Efficient RNA Isoform Identification and Quantification from RNA-Seq Data with Network Flows

Elsa Bernard, Laurent Jacob, Julien Mairal, Jean-Philippe Vert

Efficient RNA Isoform Identification and Quantification from RNA-Seq Data with Network Flows

Elsa Bernard[1,2,3], Laurent Jacob[4], Julien Mairal[5], Jean-Philippe Vert[1,2,3]

March 21, 2013

Abstract

Several state-of-the-art methods for isoform identification and quantification are based on sparse probabilistic models, such as Lasso regression. However, explicitly listing the – possibly exponentially – large set of candidate transcripts is intractable for genes with many exons. For this reason, existing approaches using sparse models are either restricted to genes with few exons, or only run the regression algorithm on a small set of pre-selected isoforms.

We introduce in this paper a new technique, called FlipFlop, based on network flow optimization which can efficiently tackle the sparse estimation problem on the full set of candidate isoforms. By removing the need of preselection step, we obtain better isoform identification while keeping a low computational cost. Experiments with synthetic and real single-end RNA-Seq data confirm that our approach is more accurate than alternatives methods and one of the fastest available.

1 Introduction

Over the past decade, quantitation of mRNA molecules in a cell population has become a popular approach to study the effect of several factors on cellular activity. Typical applications include the detection of genes whose expression varies between two or more populations of samples (differential analysis); classification of samples based on gene expression (van't Veer *et al.*, 2002); and clustering, which consists of identifying a grouping structure in a sample set (Perou *et al.*, 2000). While probe-based DNA microarray technologies only allow to quantitate mRNA molecules whose sequence is known in advance, the recent development of deep sequencing has removed this restriction. More specifically, RNA-Seq technologies (Mortazavi *et al.*, 2008) allow the sequencing of cDNA molecules obtained by reverse transcription of RNA molecules present in the cell. Consequently, any transcript can be sequenced and therefore quantitated, even when its sequence is not available a priori to design a specific probe. In addition to facilitating the study of non-coding parts of known genomes and organisms whose genome has not been sequenced (Mortazavi *et al.*, 2010), this facilitates the quantitation of alternatively spliced genes. Genes in eukaryote cells indeed contain a succession of exon and intron sequences. Gene transcription results in a pre-mRNA molecule from which most introns are removed and some exons are retained during a processing step called RNA splicing. It is estimated that more than 95% of multiexonic genes are subject to alternative splicing (Pan *et al.*, 2008) : the set of exons — and possibly introns — retained during splicing can vary, resulting in different versions of the mRNA molecule for the same gene, referred to as

[1]Centre for Computational Biology – CBIO, Mines ParisTech, Fontainebleau, France, [2]Institut Curie, Paris, France, [3]INSERM U900, Paris, France, [4]LBBE, Lyon, France, [5]LEAR Project-Team, INRIA Grenoble - Rhône Alpes, France

transcripts or isoforms. Identification and quantification of isoforms present in a sample is of outmost interest because different isoforms can later be translated as different proteins. Detection of isoforms whose presence or quantity varies between samples may lead to new biomarkers and highlight novel biological processes invisible at the gene level.

Sequencing technologies are well suited to transcript quantitation as the read density observed along the different exons of a gene provide information on which alternatively spliced mRNAs were expressed in the sample, and in which proportions. Since the read length is typically smaller than the mRNA molecule of a transcript, identifying and quantifying the transcripts is however non-trivial : an observed read mapping to a particular exon may come from an mRNA molecule of any transcript containing this exon. Some methods consider that the set (Jiang and Wong, 2009) of expressed isoforms or a candidate superset (Huang *et al.*, 2012; Xing *et al.*, 2006) is known in advance, in which case the only problem is to estimate their expression. However in practice little is known about the possible isoforms of genes, and restricting oneself to isoforms which have been described in the literature may lead to missing new ones.

Two main paradigms have been used so far to estimate expression at the transcript level while allowing de novo transcript discovery. On the one hand, the widely used Cufflinks software package (Trapnell *et al.*, 2010) proceeds in two separate steps to identify expressed isoforms and estimate their abundances. The list of alternatively spliced transcripts is estimated by building a small set of isoforms containing all observed exons and exon junctions. The expression of each transcript is then quantified in a separate step by likelihood maximization given the list of transcripts. Identification and quantification are therefore done independently. On the other hand, a second family of methods (Xia *et al.*, 2011; Li *et al.*, 2011b; Bohnert and Rätsch, 2010; Li *et al.*, 2011a; Mezlini *et al.*, 2013) jointly estimates the set of transcripts and their expression using a penalized likelihood approach. The likelihood models the expression of all possible transcripts, possibly after some filtering, and the penalty induces sparsity on the set of selected transcripts.

The two step approach of Cufflinks (Trapnell *et al.*, 2010) results in a reasonably fast method but does not exploit the observed read density along the gene, which can be a valuable information to identify the set of transcripts. This is indeed a conclusion drawn experimentally using methods from the more computationally expensive second paradigm (see Xia *et al.*, 2011; Li *et al.*, 2011b; Bohnert and Rätsch, 2010; Li *et al.*, 2011a; Mezlini *et al.*, 2013). To summarize, the first paradigm is fast but can be less statistically powerful than the second one in some cases, and the second paradigm should always be powerful but becomes untractable for genes with many exons. The contribution of this paper is to allow methods of the second family to run efficiently without prefiltering the set of isoform candidates, although they solve a non-smooth optimization problem over an exponential number of variables. To do so, we show that the penalized likelihood maximization can be reformulated as a convex cost network flow problem, which can be solved efficiently (Ahuja *et al.*, 1993; Bertsekas, 1998; Mairal and Yu, 2012).

