *size* of a topological cut  $(S, T)$  is  $|\{(u, v) \in E: u \in S \text{ and } v \in T\}|$ .

**Proposition:** The algorithm given above does indeed provide the size of a minimal edge covering.

**Proof:** First note that the size of any edge covering of  $G$  is larger than the size of any topological cut of  $G$ ; we require at least one path in any edge covering for each edge crossing from  $S$  to  $T$ . Hence, for any topological cut and any edge covering:

$$\text{size}(\text{topological cut}) \leq \max_{\text{cuts}} (\text{size}(\text{topological cut})) \leq \min_{\text{coverings}} (\text{size}(\text{edge covering})) \leq \text{size}(\text{edge covering}).$$

So if we find a topological cut and edge covering of the same size, they must be maximal and minimal respectively. It suffices to show (A) that when the algorithm terminates we can construct a topological cut of size  $n$ , and (B) there exists a new edge covering with one fewer path each time we visit step 4. First we show this in the case when there is only one source  $s \in V$  and one sink  $t \in V$ .

(A) The algorithm settles on its final edge covering when there does not exist a passage from  $s$  to  $t$  traversing only edges forwards with weight  $\geq 2$  or edges backwards of any weight. Define  $S$  to be all vertices that *can* be reached from  $s$  in this manner, and  $T = S^c$ . Now, for any  $u \in T, \forall v \in V$  such that  $(u, v) \in E$  we must have  $v \in T$ , otherwise  $u$  can be reached by traversing backwards from  $v$  in  $S$  and we should have had  $u \in S$  (by definition of  $S$ ). Hence  $S$  and  $T$  define a topological cut. Moreover, the weight of any edge  $(u, v)$  crossing the cut with  $u \in S, v \in T$  must be 1; if it were any higher then we could construct a passage from  $s$  to  $v$  subject to ( $\dagger$ )—and so we should have had  $v \in S$ . So the size of this topological cut is equal to the size of the corresponding edge covering.

(B) It remains to argue that for each visit to step 4 of the algorithm we can construct a new edge covering based on the current one, by splicing and stitching different elements of the existing paths appropriately, so that one of the paths becomes redundant and can simply be deleted. (Note that the algorithm does not actually have to record the edge covering at each step, since we are only interested in its size.) We have a passage satisfying ( $\dagger$ ). Proceed by considering each edge of the passage in turn. We aim to construct a new edge covering of  $G$  such that each forward edge in the passage will ultimately be covered by one fewer edge, each backward edge in the passage will ultimately be covered by one additional edge, and the size of the new edge covering will be one less than the size of the current one. Each edge in the passage is dealt with in turn. At each stage we have a current edge covering  $\{P'_1, \dots, P'_{k-1}, P\}$  containing a path  $P$  we are in the process of making deletable. To initiate the procedure set the current edge covering to be one which was passed to step 4 of the algorithm:  $\{P'_1, \dots, P'_{k-1}, P\} = \{P_1, \dots, P_k\}$ . Without loss of generality we can begin by choosing  $P = P_k$ . Suppose we have reached a vertex  $u$  in the passage, having

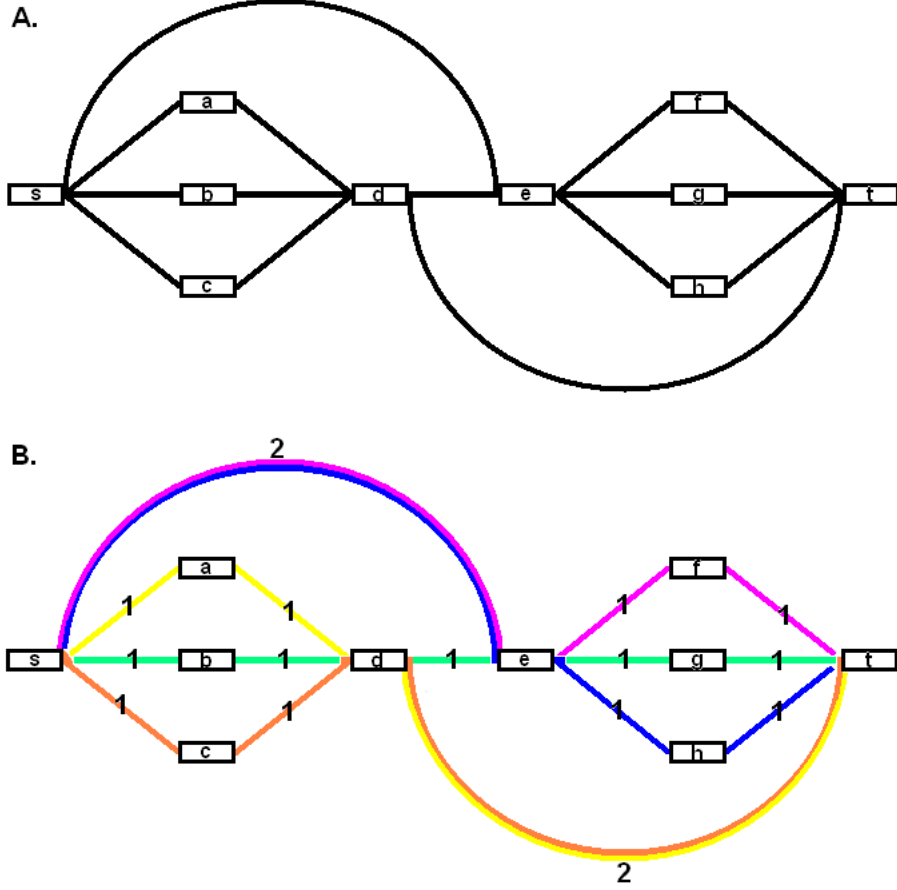


Figure 7: **A.** An example directed acyclic graph, to which the algorithm in the text will be applied. The vertex set is  $\{s, a, b, c, d, e, f, g, h, t\}$ . All edges proceed from left to right. It is easy to see that a minimal edge covering requires precisely 4 paths. **B.** An edge covering which utilizes 5 paths, chosen in such a way that to obtain the minimal edge covering the passage will need to explore both forward and backward edges. We will try to make the path marked pink the deletable path  $P$ . The weight of each edge with respect to this covering is shown. A passage satisfying  $(\dagger)$  is  $\{(s, e), (e, d), (d, t)\}$ .

dealt with each upstream vertex in the passage and performed the appropriate stitchings. Denote the next edge in the passage  $(u, v)$ . Either  $(u, v) \in E$  or  $(v, u) \in E$ . Suppose  $(u, v) \in E$ , so its weight is  $\geq 2$ . If  $(u, v) \in P$  just proceed to the next vertex in the passage; this edge in  $P$  is verified also to be deletable. Otherwise choose a path  $P'_j$  in the current edge covering with  $(u, v) \in P'_j$ . If the paths are of the form

$$P = \{(s, x_1), (x_1, x_2), \dots, (x_m, u), (u, w), (w, x_{m+1}), \dots, (x_n, t)\},$$

$$P'_j = \{(s, y_1), (y_1, y_2), \dots, (y_m, u), (u, v), (v, y_{m+1}), \dots, (y_n, t)\}$$

with  $w \neq v$  (and the  $x$  and  $y$  some vertices in which we are uninterested), define the new edge covering by modifying these two paths:

$$P \mapsto \{(s, x_1), (x_1, x_2), \dots, (x_m, u), (u, v), (v, y_{m+1}), \dots, (y_n, t)\}$$

$$P'_j \mapsto \{(s, y_1), (y_1, y_2), \dots, (y_m, u), (u, w), (w, x_{m+1}), \dots, (x_n, t)\}$$

i.e. cut and switch the two paths at  $u$ . This ensures both that the next edge in  $P$  is also deletable, and that  $\{P'_1, \dots, P'_{k-1}, P\}$  still forms an edge covering of  $G$ . Decrementing the weight of  $(u, v)$  by one reflects that when we delete  $P$ , one fewer path passes through this edge. See figures 7 and 8 for illustration.

In the second case, consider  $(v, u) \in E$ . If  $(v, u) \in P$  just proceed to the next vertex in the passage; this edge in  $P$  is verified to be deletable (this case occurs when there are two backward edges of the



The process is continued until  $t$  is reached. Due to the way  $P$  has been constructed, it can simply be removed and  $\{P'_1, \dots, P'_{k-1}\}$  is still an edge covering.

Finally we need to consider the case when there exists possibly more than one source and sink. Simply apply the above algorithm to  $G' = (V', E')$ , defined as follows.  $V' = V \cup \{a, b\}$ ,  $E' = E \cup \{(a, s): s \text{ is a source in } G\} \cup \{(t, b): t \text{ is a sink in } G\}$ .  $a$  is a ‘supersource’,  $b$  a ‘supersink’. These are the only sources and sinks in  $G'$ . It is clear that any edge covering of  $G$  can be extended to one of  $G'$ : simply extend each path by one edge at each end to reach  $a$  and  $b$ . This process is reversible and bijective, thus a minimal edge covering for  $G'$  provides a minimal edge covering for  $G$  of the same size.  $\square$

## Results

### Data simulation

It would be of interest to perform a large number of simulations, each of which samples a large number of transcripts from a gene and reconstructs the corresponding ASG accordingly. We can then begin to ask questions such as: how often is the full ASG reconstructed after sampling 10 transcripts? After 100 transcripts? What is the expected number of transcripts required to cover 50% of the ASG? To cover 99% of the ASG? To cover 100%? To illustrate the sorts of questions that can be investigated, consider an example. Take one of the more complicated human genes with more than 5000 putative transcripts, ABCB5 (Ensembl ID ENSG00000004846). Its complexity arises from the long series of unusual inter-relationships between sequential exon fragments (figure 9). Each sample represents a path from one of the sources in its ASG to one of the sinks. We might assume under model 1—to which we shall restrict our attention during data simulations—that the known ASG (a portion of which is shown in figure 9) is in fact the full one. Snapshot reconstructions of the ASG during one simulation are shown in figure 10.

