Approximations for the Transcript Expression Problem

Travis Gagie[1], Veli Mäkinen[1], Alexandru Popa[2], and Alexandru I. Tomescu[1]

[1] Helsinki Institute for Information Technology HIIT,
Department of Computer Science, University of Helsinki, Finland
travis.gagie@gmail.com,{vmakinen,tomescu}@cs.helsinki.fi
[2] Department of Communications & Networking,
Aalto University School of Electrical Engineering, Aalto, Finland
alexandru.popa@aalto.fi

Abstract. The recent RNA-Seq technology allows for the sequencing of a large number of short reads from the RNA transcripts of a gene. The assembly of these reads back into the multiple transcripts from which they originated, and the quantification of their abundances, are an important problem for many genomic studies. This problem is usually formalized as follows: given a weighted directed acyclic graph (DAG), find a collection of k paths from a source to a sink, and an integer weight for each path, such that for each edge, its weight is "as close as possible" to the sum of the weights of the paths that traverse it. In this paper we define three new plausible objective functions and present approximation algorithms and hardness results for the corresponding problems. One of these objective functions asks to minimize the maximum ratio between the weight of an edge and the sum of the weights of the paths that cross it. For it, we obtain a Fully Polynomial Time Approximation Scheme (FPTAS).

1 Introduction

For protein synthesis, a gene is first transcribed into precursor mRNA; through RNA splicing, some portions (*introns*) of the precursor mRNA are chopped off and the remaining portions (*exons*) are pasted together to form the mRNA transcripts from which proteins are subsequently translated. Depending on the individual, and on the tissue containing a cell, the splicing process can be different [16], and accordingly, the resulting set of transcripts can be different. Identifying and quantifying them is a basic step in many genomic studies, and has gathered great interest recently [4, 8, 7, 5, 9] thanks in part to the new high-throughput RNA-Seq technology [12, 15, 14].

In an RNA-seq experiment a set of short reads from mRNA transcripts is produced. Even though some *de novo* assembly tools try to assemble the transcripts only from the reads [2], most tools use reference information. This second setting consists of two non-trivial steps. The first is the spliced alignment of the RNA-Seq reads to the reference genome, as solved by [19, 1]. The second problem, which is the one we tackle in this paper, is separating the coverage obtained in the first step into individual transcripts [4, 8, 7, 5, 9].

1.1 Previous work

The difficulty in assigning the reads to different transcripts comes from the fact that the transcripts may have identical exons, and that the length of the reads is generally (much) shorter than the length of the exons. As a result, most methods, such as SLIDE [9], CLIIQ [7] IsoInfer/IsoLasso [4, 8], Scripture [5], rely on a graph model, called *splicing graph* [6]. Its nodes represent contiguous stretches of DNA uninterrupted by spliced reads, while its edges are derived from reads whose prefix aligns to one exon and whose suffix aligns to another. Each node and edge has an associated observed coverage.

The biological multi-assembly and quantification problem becomes the one of covering the graph with intersecting paths, under different cost models, such as the least sum of absolute differences. Most of these tools work by exhaustively enumerating all possible paths, with some restrictions, and then estimating their fitness with an integer linear program, or a quadratic program. In [18] we formalize these approaches through the following problem and show that it has better accuracy in practice.

Problem 1. Given a DAG $G = (V, E)$ and a weight function $w : E \to \mathbb{Z}_+$, find a tuple $\mathcal{P}$ of paths in G starting in some source and ending in some sink, and for each $P \in \mathcal{P}$ a weight $w(P)$, which minimize

$$\sum_{(u,v)\in E} \Big| w(u,v) - \sum_{\substack{P\in\mathcal{P} \\ \text{s.t. } (u,v)\in P}} w(P) \Big|.$$

In the same work [18], we show that Problem 1 can be solved in polynomial time by reducing it to a minimum cost flow problem, with linear cost functions. The reduction works by finding the optimal flow, and then splitting this flow into at most $|E|$ paths.