The paper is organized as follows. Section 2 introduces the statistical model and the penalized likelihood approach. Section 3 describes our method dubbed FlipFlop (Fast Lasso-based Isoform Prediction as a FLOw Problem) for solving the regularized maximum likelihood problem. Section 4 empirically compares existing approaches with the one we introduce on simulated and real sequencing data. Our experiments show that the proposed approach leads to higher accuracy in isoform discovery than methods which treat discovery and abundance estimation as two separate steps, and that it does so much faster than the methods which explicitly list the candidate isoforms. A discussion is given in Section 5.

2 Approach

Our approach to isoform deconvolution from RNA-Seq data consists of fitting a sparse probabilistic model, like several existing methods including rQuant (Bohnert and Rätsch, 2010), NSMAP (Xia *et al.*, 2011), IsoLasso (Li *et al.*, 2011b), SLIDE (Li *et al.*, 2011a) or iReckon (Mezlini *et al.*, 2013). The reads from RNA-Seq data are modeled as a linear combination of isoforms expressions that are estimated using the maximum likelihood principle. Because the number of candidate isoforms grows exponentially with the number of exons, the above methods are either computationally expensive for genes with many exons (such as NSMAP or SLIDE), or include a prefiltering step to reduce the number of candidates.

The main novelty of our paper is to tackle the sparse estimation problem efficiently *without prefiltering*. More precisely, we show in the methodological section that the corresponding penalized maximum likelihood estimator can be computed in polynomial time despite the exponential number of candidate transcripts. The key is the use of a non-trivial optimization technique based on the concept of flow in a graph (Ahuja *et al.*, 1993; Mairal and Yu, 2012).

2.1 Statistical Model

We consider the same statistical model as NSMAP, which was originally introduced by Jiang and Wong (2009) for estimating isoform expression for a known set of expressed transcripts. Given a gene of interest, we assume that the list of its n exons is known, and that the reads of the RNA-Seq experiments have been mapped to a reference genome.

We summarize the read information by the counts $y_1, \ldots, y_q$ of reads falling in q bins, where each bin is either an exon or a junction between two exons. A read is counted in a junction between two exons if it starts in one of them and ends in the other. For the purpose of our work, an exon can either be defined by read alignment softwares as a cluster of reads, or from a pre-defined annotation such as the one provided by the UCSC genome browser[1]. In the latter case, exons with alternative 5' donor and 3' acceptor sites are considered as two separate exons. For alternative 5' donor sites, the exon is broken down as one exon ending at the first 5' donor site, and another one start at this same point and ending at the second 5' donor site (similarly for exons with 3' acceptor sites).

Formally, reads are modeled as random variables whose mean is the sum of isoform abundances. We consider in our model all possible candidate isoforms consisting of a sequence of exons linked by junctions. We denote by U the $m \times q$ binary matrix defined as $U_{ji} = 1$ if bin i is present in isoform j and 0 otherwise, and by $\theta_j \in \mathbb{R}_+$ the expression of isoform j (the expected number of reads per base in isoform j). Thus, $\sum_{j=1}^{m} U_{ji}\theta_j$ represents the sum of expressions of all isoforms involving bin i. We expect the observed count for bin i to be distributed around this value times the length of the bin l_i. More specifically, we assume that the read count y_i follows a Poisson distribution with parameter $\delta_i = l_i \sum_{j=1}^{m} U_{ji}\theta_j$. This yields for a vector $\theta = [\theta_j]_{j=1}^{m}$ in $\mathbb{R}_+^m$ the log-likelihood

$$\mathcal{L}(\theta) = \sum_{i=1}^{q} \left[-\delta_i + y_i \log \delta_i - \log(y_i!)\right], \tag{1}$$

where the δ_i's depend linearly on θ. Note that for exons, l_i is simply the exon length whereas for junctions it is computed as $\min(l_{\text{left}}$, read length - 1) + $\min(l_{\text{right}}$, read length - 1) where l_{left} and l_{right} are the lengths of the left and right exons of the junction respectively.

Maximizing the likelihood (1) allows to quantify the relative abundance of each transcript when the model only includes the list of "true" isoforms present in the sample (Jiang and Wong, 2009).

[1] `http://genome.ucsc.edu/`

Since this list is unknown a priori, we present in the next section the sparse estimation approach that can jointly quantify and identify the transcripts using all candidate isoforms (Xia *et al.*, 2011).

2.2 Isoform Detection by Sparse Estimation

Since we do not assume that the list of expressed isoforms — *i.e.* such that $\theta_j \neq 0$ — is known in advance, we endow θ with an exponential prior $\theta_j \overset{\text{iid}}{\sim} E(\lambda)$ and maximize over all candidate isoforms the resulting posterior likelihood, leading to the estimator

$$\hat{\theta}_\lambda = \underset{\theta \in \mathbb{R}_+^m}{\arg\min} [-\mathcal{L}(\theta) + \lambda \| \theta \|_1] \ , \tag{2}$$

where λ acts as a regularization parameter, and the ℓ_1-norm is defined as $\| \theta \|_1 = \sum_{j=1}^m |\theta_j|$. It is well-known that the ℓ_1-norm penalty and the non-negativity constraint have a sparsity-inducing effect — that is, lead to estimators $\hat{\theta}_\lambda$ that contain many zeroes (Tibshirani, 1996). The parameter λ controls the number of non-zero elements in the solution $\hat{\theta}_\lambda$, *i.e.*, of selected isoforms, with larger λ corresponding to fewer isoforms.