A more concise representation is a graphical one. The growth of individual simulations for gene ABCB5 are shown in figure 11A. As observed in figure 10, there is indeed a general trend of rapid initial growth followed by a slowing down, as only a few edges remain undiscovered by our collection of transcripts. For a very large number of simulations, the mean trend can be plotted as in figure 11B.

Conversely we might ask: what is the expected number of transcripts required to discover particular fractions of the ASG? Data obtained from the simulations shown in figure 11 is shown in table 1.

% of ASG	50	90	99	100
Expected number of transcripts	2.6	16.5	44.5	59.2

Table 1: Expected number of transcripts required to cover different portions of the ASG of human gene ABCB5, calculated across 10000 simulations with equally weighted probabilities.

Data generated by these simulations ascribes equal probabilities to splicing events for each ‘decision’ that a simulation makes—the situation in which without prior knowledge of the gene one might expect growth of the reconstructed ASG to be fastest (see below). Even in this favourable situation, from the table we observe a rapid increase in the number of transcripts we expect to need in order to cover more demanding proportions of the true ASG. To reconstruct 99% of the ASG for this gene we expect to have to sample at least 45 transcripts. With certain splicing events in reality more favourable than others,

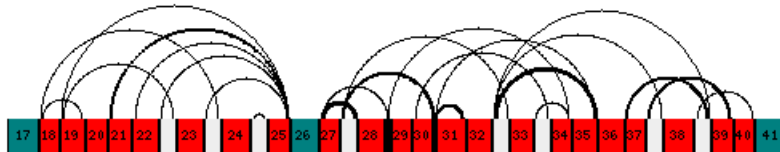


Figure 9: Exon fragments 17–41 of the human gene ABCB5 (not to scale). Image derived using the Alternative Splicing Gallery (Leipzig *et al.* 2004).

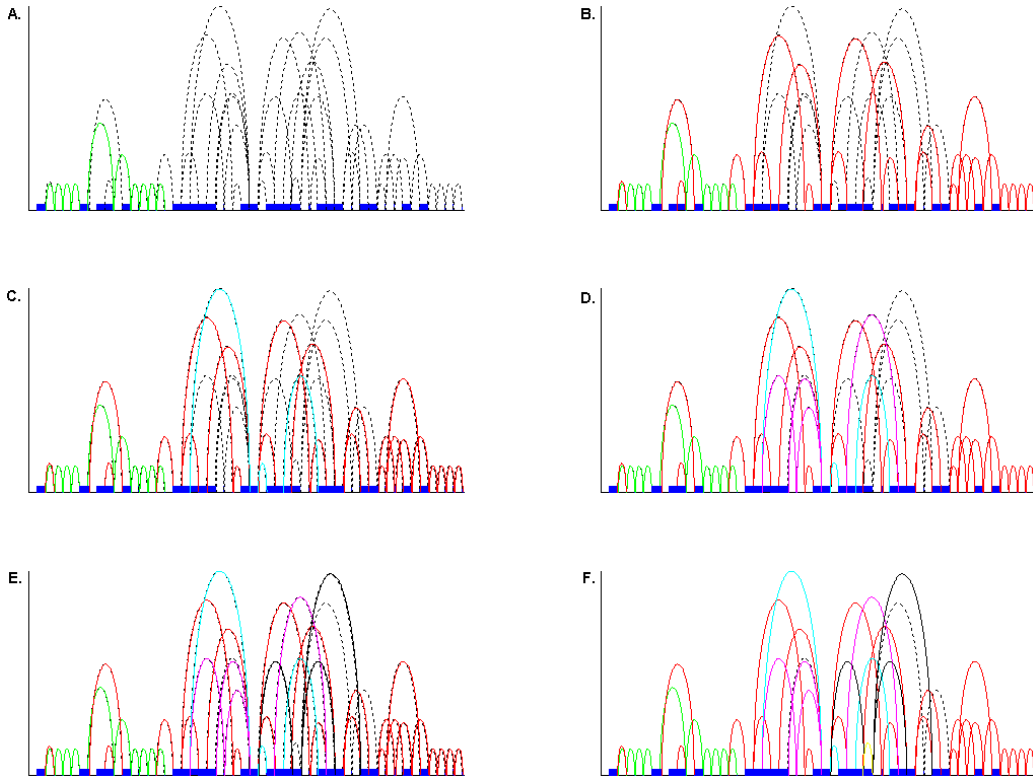


Figure 10: Example illustration of growth of the ASG of human gene ABCB5 under model 1. Edges in the full ASG are shown as dotted lines. All probability parameters  $p_i^{start}$  and  $p_{ij}$  are set with equal weightings. **A.** After 1 transcript (new edges in green). **B.** After 5 transcripts (red). **C.** After 10 transcripts (cyan). **D.** After 20 transcripts (magenta). **E.** After 30 transcripts (black). **F.** After 40 transcripts (yellow). Note the general trend of rapid early growth preceding fewer additions later on. (See appendix for MATLAB code to generate these images.)

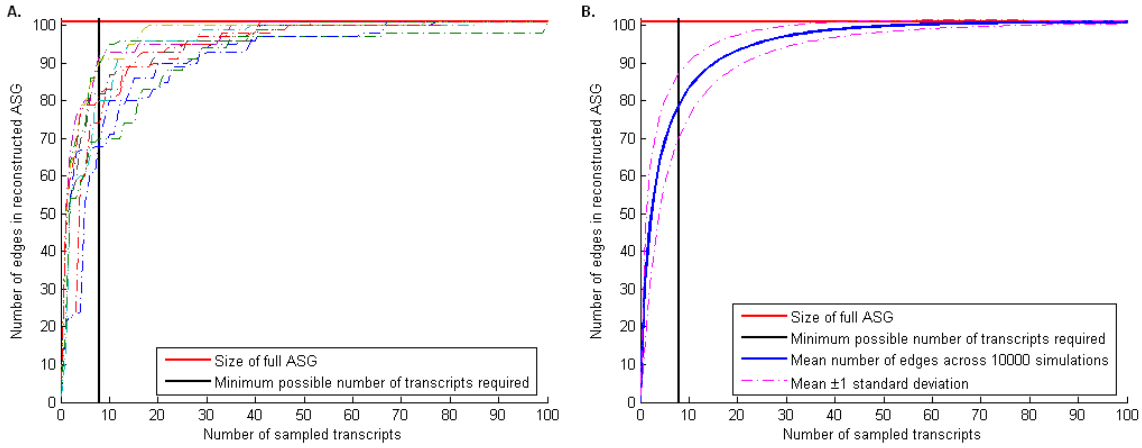


Figure 11: **A.** Ten simulated reconstructions of the ASG for human gene ABCB5, under model 1. Size of the reconstructed ASG is measured in the number of edges between exon fragments that have been ‘discovered’: this is plotted against the number of sampled transcripts. The full ASG size is marked in red (MATLAB code for the calculation of this value is given in the appendix). The minimal number of transcripts required to reach this full ASG size, as calculated using the proposition above, is shown in black. **B.** Mean number of reconstructed edges across 10000 simulations.

and different splices being more prevalent in different tissue-types or at different development times, this expectation will surely increase.

Another problem that can be addressed is to consider: of the transcripts required to reconstruct the full ASG, which ones were ‘*useful*’—i.e. which ones actually revealed additional edges of the ASG? Now, in our simulated data under model 1, since the probability distribution for a splicing event is independent of splices that have already occurred in the same transcript, this question will not be very revealing. The answer will merely be the first transcripts to uncover a new edge at a particular position on the graph. But real genes do not always behave in this way. For experimentally obtained transcripts answering this question will be a vital tool in determining phenomena such as distant coupling of alternatively splicing exons, a major question for future research. Useful transcripts can be used alongside the ASG to demonstrate correlations between the inclusion-level of different pairs of exons, a feature explicitly indiscernible from the ASG alone. For our simulated data we shall restrict ourselves not to the nature of the useful transcripts but rather their number. Figure 12 shows the distribution of the number of useful transcripts across the 10000 simulations reported in figure 11B. The minimal number of transcripts required to reconstruct the full ASG is annotated; it is also the minimal possible number of useful transcripts.

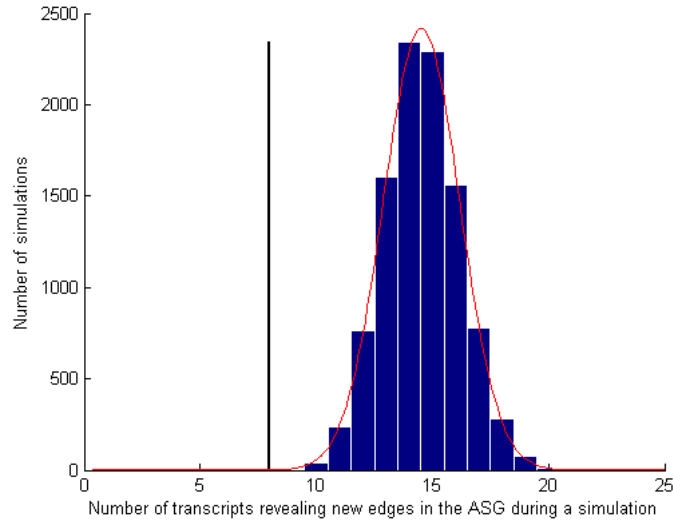


Figure 12: Distribution of the number of transcripts revealing new edges during a simulation, for 10000 simulations of transcript generation of human gene ABCB5 under model 1. The minimal number of transcripts required to recover the full ASG is indicated by a black line. Note that the data displays a good fit to the best Normal approximation (in red:  $\hat{\mu} = 14.53$ ,  $\hat{\sigma} = 1.65$ ). We will use this for plotting convenience in other displays (figure 17).