However, for practical reasons, one is interested in parsimoniously explaining the given DAG with paths, since a small fraction of the graph may be erroneous and should not be explained by transcripts. This can be due to various biological events or technical errors, like template switching, self-priming, intron retention, reading errors, or wrong splicing alignment [3, 14, 10, 11]. Thus, it is natural to impose a bound k on the number of paths in an optimal solution [17] and study the following problem.

Problem 2. Given a DAG $G = (V, E)$, a weight function $w : E \to \mathbb{Z}_+$, and an integer k, find a k-tuple $\mathcal{P} = (P_1, \ldots, P_k)$ of paths in G starting in some source and ending in some sink, and for each P_i a weight w_i, which minimize

$$\sum_{(u,v)\in E} \Big| w(u,v) - \sum_{\substack{i\in\{1,\ldots,k\} \\ \text{s.t. } (u,v)\in P_i}} w_i \Big|.$$

Tomescu et al. [17] show that Problem 2 is NP-hard in the strong sense by a reduction from the 3-Partition problem. In practice, it is generally safe to assume the k is small, since a gene has few transcripts.

1.2 New problem formulations and results

Given the practical importance of the problem, and the recent interest it has gathered, we propose three new, different, objective functions which lead to three different problems.

In the first problem we consider, for each edge, the absolute difference between the sum of the weights of the paths that go through that edge and the weight of the edge. The goal is to minimize the maximum such difference.

Problem 3 (MA-L_∞). Given a DAG $G = (V, E)$, a weight function $w : E \to \mathbb{Z}_+$, and an integer k, find a k-tuple $\mathcal{P} = (P_1, \ldots, P_k)$ of paths in G starting in some source and ending in some sink, and for each P_i a weight w_i, which minimize

$$\mathsf{errMA\text{-}L}_\infty(G, \mathcal{P}) := \max_{(u,v)\in E} \Big| w(u,v) - \sum_{\substack{i\in\{1,\ldots,k\} \\ \text{s.t. } (u,v)\in P_i}} w_i \Big|.$$

In the second problem, we consider the ratio between the weight of an edge and the sum of the weights of the paths that go through it. Observe that we now require that all edges are covered by at least one path, and thus for certain values of k the problem has no solution.

Problem 4 (MA-ratio). Given a DAG $G = (V, E)$, a weight function $w : E \to \mathbb{Z}_+$, and an integer k, find, if it exists, a k-tuple $\mathcal{P} = (P_1, \ldots, P_k)$ of paths in G starting in some source and ending in some sink, and for each P_i a weight w_i, such that each edge of G is covered by at least one path, and $\mathcal{P}$ minimize

$$\mathsf{errMA\text{-}r}(G, \mathcal{P}) := \max_{(u,v)\in E} \max \Big\{ \frac{w(u,v)}{\sum_{\substack{i\in\{1,\ldots,k\} \\ \text{s.t. } (u,v)\in P_i}} w_i}, \Big(\frac{w(u,v)}{\sum_{\substack{i\in\{1,\ldots,k\} \\ \text{s.t. } (u,v)\in P_i}} w_i} \Big)^{-1} \Big\}.$$

The **NP**-hardness proof in [17] for Problem 2 can be adapted easily to show that Problems MA-L_∞ and MA-ratio are also **NP**-hard in the strong sense. In that reduction, given an instance I for the 3-Partition problem, we constructed an instance G_I for Problem 2 such that I was a "yes" instance iff G_I admited a $3q$-tuple of paths having a cost of the objective function 0. Thus, on the same instance G_I we can also require a $3q$-tuple of paths having a cost of the objective function 0, for Problem MA-L_∞, and cost 1, for MA-ratio. This also implies:

