A Novel Min-Cost Flow Method for Estimating Transcript Expression with RNA-Seq

Alexandru I. Tomescu[1], Anna Kuosmanen[1], Romeo Rizzi[2], and Veli Mäkinen[1]

[1] Helsinki Institute for Information Technology (HIIT),
Department of Computer Science, University of Helsinki, Finland
{tomescu,aekuosma,vmakinen}@cs.helsinki.fi
[2] Department of Computer Science, University of Verona, Italy
romeo.rizzi@univr.it

Abstract. Through transcription and alternative splicing, a gene can be transcribed into different RNA sequences (an isoform), depending on the individual and on the tissue the cell is in. Recent RNA-Seq technology allows for new high-throughput ways for isoform identification and quantification based on short reads, and various methods have been put forward for this non-trivial problem.
In this paper we propose a novel radically different method based on minimum-cost network flows. This has a two-fold advantage: on the one hand, it translates the problem as an established one in the field of network flows, which can be solved in polynomial time, with different existing solvers; on the other hand, it is general enough to encompass many of the previous proposals under the least sum of squares model. Our method works as follows: in order to find the transcripts which best explain, under a given fitness model, a splicing graph resulting from an RNA-Seq experiment, we find a min-cost flow in an offset flow network, under an equivalent cost model. Under very weak assumptions on the fitness model, the optimal flow can be computed in polynomial time. Parsimoniously splitting the flow back into few path transcripts can be done with any of the heuristics and approximations available from the theory of network flows. In the present implementation, we choose the simple strategy of removing the heaviest path. Experimental results on prediction accuracy show that our method is very competitive as it provides better precision and recall than popular tools such as Cufflinks and IsoLasso.

Keywords: Isoform Identification, Isoform Quantification, RNA-Seq, Graph Optimization Problem, Network Flow, Min-Cost Flow

1 Introduction

Recent RNA-Seq technology [13, 15] opened a new high-throughput, low cost way for isoform identification and quantification, leading to new understanding of gene regulation in development and disease (e.g., [18]). In an RNA-Seq experiment a set of short reads is produced from mRNA transcripts. The difficulty in assembling these short reads into the transcripts from which they were sampled is non-trivial due to the fact that the transcripts may share exons. As a result, all methods for solving this problem rely on an explicit or implicit graph model. The nodes represent individual reads (overlap graph [21]), or contiguous stretches of DNA uninterrupted by spliced reads (splicing graph [6, 11, 9], connectivity graph [3, 10, 4]), while the edges are derived from overlaps or from spliced read alignments. Each node and edge has an associated observed coverage, and the problem of isoform identification and quantification is seen as separating the coverage of the graph into individual path components, under different models. Furthermore, this problem was also coined under the broad name 'Multiassembly Problem' [26], a hint that it can arise not only with RNA-Seq data, but also in other biological settings, such as assembling metagenomics reads [14].

Except for Cufflinks [21], all tools mentioned above rely on some optimization engine, whose solving is generally difficult. IsoInfer/IsoLasso [3, 10], SLIDE [9], Scripture [4], and CLIIQ [11] exhaustively enumerate all possible candidate paths. For efficiency reasons, each has some restrictions on what a valid candidate path might be, and for each candidate isoform, they define a fitness function. IsoInfer/IsoLasso and SLIDE use a least sum of squares fitness function; IsoLasso and SLIDE

both add different shrinkage terms to the fitness function in order to favor isoforms with fewer transcripts, which is computed with a modified LASSO algorithm, or a quadratic program; CLIIQ uses a least sum of absolute differences fitness function, solved by a linear integer program. Cufflinks avoids the problem of exhaustively enumerating all possible paths by returning a minimum path cover, and then assigning expression levels to each path in this cover based on a statistical model. Incidentally, note that computing a minimum path cover (in an acyclic digraph) is done by computing a maximum matching, which can be easily reduced to a flow problem. However, such reduction solves a different (implicitly defined) optimization problem than can be considered as a consensus model in the literature [3, 10, 9, 4, 11], mostly because the fitting of expression levels is separated in the process.

Our contribution. In this paper we propose a radically different and very general method relying on the established field of minimum-cost network flow problems [1]. This will not only provide a simple method and a fast polynomial time algorithm for solving it (as opposed to exhaustively enumerating all possible candidate paths, and then solving a quadratic/integer linear program for evaluating the fitness of each candidate isoform), but it can also lean on the ample literature on splitting a (min-cost) flow into paths, e.g., [24, 5, 7, 17].

As in the case of the other tools, our method assumes that a splicing graph has been built for each gene. Each node of the graph corresponds to a stretch of DNA uninterrupted by any spliced read alignment; such sequences are called *segments* in [10], but for simplicity we just call them *exons*. Each edge of the graph corresponds to two exons consecutive in some transcript, that is, to some spliced read whose prefix aligns to the suffix of one exon, and whose suffix aligns to the prefix of another exon. Observe that such a graph can be seen as a directed acyclic graph (DAG, for short), the direction of the edges being according to the absolute position of the exons in the genome. For each exon v we can deduct its coverage $cov(v)$ as the total number of reads aligned to the exon divided by the exon length, and the coverage $cov(u, v)$ of an edge (u, v) as the total number of reads split aligned to that junction between exons u and v. An mRNA transcript candidate thus becomes a path from a source node $s \in V$ to a sink node $t \in V$.[1]

In order to define a fitness function in the broadest possible terms, let us assume that for each node v and edge (u, v) of the graph we have convex cost functions $f_v, f_{uv} : \mathbb{R} \to \mathbb{R}$ modeling how close that node and edge must be explained by the candidate isoform. Then, we can state the problem of isoform identification and quantification as following problem.