Simulations to investigate the questions detailed above were performed on a sample of 10 genes from the cohort of the 89 most complex human genes. The results for these genes are shown in figures 13, 14, 15, 16 and 17. Table 2 illustrates the varied nature of this sample. It would be of interest to repeat these simulations for the same splicing graphs but with different probability distributions attached to each splice site. To investigate this behaviour in a systematic way, introduce a variable we’ll call *splicing bias*, denoted in the figures and appendices as  $q$ . This is a simple measure of the probability of a splicing event occurring or not occurring, at any splice site where this decision can be taken.  $q$  ranges from 0 (splicing events do not occur) to 1 (each possible splicing decision is equally likely). So at a splice site with  $m$  outgoing edges, one of which represents continuing inside this exon, each outgoing edge has probability  $\frac{q}{m}$  apart from the ‘remain inside’ edge which has probability  $1 - (m - 1)\frac{q}{m}$ . Varying this parameter does not allow us to examine the effect of the probability distribution at specific splice sites, but it does provide us with a general impression of the effect of changing the probability distributions together across the whole gene.

Gene	Ensembl ID	Exon fragments	Starting exons	Terminal exons	Retained introns	Competing 5' sites	Cassette exons	Competing 3' sites	Other splices
ABCB5	ENSG00000004846	51	2	2	1	1	0	0	31
ACADVL	ENSG00000072778	52	2	1	11	1	0	5	15
CAPN3	ENSG00000092529	67	4	3	2	6	3	4	11
SORBS1	ENSG00000095637	38	2	1	1	1	11	3	4
EEF1D	ENSG00000104529	33	2	1	4	4	0	2	20
Y310_HUMAN	ENSG00000148396	53	5	2	3	1	3	4	7
BPAG1	ENSG00000151914	124	6	3	3	4	2	1	10
ATP9B	ENSG00000166377	53	1	3	1	0	9	3	13
SHMT2	ENSG00000182199	32	3	1	6	5	0	2	8
CLN3	ENSG00000186015	28	1	1	5	1	8	0	7

Table 2: Counts of features of 10 sample genes on which simulations of transcript generation were performed. ‘Other splices’ includes mutually exclusive exons, nested events and any other unclassified splicing relationships.

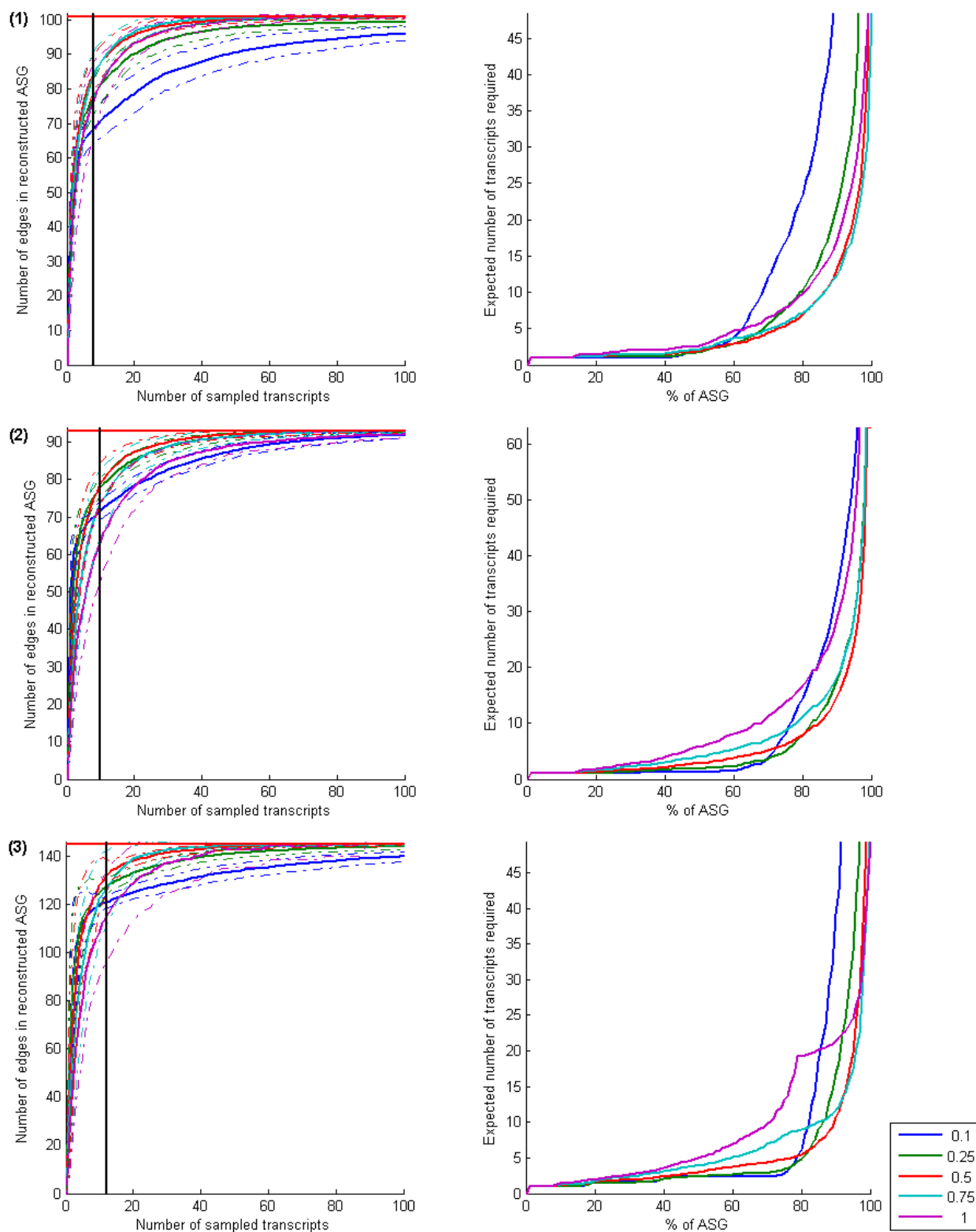


Figure 13: **Left.** Mean number of reconstructed edges across 100 simulations for  $q = 0.1, 0.25, 0.5, 0.75, 0.1$ . Mean  $\pm 1$  standard deviation are shown as dashed lines. **Right.** Expected number of transcripts required to reconstruct different fractions of the ASG, for the same selected values of  $q$ . Data are for human genes (1) ABCB5, (2) ACADV1, (3) CAPN3.

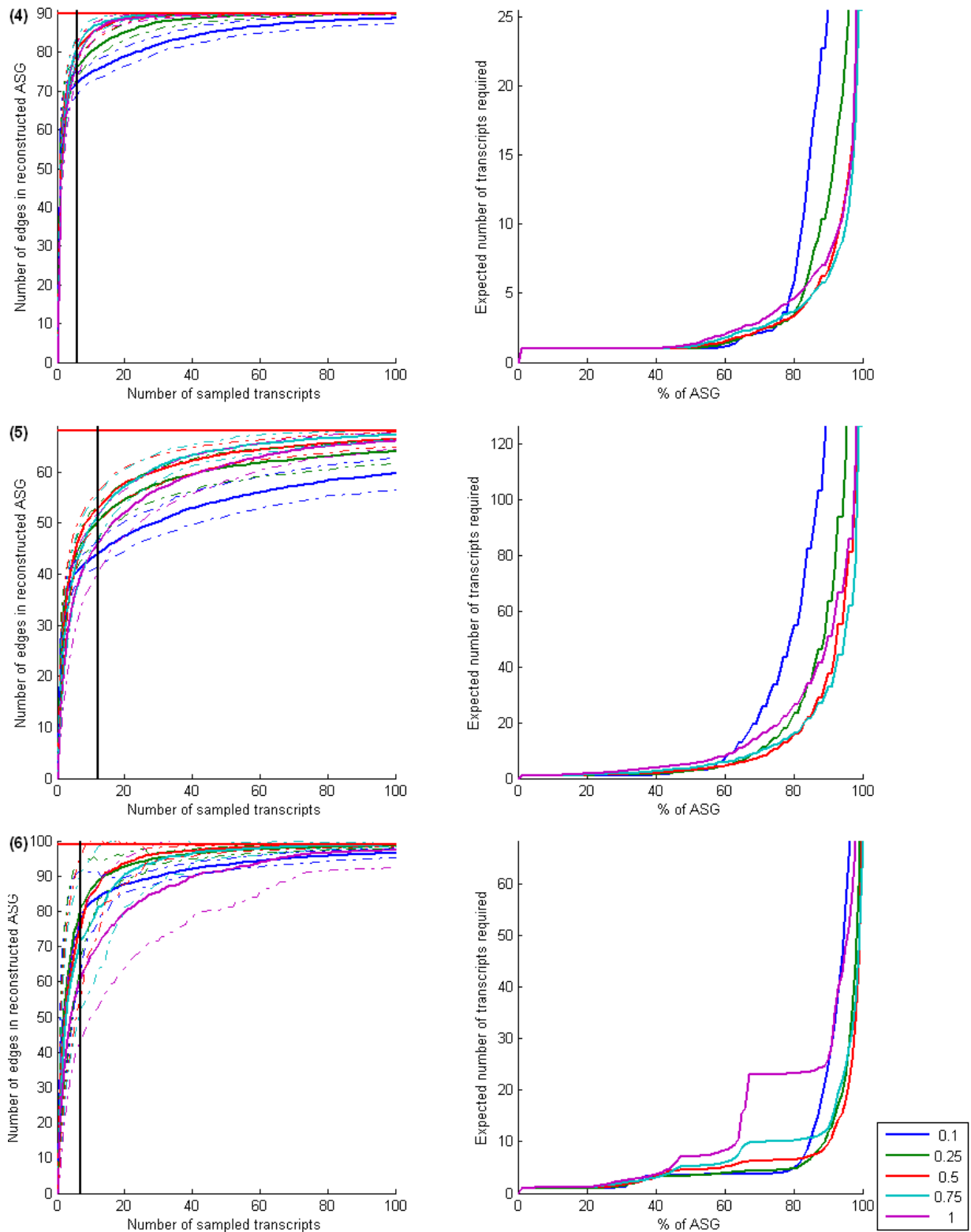


Figure 14: **Left.** Mean number of reconstructed edges across 100 simulations for  $q = 0.1, 0.25, 0.5, 0.75, 0.1$ . Mean  $\pm 1$  standard deviation are shown as dashed lines. **Right.** Expected number of transcripts required to reconstruct different fractions of the ASG, for the same selected values of  $q$ . Data are for human genes (4) SORBS1, (5) EEF1D, (6) Y310\_HUMAN.