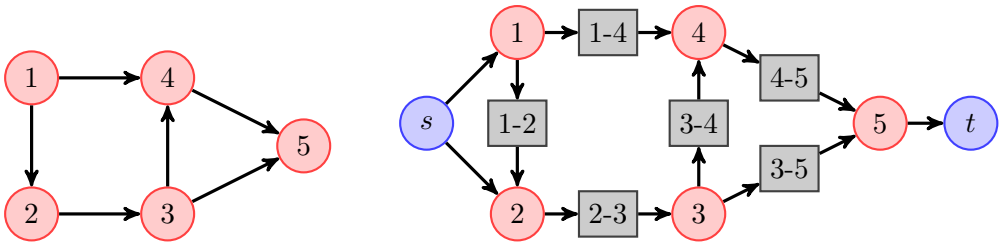
Mezlini *et al.* (2013) claim that the ℓ_1-penalty is inappropriate for isoform selection, a claim we disagree with. As they note, the sum of true abundances in RPKM weighted by isoform lengths is by definition the true proportion of reads coming from the gene times 10^9. They conclude that penalizing by $\sum_j \theta_j$ has little effect on the estimate. However, the sum of the *estimator* $\hat{\theta}_\lambda$ weighted by isoform lengths has no reason to be equal to the *observed* number of reads mapping to the gene. There are several causes for that: model inadequacy, various noise sources, finite sample size, and bias of the estimator. Penalizing this sum therefore modifies the sparsity level of $\hat{\theta}_\lambda$ as observed in our and other's experiments (Bohnert and Rätsch, 2010; Xia *et al.*, 2011; Li *et al.*, 2011a,b).

Note also that (2) is exactly the problem that NSMAP (Xia *et al.*, 2011) tries to solve, while rQuant (Bohnert and Rätsch, 2010), IsoLasso (Li *et al.*, 2011b) and SLIDE (Li *et al.*, 2011a) solve a similar problem where the likelihood is a simpler quadratic function, corresponding to a Gaussian model for the read counts. A difficulty with these approaches is that the dimension m of the optimization problem (2) grows exponentially in n leading to computational intractability when n is large. For example, Li *et al.* (2011a) restrict themselves to experiments with genes having less than 10 exons, due to high computational cost for larger genes. Xia *et al.* (2011) restrict themselves to genes with less than 80 exons, but only consider candidates with transcription start/polyadenylation sites (TSS/PAS) pairs already observed in annotations, and which involve more than half of the exons of the gene. Other approaches such as IsoLasso include a filtering step to reduce the number of isoforms, similarly as Cufflinks does — in the case of single end reads, their set of candidates is the set of isoforms returned by Cufflinks. As pointed out in Section 1, this filtering may lead to a loss of power in isoform detection, because it disregards the read density information when constructing the set of candidates.

In the next section we show that, surprisingly, problem (2) can be solved efficiently without prefiltering the isoforms using network flow optimization (Ahuja *et al.*, 1993; Mairal and Yu, 2012).

3 Methods

Our first step is to reformulate isoform detection as an optimization problem over the paths of a directed acyclic graph (DAG), similar to the classical splicing graph (Heber *et al.*, 2002).



(a) Splicing graph for a gene with 5 exons.

(b) Graph G' with junctions, source s and sink t nodes.

Figure 1: Illustration of the graph construction on a gene with 5 exons. The original splicing graph is represented in (a). The 5 exons are represented as red circles and an arrow between two nodes indicates a junction. The graph G' in (b) includes junction nodes denoted by gray squares, as well as source s and sink t nodes. When an exon is linked to the source, it means that it is considered as a starting exon. Conversely, an exon linked to the sink is a stopping exon. There is a one-to-one correspondence between (s,t)-paths in G' (paths starting at s and ending at t) and isoform candidates. For example, the path $(s, 1, 1\text{-}4, 4, 4\text{-}5, 5, t)$ corresponds to isoform 1-4-5.

3.1 Isoform Detection as a Path Selection Problem

Remember that a graph $G = (V, E)$ is composed of a finite set of vertices V and edges $E \subseteq V \times V$. A path is a sequence of vertices $v_1, \ldots, v_k \in V$ such that (v_i, v_{i+1}) is an arc in E for all indices $1 \leq i < k$. A graph is a DAG if it contains no path $(v_1, \ldots, v_k)$ with $v_1 = v_k$. In other words, the graph does not contain any cycle.

For a given gene, we construct the graph $G = (V, E)$ whose vertex set V is the set of bins, *i.e.*, the set of exons and exon-exon junctions, and whose edges connect exon bins to junction bins according to the following rule: connect exon e to the junction e-e', and connect the junction e-e' to the exon e'. Note that the set of exons $(e_1, \ldots, e_n)$ is ordered, and that junctions only connect exons in strictly increasing order, *i.e.*, e_i-e_j is a junction only if $i < j$. The resulting graph G is therefore a DAG, since any path must move along vertices with strictly increasing exons. This graph is similar to the splicing graph (Heber *et al.*, 2002), whose vertices are single exons and edges are exon-exon junctions. For reasons that will become clear in the next section, we simply replace each edge of the splicing graph by an exon-exon junction vertex.