Problem 1 (UTEC). Given a splicing DAG $G = (V, E)$ with coverage values $cov(v)$ and $cov(u, v)$, and cost functions $f_v(\cdot)$ and $f_{uv}(\cdot)$, for all $v \in V$ and $(u, v) \in E$, the *Unannotated Transcript Expression Cover* problem is to find a tuple $\mathcal{P}$ of paths from the sources of G to the sinks of G, with an estimated expression level $e(P)$ for each path $P \in \mathcal{P}$, which minimize

$$\mathsf{sum\_err}(G, \mathcal{P}) := \sum_{v \in V} f_v\left(\left|cov(v) - \sum_{P \in \mathcal{P}:\, v \in P} e(P)\right|\right) + \sum_{(u,v) \in E} f_{uv}\left(\left|cov(u, v) - \sum_{P \in \mathcal{P}:\, (u,v) \in P} e(P)\right|\right).$$

For example, if for all nodes v and edges (u, v), $f_v(x) = x$, $f_{uv}(x) = x$, then we have a least sum of absolute differences model as in CLIIQ. If $f_u(x) = x^2$, $f_{uv}(x) = x^2$, then we have a least sum

[1] The requirement that the transcripts start in a source node and end in a sink node is no restriction, as we can add dummy source/sink nodes as in-/out-neighbors to the nodes where we have indication that some transcript might start/end. Indeed, our splicing graph creation tool uses splicing alignments and coverage information to discover such start/end nodes and accordingly indicates them for our tool.

of squares model as in IsoInfer, IsoLasso and SLIDE. Observe that many of the other biological assumptions of the other tools can be incorporated in this model. Motivated by also by [23], in this paper we focus on two convex cost function, f^1 and f^2, of the form

$$f^1_v(x) = \frac{x}{cov(v)}, \quad f^1_{uv}(x) = \frac{x}{cov(u,v)}, \qquad\qquad f^2_v(x) = x^2, \quad f^2_{uv}(x) = x^2. \tag{1}$$

We will show that Problem UTEC can be solved in polynomial time, by a reduction to a min-cost flow problem with convex cost functions. We will argue that finding the optimal tuple of paths explaining the graph is equivalent to finding the optimal flow in an offset flow network. Moreover, any splitting of this optimal flow into paths attains the minimum of Problem UTEC. In the same way as some of the other tools try to limit the number of paths explaining a splicing graph by a LASSO approach, we can rely on established methods for splitting any flow into few paths (e.g., [24, 5, 7, 17]). In this paper, we employ only the simple linear-time heuristic of repeatedly removing the heaviest path, see e.g., [5].

We give experimental results to study how well the predictions match the ground-truth on simulated data, and how well it fares on real-data, compared to Cufflinks [21] and IsoLasso [10]; our method is very competitive, providing better precision and recall.

2 Method

We begin by recalling the basic notions of flow and of a min-cost flow problem, and refer to the excellent monograph [1] for further details. A *flow network* (or simply *network*) is a tuple $N = (G, b, q)$, where $G = (V, E)$ is a directed graph, b is a function assigning a *capacity* $b_{uv} \in \mathbb{N}$ to every arc $(u, v) \in E$, and q is a function assigning an *exogenous flow* $q_v \in \mathbb{N}$ to every node $v \in V$, such that $\sum_{v \in V} q_v = 0$. We say that a function x assigning to every arc $(u, v) \in E$ a number $x_{uv} \in \mathbb{N}$ is a *flow* over the network N, if the following two conditions are satisfied:

1. $0 \leqslant x_{uv} \leqslant b_{uv}$, for every $(u, v) \in E$,
2. $\sum_{u \in V} x_{vu} - \sum_{u \in V} x_{uv} = q_v$, for every $v \in V$,

In a min-cost flow problem, one is additionally given flow cost functions $c_{uv}(\cdot)$, for every arc $(u, v) \in E$, and is required to find a flow which minimizes:

$$\sum_{(u,v) \in E} c_{uv}(x_{uv}).$$

It is well-known that, under the assumption that all the flow cost functions $c_{uv}(\cdot)$ are convex, a min-cost flow can be found in polynomial time [12] (see also [25] for the real-valued flow case).

2.1 The Reduction to a Min-Cost Flow Problem

We will model Problem UTEC as a min-cost flow problem, thus showing that it can be solved in polynomial time. First, we argue that it can be transformed into the following equivalent problem, where the input exon chaining graph has measured coverages only on arcs.

Problem 2 (UTEJC). Given a splicing DAG $G = (V, E)$ with coverage values $cov(u, v)$, and cost functions $f_{uv}(\cdot)$, for all $(u, v) \in E$, the *Unannotated Transcript Expression Junction Cover* problem

is to find a tuple $\mathcal{P}$ of paths from the sources of G to the sinks of G with an estimated expression level $e(P)$ for each path $P \in \mathcal{P}$, which minimize

$$\sum_{(u,v)\in E} f_{uv}\left(\left|cov(u,v) - \sum_{P\in\mathcal{P}:\,(u,v)\in P} e(P)\right|\right).$$

Given an input $G = (V, E)$ for Problem UTEC, we construct an input for Problem UTEJC by replacing every node $v \in V$ with two new nodes, v_{in} and v_{out}, and an arc (v_{in}, v_{out}), with $cov(v_{in}, v_{out}) = cov(v)$, and $f_{v_{in}v_{out}}(x) = f_v(x)$. Furthermore, for every arc $(u, v) \in E$, we replace arc (u, v) with the arc (u_{out}, v_{in}), with the same coverage as (u, v). It is immediate that optimal solutions for G to Problem UTEC are in bijection with the optimal solutions for the transformed graph to Problem UTEJC.

To solve Problem UTEJC, we build an auxiliary offset network with convex costs of the form $c_{uv}(x) = f_{uv}(x)$. An optimal flow for this network will model the offsets (positive or negative) between the measured coverages of the exon chaining graph and their actual expression levels in an optimal solution. Then, we argue that a min-cost flow on this network naturally induces a solution for the UTEJC problem.