Corollary 1. *Problem MA-L_∞ is hard to approximate within any factor $c > 0$.*

In Section 2 we show that Problem MA-L_∞ can be solved in time $O(W^k n^k \Delta^k)$, where W denotes the maximum weight of an edge; we then show how, for every $\varepsilon > 0$, this algorithm can be transformed into an εW-additive approximation algorithm, running in time $O((\frac{1}{\varepsilon}W)^k n^k \Delta^k)$. In Section 3 we show that Problem MA-ratio can be solved in time $O(W^k k^k n + m)$, and that, for every $\varepsilon > 0$, this can be transformed into a $(1+\varepsilon)$-approximation algorithm running in time $O((\log_{1+\varepsilon} W)^k k^k n + m)$. If we assume k to be bounded, this implies the existence of a Fully Polynomial-Time Approximation Scheme (FPTAS) for Problem MA-ratio.

The third problem has the goal of maximizing the number of edges that are *satisfied* by paths, an edge being called *satisfied* if its weight is equal to the sum of the weights of the paths crossing it.

Problem 5 (MA-cover). Given a DAG $G = (V, E)$, a weight function $w : E \to \mathbb{Z}_+$, and integer k, find a k-tuple $\mathcal{P} = (P_1, \ldots, P_k)$ of paths in G starting in some source and ending in some sink, and a tuple $(w_1, \ldots, w_k)$ of integer weights for the paths, which maximizes the number of edges $(u, v) \in E$ for which

$$w(u, v) = \sum_{i \in \{1,\ldots,k\} \text{ such that } (u,v) \in P_i} w_i.$$

Our contribution pertaining to Problem MA-cover is to show that it is **NP**-hard; the proof is given in Section 4.

2 MA-L_∞

In this section we propose a dynamic programming algorithm for Problem MA-L_∞; this closely follows the dynamic programming algorithm given in [17]. Our strategy is to find the optimal k-tuple of paths having a fixed k-tuple of weights. If W denotes the maximum weight of an edge, then the solution is obtained by exhaustively enumerating all k-tuples of weights less than or equal to W, and taking the k-tuple of paths that optimizes objective function.

Let us fix, hereafter, on one choice $(w_1, \ldots, w_k)$ of weights. The main difficulty behind the algorithm is that the paths can share vertices. Accordingly, we have to process all k-tuples of vertices of V. For every $(v_1, \ldots, v_k) \in V^k$ we define

$$\mathsf{solution}(v_1, \ldots, v_k) := \min_{\substack{\text{paths } P_1, \ldots, P_k \text{ in } G_{v_1,\ldots,v_k}, \\ \text{each } P_i \text{ is from a source to } v_i}} \mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,v_k}, (P_1, \ldots, P_k)), \tag{1}$$

where $G_{v_1,\ldots,v_k}$ denotes the subgraph of G induced by the vertices from which there is a directed path to one of $v_1, \ldots, v_k$. Since G is acyclic, then $G_{v_1,\ldots,v_k}$ is acyclic; thus $G_{v_1,\ldots,v_k}$ must have at least one sink. Moreover, from the definition, the sinks of $G_{v_1,\ldots,v_k}$ are among the nodes $v_1, \ldots, v_k$, and the sources of $G_{v_1,\ldots,v_k}$ are also sources in G.

The solution to Problem MA-L_∞ with a given number k of paths, on input G having S as the set of sources and T as the set of sinks, is obtained by computing the entire table $\mathsf{solution}$ for every tuple $(w_1, \ldots, w_k)$ of weights less than or equal to W, and taking

$$\min_{(t_1,\ldots,t_k) \in T^k} \max \left\{ \mathsf{solution}(t_1, \ldots, t_k), \max_{(w,z) \in E(G) \setminus E(G_{t_1,\ldots,t_k})} w(w, z) \right\}. \tag{2}$$

The pseudo-code showing how to recursively compute $\mathsf{solution}$ is depicted as Algorithm 1, where $N^-(v)$ denotes the in-neighborhood of a node v. The following theorem shows its correctness and bounds its running time.