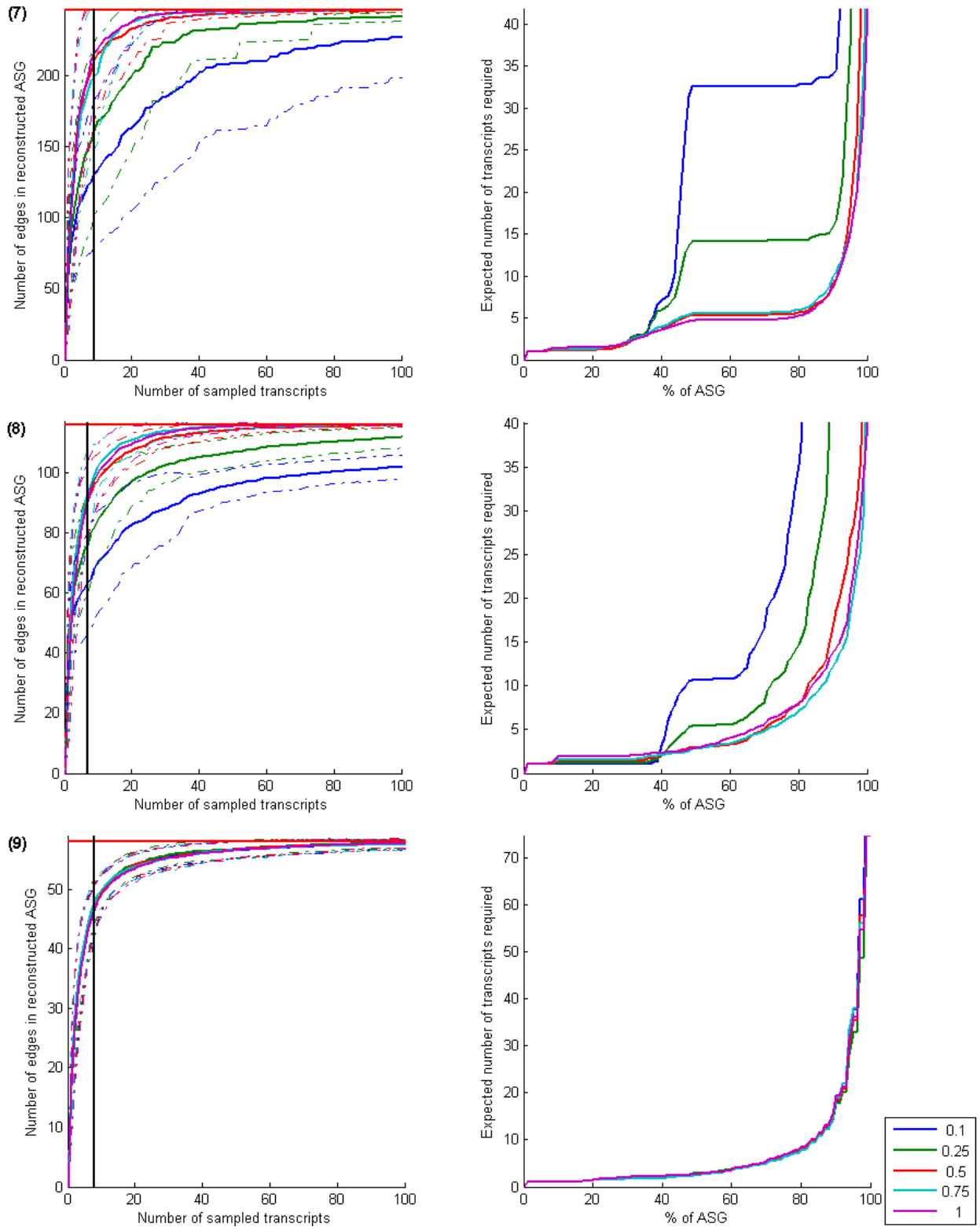


Figure 15: **Left.** Mean number of reconstructed edges across 100 simulations for  $q = 0.1, 0.25, 0.5, 0.75, 1$ . Mean  $\pm 1$  standard deviation are shown as dashed lines. **Right.** Expected number of transcripts required to reconstruct different fractions of the ASG, for the same selected values of  $q$ . Data are for human genes (7) BPAG1, (8) ATP9B, (9) SHMT2.

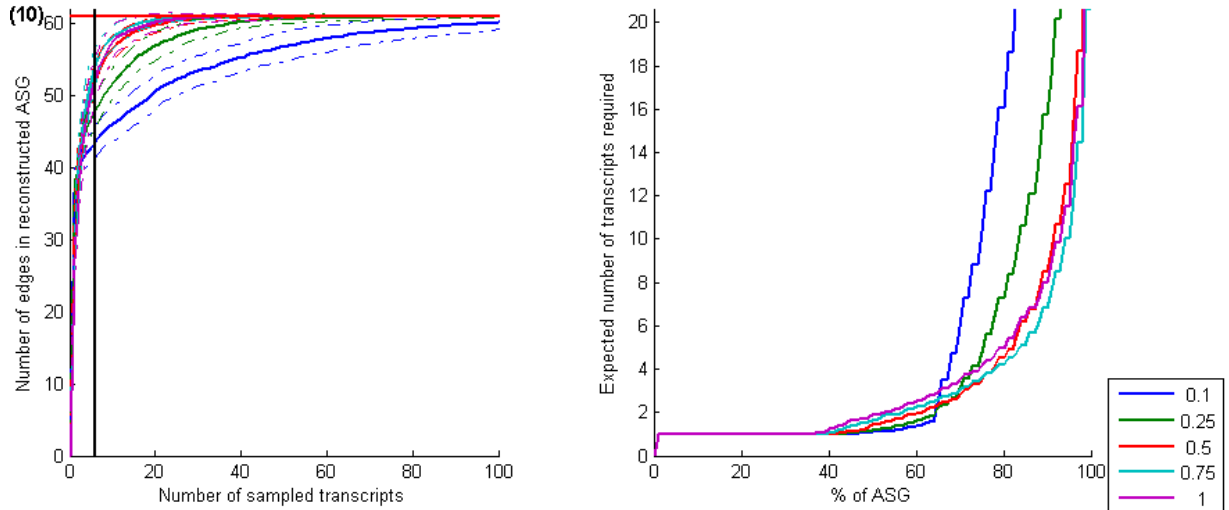


Figure 16: **Left.** Mean number of reconstructed edges across 100 simulations for  $q = 0.1, 0.25, 0.5, 0.75, 1.0$ . Mean  $\pm 1$  standard deviation are shown as dashed lines. **Right.** Expected number of transcripts required to reconstruct different fractions of the ASG, for the same selected values of  $q$ . Data are for human gene (10) CLN3.

## Statistical testing

Thus far we have taken the ASGs for real genes and assumed in our modelling that these are indeed the full ASGs. They provide convenient examples of the sorts of graphs we hope to reconstruct. Remember however that they themselves are reconstructions from experimental samples of transcripts or partial transcripts. An interesting problem would be to use these reconstructed ASGs to try to ask questions of the nature of the full ASGs for these genes. We would like to identify whether any genes show statistical evidence for a rather freer relationship between exons than their reconstructed ASGs would suggest. This can be achieved by performing a likelihood ratio test to compare models 1 and 2, the latter being nested inside the former since in model 2 each  $p_{ij}$  is determined by fewer parameters. Given a sample of transcripts from a gene, its reconstructed ASG can be explained either by model 1 or by model 2. Model 2 is more ‘daring’ in the sense that it attempts to explain the same data with an order of magnitude fewer parameters. Thus, for a sample of transcripts  $\{s_1, \dots, s_n\}$  starting and terminating at two constitutive exons of interest, the likelihood ratio statistic in which we are interested is:

$$\Lambda = \frac{\sup_{\mathbf{q}_2} L_2(\mathbf{q}_2)}{\sup_{\mathbf{q}_1} L_1(\mathbf{q}_1)},$$

where  $\mathbf{q}_1, \mathbf{q}_2$  are the sets of parameters under each model and  $L_1 \propto \prod_{i=1}^n P(s_i)$ ,  $L_2 \propto \prod_{i=1}^n P(s_i)$  are the likelihoods of the data under each model. The constant of proportionality in each likelihood is the same multinomial coefficient, which cancels from  $\Lambda$ . The probability of each transcript is the product of the relevant probabilities involved in its generation. For example in figure 5,  $P([1, 2] \cup [3, 4] \cup [5, 6]) = p_{23}p_{45}$ , and in figure 6,  $P([1, 2] \cup [5, 6]) = p_2^{out}(1 - p_3^{in})p_5^{in}$ . If model 2 holds then  $-2 \ln \Lambda \sim \chi^2(z)$ , where  $z$  is the difference in the number of parameters between the two models.