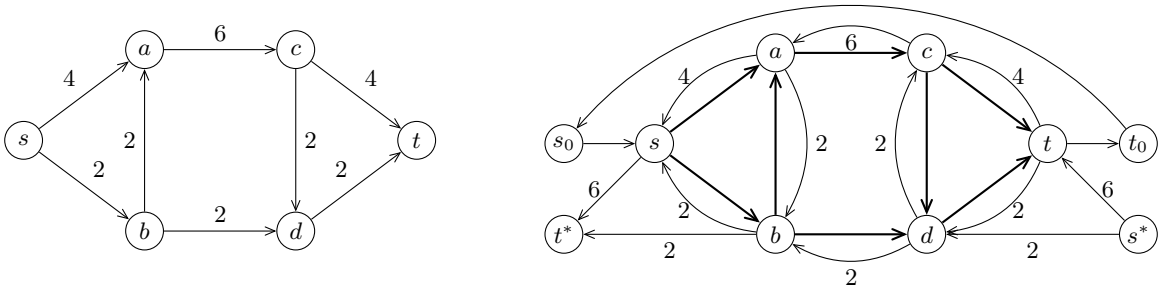
Onwards, we denote by $N^+_G(v)$ the set of *out-neighbors* of v in the directed graph G, that is, the set $\{w \ : \ (v, w) \in E(G)\}$. Similarly, we denote by $N^-_G(v)$ the set of *in-neighbors* of v in the directed graph G, that is, the set $\{u \ : \ (u, v) \in E(G)\}$. When G is clear from the context, we will skip it as subscript.

Given a splicing DAG G with coverage values $cov(u, v)$, and cost functions f_{uv}, for all $(u, v) \in E$, we construct the *offset network* $N^* = (G^*, b, q)$ with cost function c, as follows (see Fig. 1 for an example):

1. we add to G^* all nodes and edges of G, together with
   (a) a new source s_0 and a new sink t_0 with $q_{s_0} := q_{t_0} := 0$,
   (b) arcs (s_0, s), for every source s of G, and arcs (t, t_0) for every sink t of G, each with infinite capacity and null cost function,
   (c) arc (t_0, s_0) with infinite capacity and null cost function,
   (d) nodes s^* and t^*, with initial exogenous flow $q_{s^*} := q_{t^*} := 0$;
2. for every arc $(u, v) \in E(G)$,
   (a) $b_{uv} := \infty$, $c_{uv}(x) := f_{uv}(x)$,
   (b) we add the reverse arc (v, u) to G^* with $b_{vu} := cov(u, v)$, $c_{vu}(x) := f_{uv}(x)$;
3. for every $v \in V(G)$,
   (a) its exogenous flow q_v is zero,
   (b) if $\sum_{u\in N^+(v)} cov(v, u) - \sum_{u\in N^-(v)} cov(u, v) > 0$, we add arc (v, t^*) to G^* where:
      i. $b_{vt^*} := \sum_{u\in N^+(v)} cov(v, u) - \sum_{u\in N^-(v)} cov(u, v)$, $c_{vt^*}(x) := 0$,
      ii. we update $q_{t^*} := q_{t^*} + \sum_{u\in N^-(v)} cov(u, v) - \sum_{u\in N^+(v)} cov(v, u)$;
   (c) if $\sum_{u\in N^+(v)} cov(v, u) - \sum_{u\in N^-(v)} cov(u, v) < 0$, we add arc (s^*, v) to G^* where:
      i. $b_{s^*v} := \sum_{u\in N^-(v)} cov(v, u) - \sum_{u\in N^+(v)} cov(u, v)$, $c_{s^*v}(x) := 0$,
      ii. we update $q_{s^*} := q_{s^*} + \sum_{u\in N^-(v)} cov(v, u) - \sum_{u\in N^+(v)} cov(u, v)$.

The next lemma shows that there exists a min-cost flow x^* on N^*.

Lemma 1. *Given a digraph G with arc coverages $cov(\cdot, \cdot)$, the offset network $N^* = (G^*, b, q)$ constructed as above is a flow network, i.e., $\sum_{v\in V(G^*)} q_v = 0$.*



(a) An input G to Problem UTEJC

(b) The offset network G^*; arcs are labeled with their capacity, unlabeled arcs having infinite capacity

Fig. 1. Example of an offset network.

Proof. Since $q_v = 0$, for all $v \in V(G^*) \setminus \{s^*, t^*\}$, it remains to show that $q_{s^*} + q_{t^*} = 0$. Indeed,

$$\begin{aligned} q_{s^*} + q_{t^*} &= \sum_{v \in V(G)} \left(\sum_{u \in N_G^-(v)} cov(u,v) - \sum_{u \in N_G^+(v)} cov(v,u) \right) \\ &= \sum_{v \in V(G)} \sum_{u \in N_G^-(v)} cov(u,v) - \sum_{v \in V(G)} \sum_{u \in N_G^+(v)} cov(v,u) \\ &= \sum_{(u,v) \in E(G)} cov(u,v) - \sum_{(v,u) \in E(G)} cov(v,u) = 0 \end{aligned}$$

□

From such a flow x^*, we construct the function x on the edges G as follows. First, observe that for every arc $(u,v) \in E(G)$, at most one of x^*_{uv} or x^*_{vu} is nonnull. Indeed, if this were not the case, then a flow y^* which coincides with x^*, except for $y^*_{uv} := x^*_{uv} - \min(x^*_{uv}, x^*_{vu})$ and $y^*_{vu} := x^*_{vu} - \min(x^*_{uv}, x^*_{vu})$, is also a flow on N^* and has a strictly smaller cost than x^*, contradicting the fact that x^* is of minimum cost. Then, for each arc $(u,v) \in E(G)$ we set:

$$x_{uv} := cov(u,v) + x^*_{uv} - x^*_{vu}.$$

2.2 From a Flow to a Set of Paths

Theorem 1 below will argue that the above defined function x is a flow on G (points (1), (2)), whose arcs we consider to have unbounded capacities and whose nodes, apart from the sources and sinks, have exogenous flow 0. It is a well-known result from classical network flow theory that such a flow can be decomposed into paths, that is, there exist paths $P_1, \ldots, P_t$ from the sources of G to the sinks of G, having weights $w_1, \ldots, w_t$, respectively, such that, for every $(u,v) \in E(G)$ we have

$$x_{uv} = \sum_{i \,:\, (u,v) \text{ belongs to } P_i} w_i.$$

Moreover, a decomposition of x into at most $|E(G)|$ paths always exists and can be found in time $|V(G)| \cdot |E(G)|$. Theorem 1 also shows that the paths of any decomposition of x are an optimal solution for G to Problem UTEJC (point (3)). Its proof is available in Appendix A.