We also consider two new vertices s and t respectively dubbed *source* and *sink*, which are used to specify starting (TSS) and stopping exons (PAS). This leads to the definition of an extended graph $G' = (V', E')$ with $V' = V \cup \{s, t\}$ and E' is obtained by adding to E all edges of the form (s, e) where $e \in V$ is a bin corresponding to a starting exon, and (e, t) where $e \in V$ is a stopping exon. This graph construction is illustrated in Figure 1.

Let us denote by $\mathcal{P}$ the set of paths in G' starting from s and ending at t, which are called (s,t)-paths. By construction, any path in $\mathcal{P}$ is a sequence $(s, e_{i_1}, e_{i_1}\text{-}e_{i_2}, e_{i_2}, \ldots, e_{i_{k-1}}\text{-}e_{i_k}, e_{i_k}, t)$. It corresponds to a candidate isoform $(e_{i_1}, e_{i_2}, \ldots, e_{i_k})$, in the sense that it passes exactly on the bins contained in the isoform, e_{i_1} is a starting exon, and e_{i_k} is a stopping exon. Conversely, any candidate isoform $(e_{i_1}, e_{i_2}, \ldots, e_{i_k})$ corresponds to the path $(s, e_{i_1}, e_{i_1}\text{-}e_{i_2}, e_{i_2}, \ldots, e_{i_{k-1}}\text{-}e_{i_k}, e_{i_k}, t)$ in $\mathcal{P}$. We therefore have a one-to-one mapping between the set of candidate isoforms, on the one hand, and $\mathcal{P}$, on the other hand. Based on this one-to-one mapping, we can reformulate the penalized maximum likelihood problem (1)-(2) as follows: we want to find nonnegative weights θ_p

for each path $p \in \mathcal{P}$ which minimize:

$$\sum_{v \in V} [\delta_v - y_v \log \delta_v] + \lambda \sum_{p \in \mathcal{P}} \theta_p \quad \text{with} \quad \delta_v = \Big(l_v \sum_{p \in \mathcal{P}: p \ni v} \theta_p \Big), \tag{3}$$

where the sum $\sum_{p \in \mathcal{P}} \theta_p$ is equal to the ℓ_1-norm $\|\theta\|_1$ since the entries of θ are non-negative. Note that we have removed the constant term $\log(y_v!)$ from the log likelihood since it does not play a role in the optimization. This reformulation is therefore a path selection (finding which θ_p are non-zero) and quantification problem over G'. The next section shows how (3) can further be written as a flow problem, *i.e.*, technically a constrained optimization problem over the edges of the graph rather than the set of paths in $\mathcal{P}$. A computationally feasible approach can then be devised to solve (3) efficiently, following Mairal and Yu (2012).

3.2 Optimization with Network Flows

A *flow* f on G' is defined as a non-negative function on arcs $[f_{uv}]_{(u,v) \in E'}$ that satisfies conservation constraints: the sum of incoming flow at a vertex is equal to the sum of outgoing flow except for the source s and the sink t. Such conservation property leads to a physical interpretation about flows as quantities circulating in the network, for instance, water in a pipe network or electrons in a circuit board. The source node s injects into the network some units of flow, which move along the arcs before reaching the sink t.

For example, given a path $p \in \mathcal{P}$ and a non-negative number θ_p, we can make a flow by setting $f_{uv} = \theta_p$ when u and v are two consecutive vertices along the path p, and $f_{uv} = 0$ otherwise. This corresponds to sending θ_p units of flows from s to t along the path p. Such simple flows are called (s,t)-*path flows*. More interestingly, if we have a set of non-negative weights $\theta \in \mathbb{R}_+^{|\mathcal{P}|}$ associated to all paths in $\mathcal{P}$, then we can form a more complex flow by superimposing all (s,t)-path flows according to

$$f_{uv} = \sum_{p \in \mathcal{P}: p \ni (u,v)} \theta_p, \tag{4}$$

where $(u,v) \in p$ means that u and v are consecutive nodes on p.

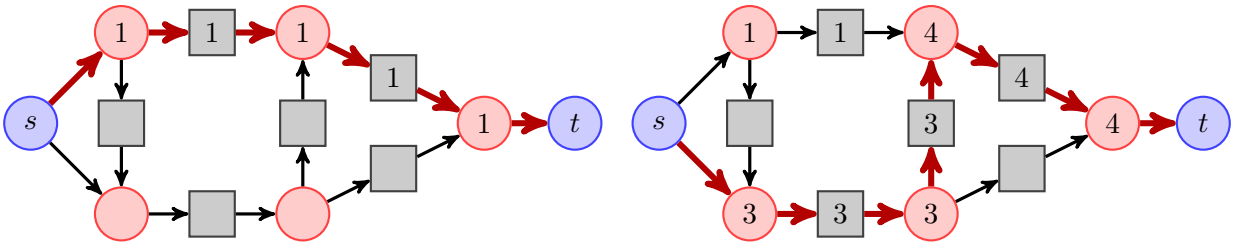
While (4) shows how to make a complex flow from simple ones, a converse exists, known as the *flow decomposition theorem* (see, e.g., Ahuja *et al.*, 1993). It says that for any DAG, every flow vector can always be decomposed into a sum of (s,t)-path flows. In other words, given a flow $[f_{uv}]_{(u,v) \in E'}$, there exists a vector θ in $\mathbb{R}_+^{|\mathcal{P}|}$ such that (4) holds. Moreover, there exists linear-time algorithms to perform this decomposition (Ahuja *et al.*, 1993). As illustrated in Figure 2, this leads to a flow interpretation for isoforms.