Algorithm 1: Computing $\mathsf{solution}(v_1, \ldots, v_k)$, for a fixed tuple $(w_1, \ldots, w_k)$ of weights for Problem MA-L_∞

foreach $(s_1, \ldots, s_k) \in S^k$ **do** $\mathsf{solution}(s_1, \ldots, s_k) \leftarrow 0$;

$\mathsf{solution}(v_1, \ldots, v_k)$

- $min \leftarrow \infty$;
- let v^* be a sink of $G_{v_1,\ldots,v_k}$ which is not a source of G;
- let $i_1, \ldots, i_\ell$ be all the positions in the tuple $(v_1, \ldots, v_k)$ where v^* appears;
- `/* we enumerate through all ℓ-tuples of in-neighbors of v* */`
- **foreach** $(u_{i_1}, \ldots, u_{i_\ell}) \in N^-(v^*)^\ell$ **do**
  - `/* we get the optimal cost for such a tuple */`
  - let $v_1, \ldots, u, \ldots, v_k$ denote the tuple $v_1, \ldots, v_{i_1-1}, u_{i_1}, v_{i_1+1}, \ldots, v_{i_\ell-1}, u_{i_\ell}, v_{i_\ell+1}, \ldots, v_k$;
  - $error \leftarrow \mathsf{solution}(v_1, \ldots, u, \ldots, v_k)$;
  - $new\_edges \leftarrow \{(u_{i_1}, v^*), \ldots, (u_{i_\ell}, v^*)\}$;
  - $error \leftarrow \max\Big\{error, \max\limits_{j \in \{1,\ldots,\ell\}} \Big| w(u_{i_j}, v^*) - \sum\limits_{\substack{t \in \{j,\ldots,\ell\} \\ \text{s.t. } u_{i_t} = u_{i_j}}} w_{i_t} \Big| \Big\}$;
  - `/* we check the error due to the new uncovered edges appearing in` $G_{v_1,\ldots,v_k}$`, that is, of the edges in` $E(G_{v_1,\ldots,v_k}) \setminus \big(E(G_{v_1,\ldots,u,\ldots,v_k}) \cup new\_edges\big)$ `*/`
  - $error \leftarrow \max\Big\{error, \max\limits_{(w,z) \in E(G_{v_1,\ldots,v_k}) \setminus \big(E(G_{v_1,\ldots,u,\ldots,v_k}) \cup new\_edges\big)} w(w,z)\Big\}$;
  - **if** $error < min$ **then** $min \leftarrow error$;
- **return** min.

Theorem 1. *Problem MA-L_∞ can be solved in time $O(W^k(n^2 + \Delta^k)n^k)$, where $n := |V(G)|$, we assume that W is the maximum weight of an edge, and the maximum in-degree of G is Δ.*

Proof. Since the input graph $G = (V, E)$ is a DAG, we let $\prec$ be a topological order on V, and define the partial order $\prec^k$ on V^k as follows:

$$(v'_1, \ldots, v'_k) \prec^k (v_1, \ldots, v_k) \text{ if and only if } \exists i \in \{1, \ldots, k\} \text{ such that } v'_i \prec v_i.$$

Then, the computation of $\mathsf{solution}$ is done by dynamic programming, by enumerating the tuples $(v_1, \ldots, v_k) \in V^k$ in the order $\prec^k$, and computing $\mathsf{solution}(v_1, \ldots, v_k)$ from the previous values, according to $\prec^k$, as indicated in Algorithm 1. We now show its correctness.