Theorem 1. *Given an optimal flow x^* on G^*, the function x on G just constructed satisfies the properties, where S denotes the set of sources of G, and T denotes the set of sinks of G:*

(1) for all $v \in V(G) \setminus \{s,t\}$, $\sum_{u \in N_G^-(v)} x_{uv} = \sum_{u \in N_G^+(v)} x_{vu}$;
(2) $\sum_{s \in S} \sum_{v \in N_G^+(s)} x_{sv} = \sum_{t \in T} \sum_{u \in N_G^-(t)} x_{ut}$
(3) any decomposition of x into paths attains the minimum of the objective function of Problem UTEJC, on input G.

In our implementation we use the min-cost flow engine available in the `LEMON Graph Library` [8]. If no engine for arbitrary convex cost functions is available, or, more generally, if the cost functions themselves happen not to be convex, one can approximate any cost function with piecewise constant or convex cost functions: e.g., one can replace an arc (u, v) of capacity b_{uv}, with $|b_{uv}|$ arcs of capacity 1, such that first arc has cost $f(1)$, and the ith arc, $i > 1$, has cost $f(i) - f(i-1)$ (this reduction is only pseudo-polynomial but reveals quite effective in practice), see [1] for further details.

2.3 Decomposing the Min-Cost Flow into Few Paths

As already shown by the other tools, we are generally interested in *parsimoniously* explaining an RNA-seq experiment, that is, in finding, among the optimal solutions to Problem UTEC, one with a low number of paths. At a closer analysis it can be seen that any flow on a graph $G = (V, E)$ can be decomposed into at most $|E| - |V| + 2$ paths [24]. However, decomposing a flow into a minimum number of paths is an NP-hard problem in the strong sense, even when restricted to DAGs [24, 5]. To overcome this limitation, various heuristics and approximations have been put forth, see, e.g., [24, 5, 7, 17] and the references therein. The advantage of our method is that once we have obtained the optimal flow, we can apply any of these methods to split the flow into few paths. For simplicity, in this paper we employ the policy of removing the heaviest path, see, e.g., [5]: until the network has null flow, we select a path from the sources to the sinks whose minimum flow on its edges is maximum, report it as transcript, and remove it from the flow network.

3 Experiments

We call our tool Traph (*T*ranscripts in G*raph*s). We compared Traph to the most used isoform prediction tool Cufflinks [21] and with IsoLasso [10]. We also tried to include SLIDE [9] and CLIIQ [11], but we could not make the former work reliably, and for the latter the publicly available version was not yet available. Full experiment data is available at [20].

3.1 Matching criteria

In order to match the predicted transcripts with the true transcripts, we take into account the DNA sequences but also the expression levels. For each gene, we construct a bipartite graph with the true transcripts $\mathcal{T} = (T_1, T_2, \dots)$ as nodes in one set of the bipartition, and the predicted transcripts $\mathcal{P} = (P_1, P_2, \dots)$ as nodes in the other set of the bipartition. Empty sequences with 0 expression level were added so that both sets of the bipartition had an equal number of nodes.

To define the costs of the edges of this bipartite graph, let us introduce (cf. Normalized Compression Distance [2]) the binary encoding of a true transcript T and its expression level $e(T)$ with respect to a predicted transcript P with expression level $e(P)$

$$\mathsf{code}(T \mid P, j) = \gamma(j)\gamma(d+1)\mathsf{editsencoded}(T, P)\gamma(f(e(T) - e(P))), \tag{2}$$

where $\gamma(x) = 0^{|bin(x)|-1}1bin(x)$, $bin(x)$ being the binary encoding of $x > 0$, j is the index of P in the list of predicted transcripts, d is the unit cost (Levenshtein) edit distance of T and P, $\mathsf{editsencoded}(T, P)$ lists the edits and gaps between edits using 2-bit fixed code for edit type, 2-bit fixed code for substituted/inserted symbol, and $\gamma(x+1)$ for gap (run of identities) of length x, and $f(x)$ is a bijection between $\{0, 1, -1, 2, -2, \ldots\}$ and $\{1, 2, 3, 4, 5, \ldots\}$ defined as $f(x) = 2x$ for $x > 0$ and $f(x) = 2(-x) + 1$ otherwise.

Then, the edge cost between nodes $T_i \in \mathcal{T}$ and $P_j \in \mathcal{P}$ is defined as $|\mathsf{code}(T_i \mid P_j, j)| - |\gamma(j)|$. The closer to zero this number is, the better the match between true transcript T_i, with true expression level $e(T_i)$ and predicted transcript P_j with predicted expression level $e(P_j)$. The minimum weight perfect matching was then computed; this gives a one-to-one mapping between true and predicted transcripts, therefore true transcripts can be ordered in the same order as they match predicted transcripts and code for the index, $\gamma(j)$, is no longer required. Let *edit code length* for an edge between T_i and P_j be $|\gamma(d+1)\mathsf{editsencoded}(T_i, P_j)|$, where d is the edit distance. Let *bitscore* be edit code length divided by $|\gamma(|T_i| + 1)\,\mathsf{editsencoded}(T_i, \epsilon)|$; bitscore is asymmetric, and possibly greater than 1 if ϵ would be a better match to T_i than to P_j, but minimum weight perfect matching chose otherwise for global minimality. Each matched node pair with relative expression level difference and (edit) bitscore under some given thresholds define a true positive event (TP). The other kind of nodes define false positive (FP) and false negative (FN) events depending on which side of the bipartite graph they reside. Prediction efficiency based on precision, recall and F-measure is also employed in [10, 11].