The likelihood ratio test could be applied to any gene and an associated set of probabilities. The test statistic will depend heavily on the choice—so far arbitrary—of probabilities  $P$ ,  $\mathbf{p}^{in}$ ,  $\mathbf{p}^{out}$ , so to make slightly more meaningful inferences we attempt here to base probabilities on real data. Ideal for obtaining large-scale data on alternative splicing events is microarray technology. However, using microarrays to study alternative splicing is still in its infancy, with only a handful of large-scale investigations into exon skipping events (Lee & Wang 2005 and references therein). Still fewer have been able to offer a reliable quantitative analysis. One exception is Pan *et al.* (2004), though this concentrates on specific alternative splicing events in mice. For now then we must content ourselves with the primary historical source for identifying alternative splicing events: EST coverage. The Alternative Splicing Gallery provided by

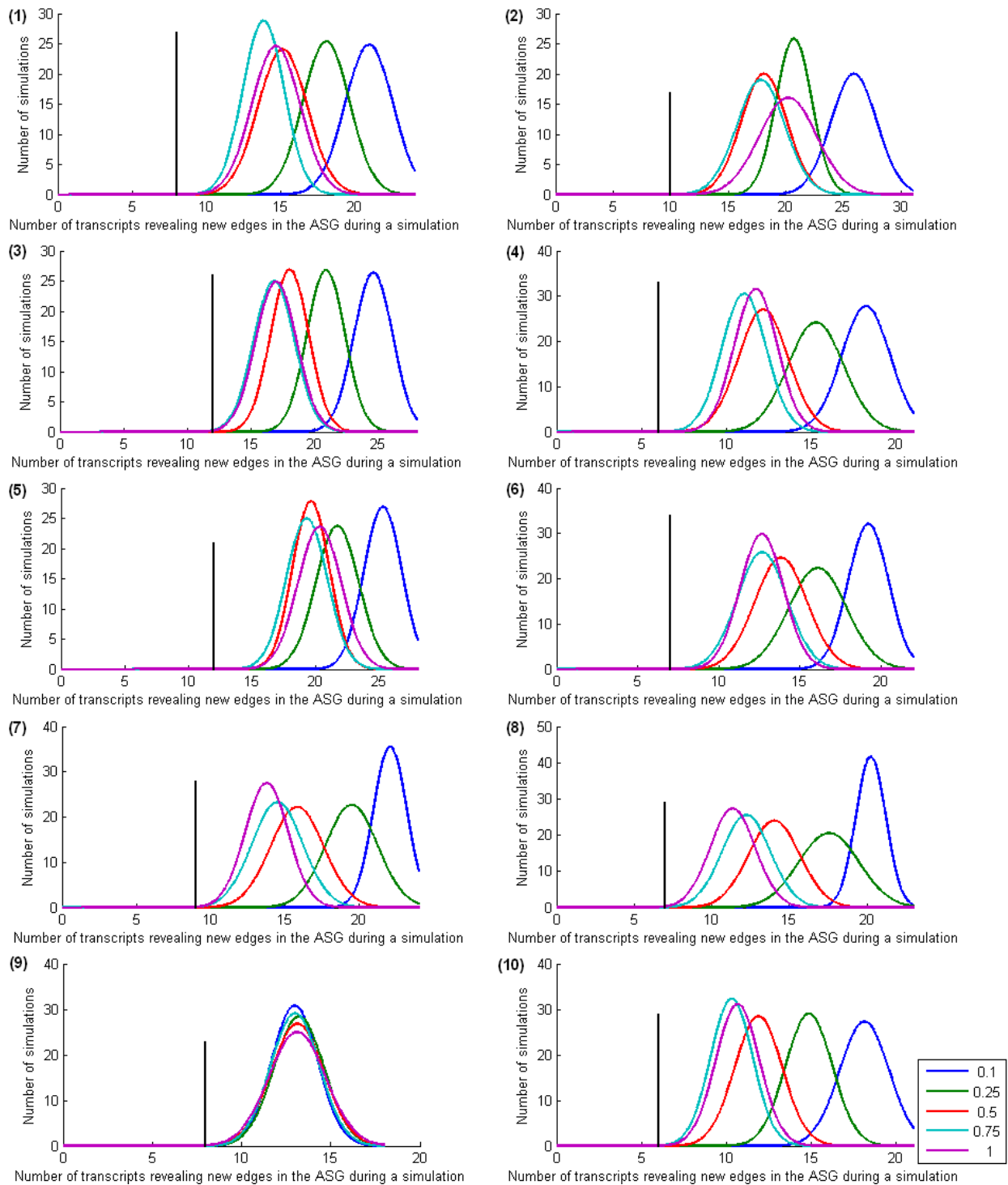


Figure 17: Normal approximations (plotted for visual convenience) to the distribution of the number of useful transcripts across 100 simulations, for  $q = 0.1, 0.25, 0.5, 0.75, 0.1$ . Minimal number of possible transcripts shown in black. Data are for human genes (1) ABCB5, (2) ACADVL, (3) CAPN3, (4) SORBS1, (5) EEF1D, (6) Y310.HUMAN, (7) BPAG1, (8) ATP9B, (9) SHMT2, (10) CLN3.

Leipzig *et al.* (2004) catalogues EST support for each gene. A simple way to calculate the probability for a particular splicing event at a splice site is to use the fraction of ESTs covering the splice site which reflect this splicing event. One must proceed with extreme care when using EST data in this way; EST

data is fraught with caveats (Modrek & Lee 2002). For example, ESTs exhibit a strong bias for the 3' end of a gene. Nevertheless, it is hoped that this statistical analysis will provide at least some indication of the relative applicability of the two models.

It would be of most interest to investigate genes complex enough to contain relatively long stretches of non-constitutive exon fragments, such as those shown in figures 2 and 9 (if they were very short then there would be almost nothing to test). For this reason we focus on the 89 human genes with ASGs complex enough to offer more than 5000 possible transcripts. Of these, 5 were selected by inspection to have associated ASGs that might reasonably be explained by model 2, and to have good EST support. Likelihood ratio statistics were computed for each applicable sequential pair of constitutive exons, using maximum likelihood parameter estimates (MLEs) determined by a simple EST-counting argument at each splice site (the probability of a splicing event was taken to be the fraction of ESTs supporting that event). Results are reported in table 3.

Gene	Ensembl ID	Exons ( $\Lambda, z$ ) $p$ -value
ABCB5	ENSG0000004846	6-12      17-26      26-41      41-47 (26.8, 4)    (26.9, 14)    (76.1, 57)    (24.6, 7) 0.000      0.020      0.047      0.001
CAPN3	ENSG00000092529	49-56 (11.4, 7) 0.121
SORBS1	ENSG00000095637	6-10      13-17      19-24 (0.5, 1)    (23.5, 2)    (25.4, 2) 0.474      0.000      0.0000
EEF1D	ENSG00000104529	1-13      13-31 (37.0, 37)    (113.8, 52) 0.470      0.000
CLN3	ENSG00000186015	4-15 (222.0, 31) 0.000

Table 3: Likelihood ratio tests of model 2 versus model 1 for 5 sample genes. For each applicable cluster of exons, the likelihood ratio statistic  $\Lambda$ , the degrees of freedom  $z$  and the corresponding  $p$ -value are given. Exon numbering is consistent with that of the Alternative Splicing Gallery (Leipzig *et al.* 2004).

## Discussion

We have introduced a mathematical framework to consider how to predict the nature of transcript generation in alternatively splicing genes. Given a gene and a sample of transcripts it can be used to simulate transcripts from its ASG in a quantitatively controlled way. The growth of a gene's reconstructed ASG can be compared to the theoretical minimal number of transcripts required to discover the complete ASG, and a concise mathematical way to evaluate this number has been provided. In the near future microarray data will be very useful to associate probabilities with the model, but in its absence we have demonstrated the use of the model by simulating transcripts from a sample of human genes. These were chosen from those known to be the most complex, and simulations performed for a range of example probabilities (as determined by our parameter  $q$ ). Figures 13-16 illustrate a number of features of these genes. The minimal number of transcripts required is consistently very low, ranging from 6 to 12 transcripts. The sample was chosen to contain a variety of features of the human genome (table 2), and this low range seems to be borne out by simulations on other genes. This suggests that with targeted sampling of transcripts from a gene (for example, by placing the cells in which it resides under different conditions in order to induce particular splicing events) its ASG can be reconstructed very quickly. Without targeted sampling, reconstruction of the ASG displays the trend one would expect of sampling

with replacement: rapid initial growth followed by very slow reconstruction of the final few edges. For an experimenter hoping to obtain 99% of the ASG for these genes the expected number of transcripts required is at least 25 or more in all cases; in one case this figure is as high as 120 ( $q = 1$ ). Other than assigning some preliminary values to questions regarding the rate of growth of the reconstructed ASG, these figures reveal two surprising things. First, there is considerable variation between the different genes. Second, the case  $q = 1$  is often not the most efficient way of reconstructing the ASG.

We would expect in general longer genes with larger ASGs to require more transcripts in the reconstruction of these ASGs, yet this is far from the case. CLN3, the smallest gene in terms of exon fragments and second smallest in terms of edges in the ASG, does require the fewest expected number of transcripts to reconstruct its ASG. Yet EEF1D, the third smallest gene both in terms of exon fragments and ASG edges, is the slowest gene to have its ASG reconstructed. BPAG1, by far the largest gene in the sample, expects a reasonable 40 transcripts to recover 99% of its ASG. This suggests that statistics such as those provided in table 2 provide a poor indication of how fast an ASG can be reconstructed.