We now have all the tools in hand to turn (3) into a flow problem by following Mairal and Yu (2012). Given a flow $f = [f_{uv}]_{(u,v) \in E'}$, let us define the amount of flow incoming to a node v in V' as $f_v \triangleq \sum_{u \in V': (u,v) \in E'} f_{uv}$. Given a vector $\theta \in \mathbb{R}_+^{|\mathcal{P}|}$ associated to f by the flow decomposition theorem, *i.e.*, such that (4) holds, we remark that $f_v = \sum_{p \in \mathcal{P}: p \ni v} \theta_p$ and that $f_t = \sum_{p \in \mathcal{P}} \theta_p$. Therefore, problem (3) can be equivalently rewritten as:

$$\min_{f \in \mathcal{F}} \sum_{v \in V} [\delta_v - y_v \log \delta_v] + \lambda f_t \quad \text{with} \quad \delta_v = l_v f_v \,. \tag{5}$$

where $\mathcal{F}$ denotes the set of possible flows. Once a solution $f^\star$ of (5) is found, a solution $\theta^\star$ of (3) can be recovered by decomposing $f^\star$ into (s,t)-path flows, as discussed in the next section.

The use of network flows has two consequences. First, (5) involves a polynomial number of variables, as many as arcs in the graph, whereas this number was exponential in (3). Second,



(a) Reads at every node corresponding to one isoform. (b) Reads at every node after adding another isoform.

Figure 2: Flow interpretation of isoforms using the same graph as in Figure 1. For simplification purposes, the length of the different bins are assumed to be equal. In (a), one unit of flow is carried along the path in red, corresponding to an isoform with abundance 1. In (b), another isoform with abundance 3 is added, yielding additional read counts at every node.

problem (5) falls into the class of *convex cost flow* problems (Ahuja *et al.*, 1993), for which efficient algorithms exist.[2] In our experiments, we implemented a variant of the scaling push-relabel algorithm (Goldberg, 1997), which also appears under the name of ε-relaxation method (Bertsekas, 1998). Note that the approach can be generalized to any concave likelihood function, including the Gaussian model used by IsoLasso and SLIDE.

We remark that network flows have been used in several occasions in bioinformatics. For example, the terminology of "flow" for RNA-Seq data appears in Montgomery *et al.* (2010); Singh *et al.* (2011). The context of these two works is significantly different than ours since they neither perform isoform detection, nor use any network flow algorithm. The work closest to ours in terms of optimization is probably the genome assembly technique of Medvedev and Brudno (2009), who solve minimum cost flow problems to find a genome maximizing a read-count likelihood. It however neither involves RNA-Seq data, nor a similar type of graph as ours.

3.3 Flow Decomposition

We have seen that after solving (5) we need to decompose $f^\star$ into (s,t)-path flows to obtain a solution $\theta^\star$ of (2). As illustrated in Figure 2, this corresponds to finding the two isoforms from 2(b). Whereas the decomposition might not be ambiguous when $f^\star$ is a sum of few (s,t)-path flows, it is not unique in general. Our approach to flow decomposition consists of finding an (s,t)-path carrying the maximum amount of flow (equivalently finding an isoform with maximum expression), removing its contribution from the flow, and repeating until convergence. We remark that finding (s,t)-path flows according to this criterion can be done efficiently using dynamic programming, similarly as for finding a shortest path in a directed acyclic graph (Ahuja *et al.*, 1993).

3.4 Model Selection

The last problem we need to solve is model selection: even if we know how to solve (2) efficiently, we need to choose a regularization parameter λ. For large values of λ, (2) yields solutions involving few expressed isoforms. As we decrease λ, more isoforms have a non-zero estimated expression θ_j, leading to a better data fit but also leading to a more complex model. A classical way of balancing

[2]The function (5) can be decomposed into costs $C_v(f_v)$ over vertices v. The general convex cost flow objective function is usually presented as a sum of costs $C_{uv}(f_{uv})$ over arcs (u,v). It is however easy to show that costs over vertices can be reduced to costs over arcs by a simple network transformation (see Ahuja *et al.*, 1993, Section 2.4). Note that all arcs have zero lower capacities and infinite upper capacities.

fit and model complexity is to use likelihood ratio tests. Xia *et al.* (2011) chose this approach, but we found the log likelihood ratio statistics to be empirically poorly calibrated due to the typically small number of samples units — exons — and the non-independence of the observed read counts. We choose a related approach, which we found better behaved, and select the model having the largest BIC criterion (Schwarz, 1978). An alternative approach taken by Li *et al.* (2011a) would be to use stability selection (Meinshausen and Bühlmann, 2010).

4 Results

We now compare our proposed method FlipFlop to Cufflinks (Trapnell *et al.*, 2010) version 2.0.0, Isolasso (Li *et al.*, 2011b) version 2.6.1, NSMAP (Xia *et al.*, 2011) and SLIDE (Li *et al.*, 2011a) on both simulated and real data. We did not include iReckon (Mezlini *et al.*, 2013) in the comparison as their software only works on paired-end data and ours on single-end. We do not include rQuant (Bohnert and Rätsch, 2010) either, as it is intended to select transcripts out of a given annotation rather than build a list of all potential transcripts and select from this list.

All experiments were run on a desktop computer on a single core of an Intel Xeon CPU X5460 3.16Ghz with 16Gb of RAM. In each case, reads are aligned to a reference genome using TopHat (Trapnell *et al.*, 2009) version 2.0.6, and the constructed alignment files are used as input to the methods we compare. IsoLasso, Cufflinks and FlipFlop only use these aligned reads as input, and estimate their exon boundaries and TSS/PAS from read density. SLIDE and NSMAP additionally require exon boundaries as input.