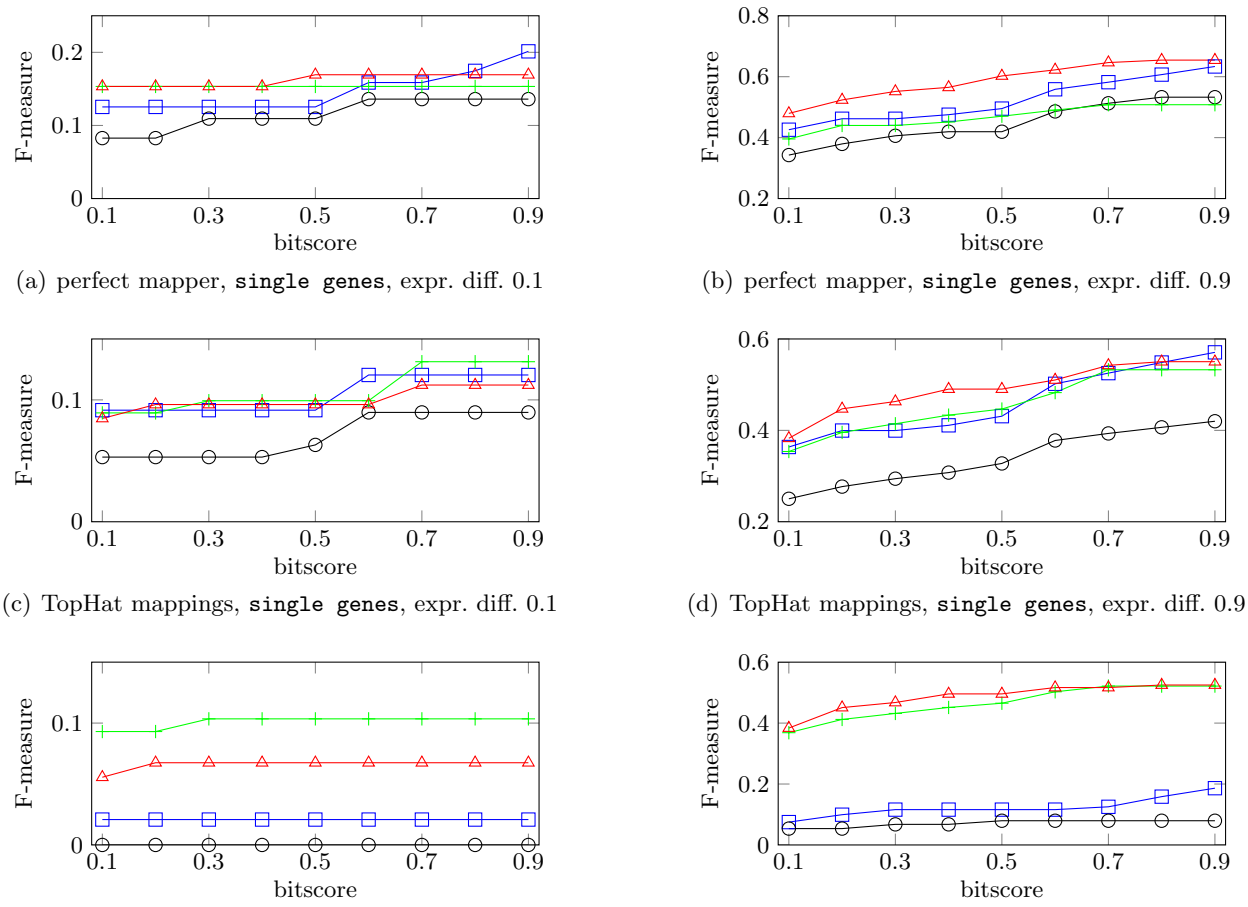
3.2 Simulated human data

As in the case of the other tools, we deem that validating against simulated data is a prerequisite, since, in general, on real data, we do not have available ground-truth. We designed the following validation experiment, closely following the approaches in [11, 10]. We chose a set of genes at random, and looked up the corresponding annotated transcripts from the Ensembl database. Out of these genes, we selected only those having between 2 and 5 transcripts. In all, we had 29 genes. For each transcript, we simulated reads with the RNASeqReadSimulator [16]. This simulator chooses an expression level at random from lognormal distribution with mean -4 and variance 1. For each gene, it simulated 300 000 reads as follows: a transcript was chosen randomly using its expression level as distribution weight, while the position of the read within the transcript was chosen uniformly. As argued in the case of IsoLasso [10], various error models can be incorporated in these steps, but we chose to compare the performance of the methods in neutral conditions. We mapped the reads with TopHat [22]: these read mapping results were given as input to the tested prediction software, and to a Python program which we wrote to construct the splicing graphs needed for Traph. Cufflinks and IsoLasso were ran with the default parameters. Since our tool is not yet employing existing gene annotation information, we ran Cufflinks and IsoLasso without annotation. We use RPKM values as expression levels.

We devised two experiments: one in which the reads simulated from each gene were fed independently to TopHat for alignment, and these independent alignment results were fed to each tool (we call it `single genes`); and second more realistic one in which all the reads simulated from all the genes were combined into one file which was given to TopHat for alignment, and these combined alignments results were fed to the tools (we call it `batch mode`).

Table 1 and Fig. 2 show selected validation results. The measures reported are precision = TP/(TP+FP), recall = TP/(TP+FN), and F-measure = 2∗precision∗recall/(precision+recall). We selected to depict two relative expression level differences, 0.1 and 0.9, illustrating opposite expression levels matching criteria. In the first, we require that the predicted expression levels be at

most 10% different from true ones, and in the second they can be at most 90% different from the true ones. Traph with fitting function f^1 generally performs best, its lead being even higher at big expression level difference. Very interestingly, in the `batch mode` experiment, very similar to how RNA-Seq data is acquired, Traph is an order of magnitude better than its two competitors, the fitness function f^2 giving better results than f^1 at small expression level difference.