Different genes also respond differently to variation in  $q$ . ATP9B displays almost no response to varying  $q$ , which can be explained by inspection of its ASG. Splicing events are all separated from each other by constitutive exons, so that events of low probability are never compounded to cause particular transcripts ever to have extremely low probability. Compare this with BPAG1 (figure 15(7)) which has a difference of 27 transcripts expected to recover 50% of the ASG as  $q$  varies from 0.1 to 1. (This feature might be explained as an artefact of the model: as  $q$  varies, some terminal exons can be reached with high probability, cutting off the transcript generator from large segments of the gene with high probability. The link seems to be confirmed by the positive correlation between this sort of unusual shape in the graph with the number of the gene’s terminal exons (table 2).) All ten genes show that the optimal value of  $q$  is nearer to 0.75 rather than 1 for the most efficient recovery of high portions of the ASG, yet one would expect  $q = 1$  to be optimal for a random directed acyclic graph. This possibly counter-intuitive result might be explained as follows. If at a splice site we do follow an edge out of the exon, we are necessarily skipping over at least one edge to be discovered—possibly more. Therefore there is a slight advantage in biasing mRNA translation slightly towards staying inside the exon at each splice site, where possible.

Figure 17 confirms that the optimal  $q$  is less than 1. In seven of the ten genes, the histogram for  $q = 0.75$  shows the lowest mean, suggesting that at this value of  $q$  fewer useful transcripts are needed to recover the ASG. All information about the ASG can be contained in fewer transcripts. On average the mean number of useful transcripts at  $q = 0.75$  is 1.65 times the minimal number of transcripts required. Comparing this value to experimentally obtained data will provide an important measure of coupling of exons—genes with closely coupled exons will require fewer useful transcripts, and the variance in this number will decrease (as the set of useful transcripts across different simulations becomes more uniform).

In table 3 we make some attempt to apply the models to real data, by testing the applicability of model 2 versus model 1. Using EST data to estimate parameters we can draw some initial conclusions about these models. Of the eleven exon clusters tested,  $p$ -values appear to fall neatly into either “significant” or “insignificant”. Taking  $\alpha = 0.05$  we would easily reject model 2 in seven cases and easily accept it in three. Only one test with  $p = 0.047$  straddles the borderline. We can infer that in many cases regulation of alternative splicing by a gene is too tight to allow the looser relationships suggested by model 2, hence reducing the possibility that there are a great many edges left undetected in the ASG. On the other hand, three of the exon clusters could easily have had their EST supports generated from model 2. If this is the case then it seems likely that there are a number of other edges in the ASG yet to be discovered, and that regulation of how these exons are ligated during splicing is not so tight. What might we draw from these inferences? Tentatively, that regulation of alternative splicing differs widely both between different genes and within genes, and hence that different ASGs vary widely in how well-characterized they are. We should be aware of bias problems in these inferences however. Not only do we suffer from the usual statistical bias towards the null hypothesis in regions of poor EST coverage, but EST databases additionally carry their own problems—experimental bias towards coverage of the 3’ end of a gene and human bias in curating useful transcripts (such as those associated with disease), to name two examples.

## Further work

Mathematical modelling of splicing graphs is in its relative infancy. There are many possible extensions to the work presented here, the most important perhaps being to assign experimentally derived probabilities to the models. Advances in microarray technology will facilitate this. There are many other ideas for continuing the work; a selection are suggested here.

- Modelling of the sort presented here could itself be used to test hypotheses about the transcripts found in databases, such as the importance of bias in transcript curation. What is the probability of observing the transcripts that have actually been seen?
- Related to the above is the idea of ‘usefulness’ of transcripts, which one would expect to be distributed differently in real databases. We can also ask: what is the expected number of transcripts needed before we observe  $x$  useful ones? How does the ratio of observed transcripts:total paths in the ASG vary as a function of ASG size? Figure 17 is annotated with the minimal number of useful transcripts; another related value which can be obtained from graph theory is the maximal number of useful transcripts. As mentioned above, classifying real transcripts by their usefulness could also have an important role to play in the analysis of phenomena such as the coupling of exons.
- The models themselves can be extended in a number of ways, by restricting further or loosening potential relationships between exons. Real genes seem to have intricate conditional relationships between different exons, a feature lost when we walk through an ASG hitting a sequence of independent probability distributions. Conditional probabilities in model 1 can be determined only by the topology of the ASG. In model 2 they are only determined inasmuch as we consider either  $p^{in}$  or  $p^{out}$  depending on the current state. A way to incorporate other conditional relationships into a transcript generator (and indeed the definition of the ASG itself) would be a valuable tool.
- Statistical testing of these and other models can also be extended, both in scope and rigour (again this will be aided by microarray data). Complementing the likelihood ratio approach suggested here, bootstrapping techniques to estimate the true size of genes’ ASGs could be worthwhile.
- Finally there are numerous other biological features for ASG modelling to encompass. These include:
  - Tissue-specific regulation, which confers a gene with two or more overlapping, weighted ASGs. How many transcripts must we sample before we can deduce the existence of this underlying structure?
  - Identifying functional versus non-functional transcripts.
  - Evolution of the ASG. Which splices have undergone selection?

## Acknowledgements

Thanks to Jotun Hein and Rune Lyngsø for their invaluable help and comments, and to the staff of the DTC for getting me into a position to do some actual research.

## Appendix

### A Example MATLAB code to input a gene to models 1 and 2

```
function [S, I, pstart, T, P, pin, pout] = ENSG00000004846(q,r,s)
% Function to return information on gene ENSG00000004846 for simulating from models 1 and 2, on inputs
% q,r,s.
% Dependent functions: mint, fASG_size, model1.

% ENSG00000004846. Modelled as [0, 511] with splice sites S at the following positions on this line (0
% will be reserved for the "supersource" as described in the text):
S = [10 20 21 30 31 40 41 50 51 60 70 71 80 90 100 101 110 120 121 130 131 140 141 150 151 160 161 ...
     170 180 190 200 210 220 221 230 231 240 241 250 260 270 271 280 289 290 300 309 310 320 321 ...
     330 332 340 350 360 370 371 380 381 390 400 410 411 420 421 430 432 440 450 451 460 470 471 ...
     480 481 490 491 500 501 510 511];
I = [10 80]; % Possible initiation splice sites I
pstart = [0.5 0.5]; % Probabilities p^start for each initiation
T = [161 511]; % Possible termination splice sites T
% Model 1 is given by the entries in P, the probability of jumping to other splice sites (if the row does
% not sum to 1 the remainder is the probability of staying 'in' this exon). Here, q is the measure of
% bias towards staying 'in', as described in the text.
P = zeros(length(splice_sites));
P(2,4) = q/2;
P(3,4) = 1;
P(5,6) = 1;
P(7,8) = 1;
P(9,10) = 1;
P(11,15) = q/2;
P(12,17) = 1;
P(14,15) = q/2;
P(16,18) = 1;
P(19,20) = 1;
P(21,22) = 1;
P(23,24) = 1;
P(25,26) = q/2; P(25,28) = 1-q/2;
P(29,31) = q/3; P(29,37) = q/3;
P(30,35) = q/3; P(30,40) = q/3;
P(32,40) = q/2;
P(33,40) = q/2;
P(34,40) = 1;
P(36,40) = 1;
P(38,39) = 1;
P(41,43) = q/3; P(41,45) = q/3;
P(42,48) = 1-q/2; P(42,51) = q/2;
P(44,54) = q/2;
P(46,53) = q/2;
P(47,49) = q/3; P(47,55) = q/3;
P(50,58) = q/3; P(50,60) = q/3; P(50,55) = 1-2*q/3;
P(52,54) = 1;
P(56,60) = q/2;
P(57,61) = 1;
P(59,62) = 1;
P(63,66) = q/2; P(63,64) = 1-q/2;
P(65,71) = q/2; P(65,68) = 1-q/2;
P(67,69) = 1;
P(70,72) = 1;
P(73,74) = 1;
P(75,76) = 1;
P(77,78) = 1;
P(79,80) = 1;

% Model 2: p^in and p^out values corresponding to the splice sites are:
pin = [0 0 0 1 0 1 0 1 0 1 0 0 0 0 r 0 r 1 0 1 0 1 0 1 0 r 0 1 0 0 r 0 0 0 r 0 r 0 r 1 0 0 r 0 r 0 0 ...
       r r 0 r 0 r r r 0 0 r 0 r r 1 0 r 0 r 0 r r 0 r 1 0 1 0 1 0 1 0 1 0];
pout = [0 s 1 0 1 0 1 0 1 0 s 1 0 s 0 1 0 0 1 0 1 0 1 0 1 0 s s 1 0 1 0 1 0 0 s 1 0 s 0 s s ...
        0 0 1 0 1 0 0 0 s 1 0 1 0 0 0 1 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1];
% In this example each splice site is assigned equal probabilities r and s for jumping 'in' and 'out'.
% They could be also altered manually.
```