All softwares are used in their default mode, except that SLIDE is run using its stability selection mode to estimate the number of isoforms. We also bring a small modification to NSMAP: it still restricts the TSS (resp. PAS) of its candidates to sites that are observed to be TSS (resp. PAS) in the annotation, but it does not restrict the TSS/PAS couples of its candidates to couples that are in the annotation. In practice, unknown isoforms with alternative TSS/PAS couples may be expressed. Therefore, using this information and validating on a set of transcripts coming from the annotation may lead to over-optimistic conclusions.

4.1 Simulated Human RNA-Seq Data

Since little is known about the true set of isoforms expressed in real data, we start our experimental validation with a set of simulations. We use the RNASeqReadSimulator software[3] to generate single-end reads from the annotated human transcripts available in the UCSC genome browser (hg19). We restrict ourselves to the 1137 multi-exons genes on the positive strand for chromosome 1, corresponding to 3553 expressed transcripts. Each experiment comes with 1 million reads of 75 base pair length, aligned against the human genome also available in the UCSC genome browser.

We follow the protocol of Isolasso (Li *et al.*, 2011b) and consider that a transcript from the annotation has been detected by a method if it predicts a transcript that (i) includes the same set of exons; and such that (ii) all internal boundary coordinates (*i.e.*, all the exon coordinates except the beginning of the first exon and the end of the last exon) are identical. The objective for each method is to recover a large proportion of transcripts that were used for read generation — high recall — without detecting too many transcripts that were not used to generate the reads — high precision.

Figure 3 shows the precision and recall of the compared methods. Since we expect the difficulty of the deconvolution problem to increase with the number of transcripts of the gene, we stratify the

[3] `http://alumni.cs.ucr.edu/~liw/rnaseqreadsimulator.html`

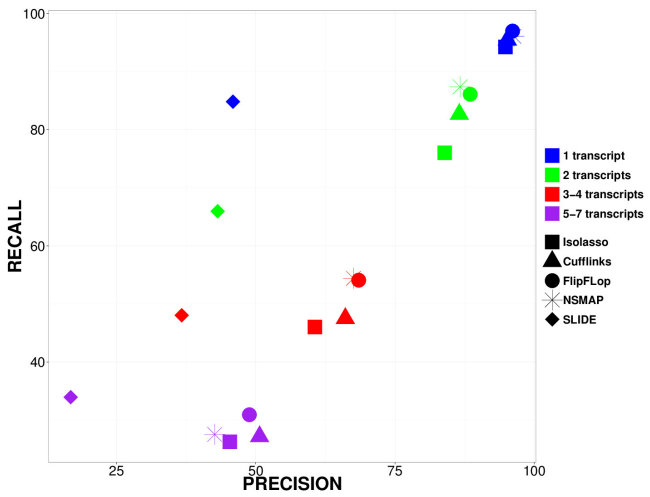


Figure 3: Precision and recall of compared methods on simulated reads from the UCSC annotated human transcripts.

result by this number: each dot represents the precision and recall of one method for genes with a particular number of transcripts in the UCSC annotation. As expected, genes with more transcripts lead to more difficult estimation problems and decreased performances for all methods. However, while all methods have similar performances for genes with a single expressed transcript, FlipFlop and NSMAP are less affected by the increase in transcript number than Cufflinks and IsoLasso. Cufflinks constructs its set of transcripts and estimates their abundances in two separate steps, and the construction of the set of returned transcripts does not take read density into account: it intends to find the smallest set of isoforms covering all the observed reads. IsoLasso is based on penalized likelihood maximization like FlipFlop and NSMAP, but starts from a very restricted set of isoforms — the same set returned by Cufflinks for single-end data. Consequently, this family of methods discards some information that can help identifying the set of expressed isoforms.

Consider for example the case of a gene with two exons A and B, with reads observed for both exons. Any method that builds the set of expressed exons as the smallest set covering all reads should conclude that only transcript A-B is present. Assume now that many more reads are observed for exon A than for exon B – *e.g.*, because an isoform only containing A is also present. The set of isoforms returned by Cufflinks and IsoLasso would not change. FlipFlop and NSMAP on the other hand maximize the penalized likelihood of the data over all possible transcripts, and are able to detect this type of structure. SLIDE maximizes the same type of objective as FlipFlop and NSMAP but behaves poorly in this experiment. We assume this may result from a problem for single-end data in the code: even though the software package of SLIDE explicitly states that it is compatible with single-end data, the experiments of Li *et al.* (2011a)— performed on paired-end data — report a similar improvement over Cufflinks as the one we observe here for FlipFlop and NSMAP. This poor behavior may also be caused by the model selection procedure: SLIDE uses stability selection whereas FlipFlop and NSMAP use similar procedures based on BIC criterion and likelihood ratio tests. Finally, FlipFlop and NSMAP yield very similar performances, which is expected given that they optimize the same objective function. The observed minor differences may arise from FlipFlop better optimizing the objective function (data not shown), NSMAP restricting its search to the TSS and PAS observed in the annotation whereas FlipFlop estimates them from reads, and the two methods using different model selection techniques.