(a) perfect mapper, `single genes`, expr. diff. 0.1

(b) perfect mapper, `single genes`, expr. diff. 0.9

(c) TopHat mappings, `single genes`, expr. diff. 0.1

(d) TopHat mappings, `single genes`, expr. diff. 0.9

(e) TopHat mappings, `batch mode`, expr. diff. 0.1

(f) TopHat mappings, `batch mode`, expr. diff. 0.9

Fig. 2. Performance of IsoLasso (black circle), Cufflinks (blue square), Traph with cost function $f^1(x) = x/cov$ (red triangle), and with cost function $f^2(x) = x^2$ (green cross) on simulated data. Figs. 2(a) and 2(b) depict results with perfect mapping, in the `single genes` scenario; Figs. 2(c) and 2(d) depict results with TopHat mappings, also in the `single genes` scenario. Figs. 2(e) and 2(f) depict results with TopHat mappings in the `batch mode` scenario.

3.3 Real human data

We used the same real dataset from the IsoLasso paper [10], Caltech RNA-Seq track from the ENCODE project [19], NCBI SRA accession number SRR065504, consisting of 75bp paired-end reads. Out of these reads, we picked the 2,406,339 which mapped to human chromosome 2. We match transcripts predicted by one software to the transcripts predicted by the other two, employing the same minimum weight perfect matching method presented in Sec. 3.1, this time without taking

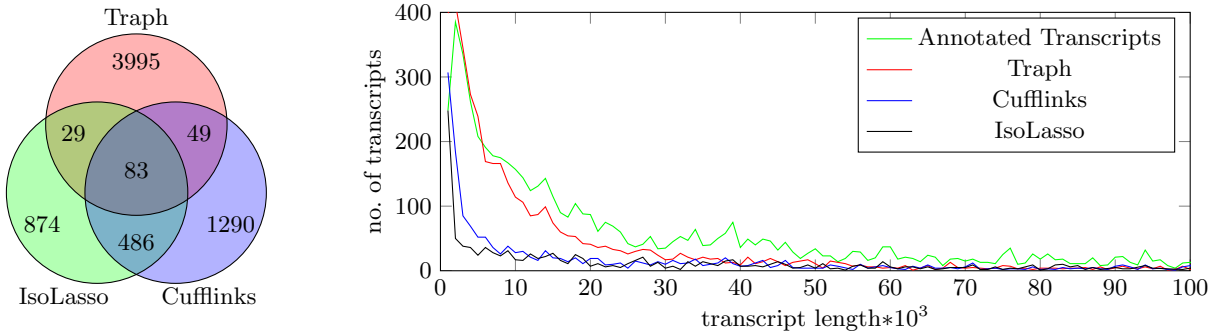
Table 1. Performance of the three tools under scrutiny, for TopHat mappings and in the `single genes` scenario; precision, recall and F-measure are computed for (relative expression level difference, bitscore) $\in \{(0.1,0.2),(0.9,0.2)\}$.

	Precision		Recall		F-measure		Avg. run time/gene
	$(0.1, 0.2)$	$(0.9, 0.2)$	$(0.1, 0.2)$	$(0.9, 0.2)$	$(0.1, 0.2)$	$(0.9, 0.2)$	
IsoLasso	0.0554	0.3310	0.0633	0.3240	0.0530	0.2769	25 s
Cufflinks	0.0966	0.4195	0.0880	0.3993	0.0915	0.3995	40 s
Traph $f^1(x) = x/cov$	**0.1000**	**0.4556**	**0.1013**	**0.4786**	**0.0961**	**0.4471**	40 s
Traph $f^2(x) = x^2$	0.0933	0.3894	0.0866	0.4440	0.0893	0.3955	72 s

into account expression levels. A true positive is a match selected by the perfect matching with bitscore under 0.2.

In all, we had 721 genes on which the three tools made some prediction. We should note that these 721 genes have 6466 annotated transcripts; Traph (with fitness function f^2) predicted in total 4156 transcripts for these genes, Cufflinks 1908, and IsoLasso 1472. As in [10], we depict in Fig. 3(a) a more detailed Venn diagram of the intersections between the sets of transcripts reported by the three tools.

We also include a histogram of the lengths of the annotated transcripts of these genes, and of the ones reported by Traph, IsoLasso, and Cufflinks. Here we round all transcript lengths to the nearest multiple of 1000. We see that the distribution in the case of Traph is similar to the distribution in the case of the annotated transcripts; the distributions for Cufflinks and IsoLasso are similar, but not very close to the annotation.



(a) Venn diagram of the intersections of the sets of reported transcripts

(b) Histogram of the distribution of transcript lengths of the annotation, and reported by Traph, Cufflinks and IsoLasso

Fig. 3. Results on real human data from ENCODE project [19], NCBI SRA accession number SRR065504.

3.4 Running times

On the real dataset, Cufflinks finished in 20 min, IsoLasso in 2 min, and Traph in 30 min. We should however stress that for solving the min-cost flow problem and for identifying the transcripts, Traph with cost function f^2 uses in fact 6 min, the rest of the time being spent by our graph creation tool, which is written in Python. We could not make such a detailed analysis in the case of the other two tools. The running time of our Python script is as well included in the last column of Table 1, were we listed the average running time per gene with simulated reads of each tool.

4 Discussion

All tools for isoform identification and quantification use an explicit or implicit graph model. Resorting to such a representation, the main contribution of this paper consists in a novel, radically different method based on minimum-cost flows, an established problem, for which there exist polynomial-time algorithms and solvers. Our framework is general enough to encompass many of the previous proposals under the least sum of squares model. We implemented this method into our tool Traph, and showed that it gives better precision and recall on simulated data, especially in the realistic setting when all simulated reads are collected into one file, which is fed to the tools. We also included results on real data, showing that the distribution of the lengths of our predicted transcripts is closer to the one of the lengths of the annotated transcripts.

Our method is general enough to easily accommodate other biological assumptions, which we leave for future work. Among these, we plan to integrate existing gene annotation into a more refined construction of the splicing graph and into the fitness model, or to correct sequencing bias. In order to evaluate the tools against real ground-truth data, we have started a process of acquiring sequencing reads (PacBio) of the true isoforms of a gene.