## B MATLAB code to calculate the minimal number of transcripts required to cover the ASG

```

function [full_ASG minimal_transcripts] = mint(S, I, T, P)
% Function to return the minimal number of transcripts required to cover an ASG (and the full ASG itself,
% as an adjacency matrix). In this code the full ASG is determined using P from model 1.

% Reconstruct ASG. Represent the rows as each site along the gene, and columns are sites along the gene
% to which we can jump.
L = max(S); ASG = zeros(L); % For coding convenience [0,L] is cut into discrete chunks.
for i = 1:L
    if ismember(i,S)
        splice_address = find(S == i);
        next_splice_addresses = find(P(splice_address,:));
        next_splices = S(next_splice_addresses);
        ASG(i,next_splices) = 1;

        if (sum(P(splice_address,:)) < 1 & ~ismember(S(splice_address),T))
            % ie. if it is also possible to proceed to the next nucleotide
            ASG(i,i+1) = 1;
        end
    elseif (i > 1 && ASG(i-1,i) == 1)
        % "else if it is possible to arrive here from the previous nucleotide"
        % (this isn't a splice site)
        ASG(i,i+1) = 1;
    end
end
full_ASG = ASG;

% Now, tie up each source to a supersource and each sink to a supersink
sources_used = intersect(I, find(sum(ASG,2)));
sinks_used = intersect(T, find(sum(ASG,1)));
ASG_supersource_row = zeros(1,L+1);
ASG_supersource_row(sources_used+1) = 1;
ASG = [zeros(L,1) ASG];
ASG = [ASG_supersource_row; ASG];
ASG = [ASG; zeros(1,L+1)];
ASG_supersink_column = zeros(L+2,1);
ASG_supersink_column(sinks_used+1) = 1;
ASG = [ASG ASG_supersink_column];

% Construct an edge covering
edge_count = 0*ASG; % A count of the number of paths passing through each edge.
                % edge_count(i,j) corresponds to the edge ASG(i,j).
[x,y] = find(ASG); % Co-ordinates in ASG of all edges
[xx, yy] = find(edge_count == 0); % Thus far uncovered edges
uncovered_edges = intersect([x y], [xx yy], 'rows');

total_paths = 0;
while ~isempty(uncovered_edges)
    total_paths = total_paths + 1;

    % First, find an uncovered edge
    newpath = [uncovered_edges(1,1) uncovered_edges(1,2)]; % We use matrix indices, not splice address
    edge_count(uncovered_edges(1,1),uncovered_edges(1,2)) = ...
        edge_count(uncovered_edges(1,1),uncovered_edges(1,2)) + 1;

    % Extend to a sink
    sink_found = 0; % Flag for discovery of a sink
    while sink_found == 0
        next_splice_site = newpath(end,2);
        next_edges = find(ASG(next_splice_site,:)); % Edges leading from the end of our newpath
        if isempty(next_edges)
            sink_found = 1;
        else
            uncovered_next_edges = intersect(next_edges, find(edge_count(next_splice_site,:) == 0));
            if isempty(uncovered_next_edges)
                new_edge = next_edges(1);
            end
        end
    end
end

```

```

        else
            new_edge = uncovered_next_edges(1);
        end
        newpath = [newpath; next_splice_site new_edge];
        edge_count(next_splice_site,new_edge) = edge_count(next_splice_site,new_edge) + 1;
    end
end

% Extend to a source
source_found = 0; % Flag for discovery of a source
while source_found == 0
    % Find all upstream edges ending at the top of "newpath"
    if newpath(1,1) == 1
        source_found = 1;
    else
        prev_S = find(ASG(:,newpath(1,1))); % Previous sites with edges leading to the top of our
            % path.
        if isempty(prev_S)
            disp('Error: the following edge cannot be linked any farther upstream');
            newpath(1,:) = 1;
            break;
        else
            edge_count(prev_S(1),newpath(1,1)) = edge_count(prev_S(1),newpath(1,1)) + 1;
            newpath = [prev_S(1) newpath(1,1); newpath];
        end
    end
end

% Remove the edges in the new path from the master list of uncovered edges
[xx, yy] = find(edge_count == 0);
uncovered_edges = intersect([x yy], [xx yy], 'rows');
end

% Now trim the edge counts, based on redundant transcripts. These are signified by a passage from source
% to sink as described in the text. Use a breadth-first search (cf. Cormen et al. 1990, P470).
search_failed = 0; % Flag to determine whether we can construct a passage
while search_failed == 0
    colour = zeros(size(ASG,1),1); % 0 = white, 1 = grey, 2 = black.
    pi = 0*colour; % Upstream marker
    Q = [1]; % Queue. Start at supersource (entries correspond to rows of ASG).

    while ~isempty(Q)
        u = Q(1);
        % Find adjacent vertices to u that we can get to by the rules of passage construction.
        adj1 = find(edge_count(u,:) > 1);
        adj2 = find(edge_count(:,u));
        adj = union(adj1,adj2);
        if ~isempty(adj)
            for i = 1:length(adj)
                if colour(adj(i)) == 0
                    colour(adj(i)) = 1;
                    pi(adj(i)) = u;
                    Q = [Q; adj(i)];
                end
            end
        end
        Q(1) = [];
        colour(u) = 2;
    end

    % Check whether we reached supersink, and reconstruct path
    if colour(end) == 2
        path = [size(ASG,1)];
        while path(1) ~= 1
            path = [pi(path(1)); path];
        end
        total_paths = total_paths - 1; % Remove a path for each passage constructed

        % Update edge counts. For those traversed forwards, deduct one.
        % For those traversed backwards add 1.

```

```

    for i = 1:length(path)-1
        if path(i+1) > path(i)
            edge_count(path(i),path(i+1)) = edge_count(path(i),path(i+1)) - 1;
        elseif path(i+1) < path(i)
            edge_count(path(i+1),path(i)) = edge_count(path(i+1),path(i)) + 1;
        else
            disp('Error')
            break;
        end
    end
else
    search_failed = 1;
end
end
minimal_transcripts = total_paths;

```

## C MATLAB code to calculate the full size of an ASG

```

function full_ASG_size = fASG_size(S, I, T, P)
% Provide full size of an ASG of a gene represented by model 1.

edges = length(find(P)); % There is an edge for each non-zero entry in a splice site's row.
internals = find(sum(P,2) < 0.9999); % Splice sites which can continue 'in' satisfy this condition...
% (it should really be < 1 but MATLAB makes some numerical errors. This fixes it).
terminals = find(ismember(S,T)); % ...apart from those at which translation is terminated.
edge_bonus = length(setdiff(internals,terminals));

full_ASG_size = edges + edge_bonus;

```

## D MATLAB code to simulate from model 1 and output plots as in figures 10, 11, 12

```

function model1(S, I, T, P)
% Function to simulate transcript generation from a gene represented by model 1.
% Dependent on: mint, fASG_size.

% Find full ASG size
full_ASG_size = fASG_size(S, I, T, P);
% Find minimal transcripts required
[ASG_minimal_transcripts] = mint(S, I, T, P);

nS = length(S); % #splice sites
nI = length(I); % #initiation sites
L = max(S); % end of gene
nT = length(T); % #terminal exons
cum_p = cumsum(p,2);
cum_p(find(p == 0)) = 0;
cum_p = [zeros(nS,1) cum_p]; % Cumulative version of P, by rows

% Get inputs
simulations = input('Enter number of simulations: ');
sample_size = input('Enter number of transcripts to be generated: ');
plot_reconstructions = input('Plot reconstructions? (Y/N): ','s'); % For detailed plotting, or not

edges = zeros(sample_size,simulations);
% edges(i,j) = Size of reconstructed ASG after i samples, in jth simulation.
useful_transcripts = zeros(1,simulations);
% useful_transcripts(1) = number of transcripts which increased edge count during simulation 1.

for l = 1:simulations
    % For reconstructing minimal ASG, use ASGin and ASGout which are respectively a series of rows of
    % exons and jumps:
    % ASGin(i,:) = Start exon | End exon

```

```

% ASGout(i,:) = End last exon | Start new exon.
% All such pairs allows us to obtain the ASG.
ASGin = []; ASGout = [];
extra_edges = zeros(1,length(S)-1); % For use in calculating reconstructed ASG size.
% extra_edges(i) is an indicator for whether the edge (splice_site(i),splice_site(i+1)) is in the
% reconstructed ASG.
transcripts = {}; % transcripts{i} contains intervals contained within ith transcript
ytop = 0; % Plotting device to ensure same y-axes
if ismember(plot_reconstructions, 'Yy')
    figure; hold on;
end

% Choose initiation sites for each sample
cum_pss = [0 cumsum(pss)];
site_choice = rand(1,sample_size);
initiation_sites = 0*site_choice;
for k = 1:nI
    initiation_sites(find(site_choice < cum_pss(k+1) & site_choice >= cum_pss(k))) = k;
    % These are the sample addresses we are going to start from this
    % initiation site.
end

jump = zeros(nS,sample_size); % Indicator for changes of state. 1 = jump in, -1 = jump out.
jump_choices = rand(nS,sample_size);

% Generate transcripts in lth simulation
for n = 1:sample_size
    terminated = 0; % Flag for whether a terminal splice site has been reached
    curr_ss = find(S == ss(initiation_sites(n))); % Current splice site during walk
    jump(curr_ss,n) = 1; % Jump in at the beginning

    while terminated == 0
        next_jump = find(cum_p(curr_ss,:) >= jump_choices(curr_ss,n),1,'first');
        if isempty(next_jump) % If we do not jump just proceed to the next nucleotide
            curr_ss = curr_ss + 1;
        else % Indicate in jump matrix where we have jumped
            jump(curr_ss,n) = -1; jump(next_jump-1,n) = 1;
            curr_ss = next_jump-1;
        end

        if ismember(S(curr_ss),T) % Test whether we have reached a terminal splice site
            terminated = 1;
            jump(curr_ss,n) = -1;
        end
    end

    % Record this transcript in transcript{n} and update size of reconstructed ASG.
    transcripts{n} = [S(find(jump(:,n) == 1))' S(find(jump(:,n) == -1))'];
    ASGin = unique([ASGin; transcripts{n}], 'rows');
    jump1 = transcripts{n}(:,2);
    jump1(end) = [];
    jump2 = transcripts{n}(:,1);
    jump2(1) = [];
    ASGout = unique([ASGout; [jump1 jump2]], 'rows');
    edges(n,1) = size(ASGout,1); % #Jumps between exons

    % Calculate edge_bonus =====
    for j = 1:size(ASGin,1)
        new_edges = find(S >= ASGin(j,1) & S < ASGin(j,2));
        extra_edges(new_edges) = 1;
    end
    edge_bonus = sum(extra_edges);
    %=====
    edges(n,1) = edges(n,1) + edge_bonus; % (See fASG_size code).

    % Track whether transcript gave us anything new for plotting histogram
    if n == 1 || edges(n,1) > edges(n-1,1)
        useful_transcripts(1) = useful_transcripts(1) + 1;
    end
end