Figure 4 shows the mean CPU time taken by each method to perform the deconvolution of genes

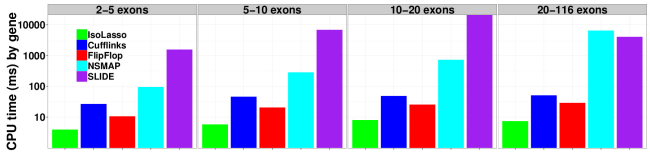


Figure 4: Average CPU times in milliseconds (on a logarithmic scale) for the compared methods to process a gene from human simulated RNA-Seq data.

with different sizes. Genes with more exons tend to have more candidate isoforms and experiments involving such genes are expected to take more time. Therefore, we stratify the observed times by exon number of the genes: each barplot represents the mean time taken by each method to find the transcripts of genes having a particular number of exons. As expected, FlipFlop is always faster than NSMAP and SLIDE which maximize similar objective function: more than a hundred times faster than SLIDE for all gene sizes, and a hundred times faster than NSMAP for genes with more than 5 exons. The fact that SLIDE is slower than NSMAP may be explained by its using stability selection for model selection — requiring to solve the maximization problem several times.

FlipFlop is actually a little faster than Cufflinks, and about 4 times slower than IsoLasso. This is because IsoLasso maximizes its objective over a very restricted set of candidates — in these simulations never more than 9 and around 2-3 on average. Overall, FlipFlop estimates the set of expressed isoforms for 1137 genes in less than 4 minutes, *i.e.*, about 5 genes per second. Note also that the time for data pre-preprocessing (finding exon boundaries and and read counts for exons and junctions) is not taken into account except for Cufflinks and SLIDE.

This simulation confirms two facts. First, methods that identify and quantify transcripts as a single penalized maximum likelihood problem yield better performances than the ones estimating the set of expressed isoforms as the smallest set covering all reads. Second, the proposed network flow strategy allows to solve the penalized likelihood approach quickly even when the set of candidate isoforms is extremely large.

4.2 Real RNA-Seq Data

Our second set of experiments involves the C57BL6 mouse liver cell line (Trapnell *et al.*, 2010), which contains about 42 million 33-bp reads (NCBI SRA accession number SRX000351). Reads are aligned to the mm9 reference genome of the UCSC genome browser. We only include FlipFlop, Cufflinks and IsoLasso in this comparison: NSMAP and SLIDE optimize similar objectives as FlipFlop and were already compared in Section 4.1.

In the experiments of Section 4.1, we generated the reads based on a known set of transcripts. In the present case, the reads come from actual mouse tissues, and we do not have access to the true set of expressed transcripts. Following Xia *et al.* (2011) and Li *et al.* (2011a), we choose to use the UCSC annotation as ground truth in the evaluation. Admittedly, this is not perfect as some expressed transcripts may be missing from the annotation, and some annotated transcripts may not be expressed in this particular experiment. However, agreement of the prediction with the set of known transcripts could be a good sign.

Figure 5 shows the speed, recall and precision of each method. Consistently with what we observed on simulated data, FlipFlop runs faster than Cufflinks but slower than Isolasso.

The reference transcripts set corresponds to the 33381 known mouse isoforms from the RefSeq track in UCSC browser. IsoLasso predicts 8096 isoforms with 4056 matching the annotations, Cufflinks predicts 24605 isoforms with 5804 matching and FlipFlop predicts 15613 isoforms with

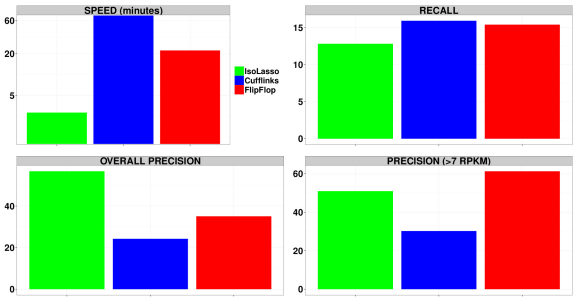


Figure 5: Precision, recall and speed of compared methods on C57BL6 mouse liver data.

4903 matching. These results lead to a much higher precision for IsoLasso than for Cufflinks and FlipFlop, and a slightly better recall for Cufflinks and FlipFlop. We remark that IsoLasso does not predict on this data set transcripts below 7 RPKM abundance. Indeed, low abundance prediction transcripts may result from mapping errors: Li *et al.* (2011a) actually only considered genes with at least 1 RPKM in their real data experiments. We therefore look at the precision of the methods when the set of predicted transcripts is limited to higher value than 7 RPKM. Consistently with what we observed on simulated data, FlipFlop gives the best precision when considering these reliable high abundance transcripts only. While it is always possible to manually discard genes with few observed reads or transcripts with low predicted abundance, this observation suggests that some systematic improvement of our method is possible, *e.g.* by using a different flow decomposition algorithm discouraging low-expression transcripts, a more efficient model selection method, or a more aggressive sparsity inducing penalty than ℓ_1, for instance, a reweighted-ℓ_1 technique (Candes *et al.*, 2008).

5 Discussion

Simultaneously tackling identification and quantitation using penalized likelihood maximization is known to be a powerful approach to estimate the set of expressed transcripts. However, existing optimization techniques cannot deal with genes that contain too many exons as the set of candidate isoforms grows exponentially with the number n of exons. By leveraging network flow optimization algorithms, we discover a few expressed transcripts among the exponential number of candidates by solving a problem with a number of variables polynomial in n.

We compared our approach to existing penalized likelihood maximization methods as well as methods that define expressed isoforms as the smallest set of transcripts covering all observed reads; the latter methods perform abundance estimation in a separate step. We observed on simulated data — where the true set of expressed transcripts was known — that penalized likelihood maximization indeed leads to better precision and recall than the second set of methods. This improvement was more important for genes with several expressed transcripts. A similar observation was made on real RNA-Seq data from mouse liver data as long as we restricted ourselves to transcripts whose abundance was not too low. In addition, the runtime of our method was comparable to the runtime of the second set of methods, and orders of magnitude faster than existing software for penalized likelihood maximization.