Acknowledgements. We wish to thank Antti Honkela for many insightful discussions on transcript prediction.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Inc. (1993)
2. Cilibrasi, R., Vitányi, P.M.B.: Clustering by compression. IEEE Transactions on Information Theory **51**(4) (2005) 1523–1545
3. Feng, J., Li, W., Jiang, T.: Inference of isoforms from short sequence reads. In Berger, B., ed.: RECOMB. Volume 6044 of Lecture Notes in Computer Science, Springer (2010) 138–157
4. Guttman, M., Garber, M., Levin, J.Z., Donaghey, J., Robinson, J., Adiconis, X., Fan, L., Koziol, M.J., Gnirke, A., Nusbaum, C., Rinn, J.L., Lander, E.S., Regev, A.: Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincrnas. Nat Biotechnol **28**(5) (May 2010) 503–510
5. Hartman, T., Hassidim, A., Kaplan, H., Raz, D., Segalov, M.: How to split a flow? In Greenberg, A.G., Sohraby, K., eds.: INFOCOM, IEEE (2012) 828–836
6. Heber, S., Alekseyev, M., Sze, S.H., Tang, H., Pevzner, P.A.: Splicing graphs and EST assembly problem. Bioinformatics **18**(suppl 1) (2002) S181–S188
7. Koch, R., Skutella, M., Spenke, I.: Maximum k-splittable s,t-flows. Theory of Computing Systems **43** (2008) 56–66
8. Lemon Graph Library: `http://lemon.cs.elte.hu/trac/lemon/`
9. Li, J.J., Jiang, C.R., Brown, J.B., Huang, H., Bickel, P.J.: Sparse linear modeling of next-generation mRNA sequencing (RNA-Seq) data for isoform discovery and abundance estimation. Proceedings of the National Academy of Sciences **108**(50) (2011) 19867–19872
10. Li, W., Feng, J., Jiang, T.: IsoLasso: a LASSO regression approach to RNA-Seq based transcriptome assembly. J. Comput. Biol. **18**(11) (2011) 1693–707
11. Lin, Y.Y., Dao, P., Hach, F., Bakhshi, M., Mo, F., Lapuk, A., Collins, C., Sahinalp, S.C.: CLIIQ: Accurate Comparative Detection and Quantification of Expressed Isoforms in a Population. In: Proc. Algorithms in Bioinformatics - 12th International Workshop, WABI 2012. Volume 7534 of Lecture Notes in Computer Science, Springer (2012) 178–189
12. Minoux, M.: Solving integer minimum cost flows with separable convex cost objective polynomially. In Gallo, G., Sandi, C., eds.: Netflow at Pisa. Volume 26 of Mathematical Programming Studies. Springer Berlin Heidelberg (1986) 237–239
13. Mortazavi, A., Williams, B.A.A., McCue, K., Schaeffer, L., Wold, B.: Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nature Methods **5** (2008) 621–628
14. Namiki, T., Hachiya, T., Tanaka, H., Sakakibara, Y.: Metavelvet: an extension of velvet assembler to *de novo* metagenome assembly from short sequence reads. Nucl. Acids Res. (2012)

15. Pepke, S., Wold, B., Mortazavi, A.: Computation for ChIP-seq and RNA-seq studies. Nature methods **6**(11) (2009) s22–s32
16. RNASeqReadSimulator: `http://www.cs.ucr.edu/~liw/rnaseqreadsimulator.html`
17. Salazar, F., Skutella, M.: Single-source k-splittable min-cost flows. Oper. Res. Lett. **37**(2) (March 2009) 71–74
18. Shah, S., et al.: The clonal and mutational evolution spectrum of primary triple-negative breast cancers. Nature **486**(7403) (June 2012) 395–399
19. The ENCODE Project Consortium: Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project. Nature **447** (June 2007) 799–816
20. Traph experiments data: `http://cs.helsinki.fi/u/tomescu/traph/experiments/`
21. Trapnell, C., et al.: Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nature Biotechnology **28** (2010) 511–515
22. Trapnell, C., Pachter, L., Salzberg, S.L.: TopHat: discovering splice junctions with RNA-Seq. Bioinformatics **25**(9) (2009) 1105–1111
23. Van Der Heijden, P.G., Cruyff, M., Van Houwelingen, H.C.: Estimating the size of a criminal population from police records using the truncated poisson regression model. Statistica Neerlandica **57**(3) (2003) 289–304
24. Vatinlen, B., Chauvet, F., Chrétienne, P., Mahey, P.: Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths. European Journal of Operational Research **185**(3) (2008) 1390 – 1401
25. Weintraub, A.: A primal algorithm to solve network flow problems with convex costs. Management Science **21**(1) (1974) 87–97
26. Xing, Y., Resch, A., Lee, C.: The multiassembly problem: reconstructing multiple transcript isoforms from EST fragment mixtures. Genome Res **14**(3) (March 2004) 426–441

A A proof of Theorem 1

Proof. (1): Let $v \in V(G)$; by the definition of x, we can write

$$\begin{aligned}\sum_{u\in N_G^-(v)} x_{uv} - \sum_{u\in N_G^+(v)} x_{vu} &= \sum_{N_G^-(v)} (cov(u,v) + x^*_{uv} - x^*_{vu}) - \sum_{u\in N_G^+(v)} (cov(v,u) + x^*_{vu} - x^*_{uv}) \\ &= \sum_{u\in N_G^-(v)} cov(u,v) - \sum_{N_G^+(v,u} cov(v,u) + \\ &+ \sum_{u\in N_G^-(v)} x^*_{uv} + \sum_{u\in N_G^+(v)} x^*_{uv} - \sum_{u\in N_G^-(v)} x^*_{vu} - \sum_{u\in N_G^+(v)} x^*_{vu} \\ &= \sum_{u\in N_G^-(v)} cov(u,v) - \sum_{u\in N_G^+(v)} cov(v,u) + \sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{uv} - \sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{vu}.\end{aligned}$$

Observe that for all edges entering t^* (exiting s^*), their flow equals their capacity, as we have adjusted the exogenous flow of t^* (of s^*) at point 3.(b)ii. (and 3.(c)ii.) in the construction of G^*. We distinguish three cases.