```

```

% Plot 0 (illustration of growth after 1, 5, 10, 20, 30, 40 transcripts)
if ismember(plot_reconstructions, 'Yy') & ismember(n,[1 5 10 20 30 40])
    frame_marker = find([1 5 10 20 30 40] == n);
    subplot(3,2,frame_marker); hold on;

    pairs = size(ASGIN,1);
    for j = 1:pairs
        plot(ASGIN(j,1:2)', [0;0], 'LineWidth', 10);
    end
    pairs = size(ASGOUT,1);
    for j = 1:pairs
        xcoord = ASGOUT(j,1):1/(ASGOUT(j,2)-ASGOUT(j,1)):ASGOUT(j,2);
        xterm = (xcoord - (ASGOUT(j,1) + ASGOUT(j,2))/2).^2;
        plot(xcoord, sqrt(((ASGOUT(j,2)-ASGOUT(j,1))/2)^2 - xterm), 'r');
        ytop = max(ytop, max(sqrt(((ASGOUT(j,2)-ASGOUT(j,1))/2)^2 - xterm)));
    end
    set(gca, 'YTick', [], 'XTick', []);
end
end

% Fix axes
if ismember(plot_reconstructions, 'Yy')
    for i = 1:6
        subplot(3,2,i);
        axis([0 L 0 ytop]);
    end
end
end

% Plot 1: growth of reconstructed ASG with each transcript
samples_required = sample_size*ones(1,simulations); % A device to choose convenient x-range in the plot

figure; hold on;
plot(0:sample_size, full_ASG_size*ones(sample_size+1,1), '-r', 'Linewidth', 2);
plot(minimal_transcripts*ones(full_ASG_size+2,1), 0:full_ASG_size+1, '-k', 'Linewidth', 2);
xlabel('Number of sampled transcripts');
ylabel('Number of edges in reconstructed ASG');

for l = 1:simulations
    plot(0:sample_size, [0; edges(:,l)], '-.');
    if ~isempty(find(edges(:,l) == full_ASG_size, 1, 'first'))
        samples_required(l) = find(edges(:,l) == full_ASG_size, 1, 'first');
    end
    hold all;
end
legend('Size of full ASG', 'Minimum possible number of transcripts required', 'Location', 'SouthEast');
axis([0 max(samples_required) 0 full_ASG_size+1]);

% Plot 2: summary statistics (mean +- 1 standard deviation)
figure; hold on;
plot(0:sample_size, full_ASG_size*ones(sample_size+1,1), '-r', 'Linewidth', 2);
plot(minimal_transcripts*ones(full_ASG_size+2,1), 0:full_ASG_size+1, '-k', 'Linewidth', 2);
xlabel('Number of sampled transcripts');
ylabel('Number of edges in reconstructed ASG');
av_edges = mean(edges, 2); % av_edges(x) = Average edges after x transcripts
std_edges = std(edges, 0, 2); % std_edges(x) = Sample standard deviation from mean after x transcripts
plot(0:sample_size, [0; av_edges], '-b', 'Linewidth', 2);
plot(0:sample_size, [0; av_edges+std_edges], '-.m', 0:sample_size, [0; av_edges-std_edges], ...
    '-.m', 'Linewidth', 1);
legend('Size of full ASG', 'Minimum possible number of transcripts required', ...
    ['Mean number of edges across ', num2str(simulations), ' simulations'], ...
    'Mean \pm 1 standard deviation', 'Location', 'SouthEast');
axis([0 max(samples_required) 0 full_ASG_size+1]);

% Plot 3: expected transcripts needed to cover different %s of ASG
ASG_portion = linspace(0, full_ASG_size); % A vector of each % in which we're interested
transcripts_needed = zeros(100, simulations); % transcripts_needed(i,j) = number of transcripts needed in
% simulation j to cover ASG_portion(i) of the ASG.
validity = find(ASG_portion <= min(edges(end,:)), 1, 'last');
% Expectation can be calculated only if all transcripts covered this much of the ASG.

```

```

% Validity is the highest such fraction.
for i = 1:validity
    for l = 1:simulations
        transcripts_needed(i,l) = find(edges(:,l) >= ASG_portion(i),1,'first');
    end
end
expected_transcripts_needed = mean(transcripts_needed,2);
if validity < full_ASG_size
    expected_transcripts_needed(validity+1:end) = -1; % Those values which cannot be calculated
end % (as not all simulations got this far).
figure; hold on;
plot(0:validity,[0; expected_transcripts_needed(1:validity)]);
xlabel('% of ASG');
ylabel('Expected number of transcripts required');

% Plot 4: histogram of number of distinct transcripts actually needed
figure; hold on;
y_upper = max(histc(useful_transcripts,0:max(useful_transcripts))); % Device to set the axes of the graph
plot(minimal_transcripts*ones(y_upper+2,1),0:y_upper+1,'-k','LineWidth',2);
hist(useful_transcripts,1:max(useful_transcripts));
h = findobj(gca,'Type','patch');
set(h,'EdgeColor','w');
% Also plot best Normal fit to the data
[muhat, sigmahat] = normfit(useful_transcripts);
Y = normpdf(0:0.1:max(useful_transcripts),muhat,sigmahat);
plot(0:0.1:max(useful_transcripts),simulations*Y,'r');
xlabel('Number of transcripts revealing new edges in the ASG during a simulation');
ylabel('Number of simulations');
legend('Minimal possible number of transcripts','Location','SouthWest');

```

## References

- Ensembl Genome Browser. <http://www.ensembl.org>
- Human And Vertebrate Analysis aNd Annotation (HAVANA). <http://www.sanger.ac.uk/HGP/havana/>
- D.L. Black. Mechanisms of alternative pre-messenger RNA splicing. *Annu. Rev. Biochem.*, **72**: 291–336 (2003).
- D.L. Black. A simple answer for a splicing conundrum. *P. Natl. Acad. Sci. USA*, **10**: 4927–4928 (2005).
- T.H. Cormen, C.H. Leiserson & R.L. Rivest. Introduction to Algorithms, MIT Press (1990).
- U. Gunthert, M. Hoffmann, W. Rudy, S. Reber, M. Zoller, I. Hausmann, S. Matzku, A. Wenzel, H. Ponta & P. Herrlich. A new variant of glycoprotein CD44 confers metastatic potential to rat carcinoma cells. *Cell*, **65**: 13–24 (1991).
- S. Heber, M. Alekseyev, S. Sze, H. Tang & P.A. Pevzner. Splicing graphs and EST assembly problem. *Bioinformatics*, **18**: 181–188 (2002).
- E.C. Ibrahim, T.D. Schaal, K.J. Hertel, R. Reed & T. Maniatis. Serine/arginine-rich protein-dependent suppression of exon skipping by exonic splicing enhancers. *P. Natl. Acad. Sci. USA*, **10**: 5002–5007 (2005).
- J.M. Johnson, J. Castle, P. Garrett-Engele, Z. Kan, P.M. Loerch, C.D. Armour, R. Santos, E.E. Schadt, R. Stoughton & D.D. Shoemaker. Genome-wide survey of human alternative pre-mRNA splicing with exon junction microarrays. *Science*, **302**: 2141–2144 (2003).
- A.R. Kornblihtt, M. De La Mata, J.P. Fededa, M.J. Muñoz, G. Nogués. Multiple links between transcription and splicing. *RNA*, **10**: 1489–1498 (2004).
- C. Lee, L. Atanelov, B. Modrek & Y. Xing. ASAP: the Alternative Splicing Annotation Project. *Nucleic Acids Res.*, **31**: 101–105 (2003).
- C. Lee & Q. Wang. Bioinformatics analysis of alternative splicing. *Brief. Bioinform.*, **6**: 23–33 (2005).
- J. Leipzig, P. Pevzner & S. Heber. The Alternative Splicing Gallery (ASG): bridging the gap between genome and transcriptome. *Nucleic Acids Res.*, **32**: 3977–3983 (2004).
- W.N. Li, S.M. Reddy & S. Sahni. On path selection in combinatorial logic circuits. *IEEE T. Comput. Aid. D.*, **8**: 56–63 (1989).
- A.J. Lopez. Alternative splicing of pre-mRNA: developmental consequences and mechanisms of regulation. *Annu. Rev. Genet.*, **32**: 279–305 (1998).
- B. Modrek & C. Lee. A genomic view of alternative splicing. *Nature Genet.*, **30**: 13–19 (2002).
- G.C. Roberts & C.W.J. Smith. Alternative splicing: combinatorial output from the genome. *Curr. Opin. Chem. Biol.*, **6**: 375–383 (2002).
- Q. Pan, O. Shai, C. Misquitta, W. Zhang, A.L. Saltzman, N. Mohammad, T. Babak, H. Siu, T.R. Hughes, Q.D. Morris, B.J. Frey & B.J. Blencowe. Revealing global regulatory features of mammalian alternative splicing using a quantitative microarray platform. *Mol. Cell*, **16**: 929–941 (2004).