We believe these results have important practical implications. In addition to the obvious gain in time when estimating the expression of transcripts for a single gene and a single sample, our approach makes the task amenable in a reasonable amount of time for all genes in a large number of samples. This is a necessary step for high throughput differential expression studies at

the transcript level, a direction we are planning to explore in future work. Differential expression studies were until now restricted to gene level studies, *i.e.*, ignoring the transcript level information, to cases where the set of expressed transcripts was known in advance or to methods which were not using the read density to estimate the set of expressed transcripts — a less efficient approach as illustrated in our experiments. Furthermore, accurately estimating the transcript level expression for all genes of all samples in a study may lead to improvements in molecular based diagnosis or prognosis tools, as well as in clustering of samples, *e.g* for cancer subtype discovery. The ability of our approach to deal with splicing graphs with potentially hundreds of nodes also paves the way to efficient *de novo* transcript identification, where we do not restrict ourselves to annotated exons within a single gene. Future versions of the proposed method should also deal with paired-end data.

Fundings: This work was supported by the European Research Council [SMAC-ERC-280032 to J-P.V., E.B.]; the European Commission [HEALTH-F5-2012-305626 to J-P.V., E.B.]; the French National Research Agency [ANR-09-BLAN-0051-04 and ANR-11-BINF-0001 to J-P.V., E.B.]; the Stand Up to Cancer grant [SU2C-AACR-DT0409 to L.J.]; and the National Science Foundation [SES-0835531, CCF-0939370 to J.M.].

References

Ahuja, R. *et al.* (1993). *Network Flows*. Prentice Hall.

Bertsekas, D. (1998). *Network Optimization: Continuous and Discrete Models*. Athena Scientific.

Bohnert, R. and Rätsch, G. (2010). rQuant.web: a tool for RNA-Seq-based transcript quantitation. *Nucleic Acids Res*, **38**(Web Server issue), W348–W351.

Candes, E. *et al.* (2008). Enhancing sparsity by reweighted 1 minimization. *J Fourier Anal Appl*, **14**(5), 877–905.

Goldberg, A. V. (1997). An efficient implementation of a scaling minimum-cost flow algorithm. *J Algorithm*, **22**(1), 1–29.

Heber, S. *et al.* (2002). Splicing graphs and EST assembly problem. *Bioinformatics*, **18**(suppl 1), S181–S188.

Huang, Y. *et al.* (2012). A robust method for transcript quantification with RNA-Seq data. In *Proceedings of the 16th Annual international conference on Research in Computational Molecular Biology*, RECOMB'12, pages 127–147.

Jiang, H. and Wong, W. H. (2009). Statistical inferences for isoform expression in RNA-Seq. *Bioinformatics*, **25**(8), 1026–1032.

Li, J. J. *et al.* (2011a). Sparse linear modeling of next-generation mRNA sequencing (RNA-Seq) data for isoform discovery and abundance estimation. *P Natl Acad Sci USA*, **108**(50), 19867–19872.

Li, W. *et al.* (2011b). IsoLasso: a LASSO regression approach to RNA-Seq based transcriptome assembly. *J Comput Biol*, **18**, 1693–1707.

Mairal, J. and Yu, B. (2012). Supervised feature selection in graphs with path coding penalties and network flows. *preprint arXiv:1204.4539v1*. to appear in JMLR.

Medvedev, P. and Brudno, M. (2009). Maximum likelihood genome assembly. *J Compute Biol*, **16**(8), 1101–1116.

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **72**(4), 417–473.

Mezlini, A. M. *et al.* (2013). iReckon: Simultaneous isoform discovery and abundance estimation from RNA-seq data. *Genome res*, **23**(3), 519–529.

Montgomery, S. B. *et al.* (2010). Transcriptome genetics using second generation sequencing in a Caucasian population. *Nature*, **464**(7289), 773–777.

Mortazavi, A. *et al.* (2008). Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat Methods*, **5**(7), 621–628.

Mortazavi, A. *et al.* (2010). Scaffolding a caenorhabditis nematode genome with RNA-Seq. *Genome Res*, **20**(12), 1740–1747.

Pan, Q. *et al.* (2008). Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nat Genet*, **40**(12), 1413–1415.

Perou, C. M. *et al.* (2000). Molecular portraits of human breast tumours. *Nature*, **406**(6797), 747–752.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics*, **6**(2), 461–464.

Singh, D. *et al.* (2011). FDM: a graph-based statistical method to detect differential transcription using RNA-seq data. *Bioinformatics*, **27**(19), 2633–2640.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *J Roy Stat Soc B*, **58**(1), 267–288.

Trapnell, C. *et al.* (2009). TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, **25**(9), 1105–1111.

Trapnell, C. *et al.* (2010). Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat Biotechnol*, **28**(5), 511–515.

van't Veer, L. J. *et al.* (2002). Gene expression profiling predicts clinical outcome of breast cancers. *Nature*, **415**(6871), 530–536.

Xia, Z. *et al.* (2011). NSMAP: a method for spliced isoforms identification and quantification from RNA-Seq. *BMC Bioinformatics*, **12**, 162.

Xing, Y. *et al.* (2006). An expectation-maximization algorithm for probabilistic reconstructions of full-length isoforms from splice graphs. *Nucleic Acids Res*, **34**(10), 3150–3160.