If $\sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{uv} - \sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{vu} > 0$, then $\sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{uv} - \sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{vu} = x^*_{vt^*}$. Since the flow x^* uses the arc (v,t^*) with its maximum capacity, we have that $x^*_{vt^*} = b_{vt^*} = \sum_{u\in N_G^+(v)} cov(v,u) - \sum_{u\in N_G^-(v)} cov(u,v)$, which shows that $\sum_{u\in N_G^-(v)} x_{uv} - \sum_{u\in N_G^+(v)} x_{vu} = 0$, proving the claim. If $\sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{uv} - \sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{vu} < 0$, then $\sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{uv} - \sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{vu} = -x^*_{s^*v}$. Since the flow x^* uses the arc (s^*,v) with its maximum capacity, we have that $x^*_{s^*v} = b_{s^*v} = \sum_{u\in N_G^-(v)} cov(v,u) - \sum_{u\in N_G^+(v)} cov(u,v)$, which again proves the claim. If $\sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{uv} - \sum_{u\in N_G^-(v)\cup N_G^+(v)} x^*_{vu} = 0$, then, by construction there is no edge between v and t^* or s^*, implying, again by construction, that $\sum_{u\in N_G^-(v)} cov(u,v) = \sum_{u\in N_G^+(v)} cov(v,u)$, from which the claim follows.

(2): From the definition of x, we have

$$\sum_{s\in S}\sum_{v\in N_G^+(s)} x_{sv} = \sum_{s\in S}\left(\sum_{v\in N_G^+(s)} (cov(s,v) + x^*_{sv} - x^*_{vs})\right) \tag{3}$$

$$= \sum_{s\in S}\left(\sum_{v\in N_G^+(s)} cov(s,v) + \sum_{v\in N_G^+(s)} x^*_{sv} - \sum_{v\in N_G^+(s)} x^*_{vs}\right) \tag{4}$$

By construction, since $q_s = 0$ for all $s \in S$, we have $x^*_{st^*} + \sum_{v\in N_G^+(s)} x^*_{sv} = \sum_{v\in N_G^+(s)} x^*_{vs} + x^*_{s_0s}$. Therefore, $\sum_{v\in N_G^+(s)} x^*_{sv} - \sum_{v\in N_G^+(s)} x^*_{vs} = x^*_{s_0s} - x^*_{st^*} = x^*_{s_0s} - b_{st^*} = x^*_{s_0s} - \sum_{v\in N_G^+(s)} cov(s,v)$. Plugging this into (4), we obtain

$$\sum_{s\in S}\sum_{v\in N_G^+(s)} x_{sv} = \sum_{s\in S} x^*_{s_0s} = x^*_{t_0s_0}. \tag{5}$$

Similarly,

$$\sum_{t\in T}\sum_{u\in N_G^-(t)} x_{ut} = \sum_{t\in T}\left(\sum_{u\in N_G^-(t)} (cov(u,t) + x^*_{ut} - x^*_{tu})\right) \tag{6}$$

$$= \sum_{t\in T}\left(\sum_{u\in N_G^-(t)} cov(u,t) + \sum_{u\in N_G^-(t)} x^*_{ut} - \sum_{u\in N_G^-(t)} x^*_{tu}\right) \tag{7}$$

By construction, since $q_t = 0$ for all $t \in T$, we have $x^*_{s^*t} + \sum_{u\in N_G^-(t)} x^*_{ut} = x^*_{tt_0} - \sum_{u\in N_G^-(t)} x^*_{tu}$. Therefore, $\sum_{u\in N_G^-(t)} x^*_{ut} - \sum_{u\in N_G^-(t)} x^*_{tu} = x^*_{tt_0} - x^*_{s^*t} = x^*_{tt_0} - b_{s^*t} = x^*_{tt_0} - \sum_{u\in N_G^-(t)} cov(u,t)$. Plugging this into (7), we prove the claim, since by (5) we get

$$\sum_{t\in T}\sum_{u\in N_G^-(t)} x_{ut} = \sum_{t\in T} x^*_{tt_0} = x^*_{t_0s_0} = \sum_{s\in S}\sum_{v\in N_G^+(s)} x_{sv}. \tag{8}$$

(3): Since any tuple of paths $\mathcal{P} = (P_1, P_2, \ldots, P_k)$ from sources of G to sinks of G, induces a flow on G, where the exogenous flow of all nodes which are not sources nor sinks is zero, and any such flow can be split into paths from sources to sinks, the minimum value of

$$\sum_{(u,v)\in E(G)} f_{uv}\left(\left|cov(u,v) - \sum_{P_i\in\mathcal{P}:\,(u,v)\in P_i} e_i\right|\right), \tag{9}$$

over all k, all k-tuples of paths $\mathcal{P} = (P_1, P_2, \ldots, P_k)$ from a source of G to a sink of G, and over all expression levels e_i for each P_i, is equal to $\min_{y \text{ is a flow on } G} \sum_{(u,v)\in E(G)} f_{uv}\left(|cov(u,v) - y_{uv}|\right)$. Since any flow on G induces a flow on G^*, and vice versa, the above is equal to

$$\min_{z \text{ is a flow on } G^*} \sum_{(u,v)\in E(G)} f_{uv}\left(|z_{uv} - z_{vu}|\right).$$

Since

$$x^* = \operatorname*{argmin}_{z \text{ is a flow on } G^*} \sum_{(u,v)\in E(G)} f_{uv}(z_{uv}) + f_{uv}(z_{vu}), \tag{10}$$

and from minimality, for all arcs $(u,v) \in E(G)$, at most one of z_{uv} or z_{vu} is non null, we have that x^* also attains the minimum in (9), proving the theorem. □