Let $(P_1, \ldots, P_k)$ be a tuple of k *optimal* paths from sources to $v_1, \ldots, v_k$, respectively, i.e.,

$$\mathsf{errMA\text{-}L}_\infty(G, (P_1, \ldots, P_k)) = \min_{\substack{\text{paths } Q_1, \ldots, Q_k \text{ in } G_{v_1,\ldots,v_k}, \\ \text{each } Q_i \text{ is from a source to } v_i}} \mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,v_k}, (Q_1, \ldots, Q_k)). \tag{3}$$

Let v^* be a sink of $G_{v_1,\ldots,v_k}$ which is not a source of $G_{v_1,\ldots,v_k}$ (if no such node exists, then all of $v_1, \ldots, v_k$ are sources in G, and $\mathsf{solution}(v_1, \ldots, v_k)$ has already been initialized). Let $i_1, \ldots, i_\ell$, $\ell \geqslant 1$, be the positions in the tuple $(v_1, \ldots, v_k)$ where v^* appears.

Let $u_{i_1}, \ldots, u_{i_\ell}$ be the predecessors of v^* on $P_{i_1}, \ldots, P_{i_\ell}$, respectively. For every $j \in \{1, \ldots, \ell\}$, denote by $P^*_{i_j}$ the path P_{i_j} from which we remove its

last node, v^*. To simplify notation, denote by $(P_1, \ldots, P^*, \ldots, P_k)$ the tuple $(P_1, \ldots, P_k)$ in which, for every $j \in \{1, \ldots, \ell\}$, we replace P_{i_j} by $P^*_{i_j}$. Similarly, we denote by $(v_1, \ldots, u, \ldots, v_k)$ the tuple $(v_1, \ldots, v_k)$ in which, for every $j \in \{1, \ldots, \ell\}$, we replace v^* by u_{i_j}.

From the fact that v^* is a sink of $G_{v_1,\ldots,v_k}$, neither the node v^*, nor the edges in $new\_edges := \{(u_{i_1}, v^*), \ldots, (u_{i_\ell}, v^*)\}$, belong to any path in G ending in $\{v_1, \ldots, v_k\} \setminus \{v^*\}$. Therefore, we can write:

$$\begin{aligned}\mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,v_k}, (P_1, \ldots, P_k)) = \max\Big\{ & \\ & \mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,u,\ldots,v_k}, (P_1, \ldots, P^*, \ldots, P_k)), \\ & \max_{j \in \{1,\ldots,\ell\}} \Big| w(u_{i_j}, v^*) - \sum_{\substack{t \in \{j,\ldots,\ell\} \\ \text{s.t. } u_{i_t} = u_{i_j}}} w_{i_t} \Big|, \max_{\substack{(w,z) \in E(G_{v_1,\ldots,v_k}) \setminus \\ \left(E(G_{v_1,\ldots,u,\ldots,v_k}) \cup new\_edges\right)}} w(w,z) \Big\}. \end{aligned} \tag{4}$$

Relation (4) holds because, on the one hand, from the definition and acyclicity of $G_{v_1,\ldots,v_k}$, any node or edge of $G_{v_1,\ldots,u,\ldots,v_k}$ not covered by one of the k optimal paths in $G_{v_1,\ldots,u,\ldots,v_k}$ remains uncovered even by adding node v^*. On the other hand, the total error must confront the error due to the new uncovered edges appearing in $G_{v_1,\ldots,v_k}$, that is, of the edges in $E(G_{v_1,\ldots,v_k}) \setminus (E(G_{v_1,\ldots,u,\ldots,v_k}) \cup new\_edges)$; this is accomplished by the two final values.

Let $(P'_1, \ldots, P'_k)$ be a tuple of k optimal paths from a source to $(v_1, \ldots, u, \ldots, v_k)$ such that $\mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,u,\ldots,v_k}, (P'_1, \ldots, P'_k)) = \mathsf{solution}(v_1, \ldots, u, \ldots, v_k)$. From the fact that the paths $P_1, \ldots, P^*, \ldots, P_k$ also end in $v_1, \ldots, u, \ldots, v_k$, and the optimality of $(P'_1, \ldots, P'_k)$, we have

$$\mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,u,\ldots,v_k}, (P_1, \ldots, P^*, \ldots, P_k)) \geqslant \mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,u,\ldots,v_k}, (P'_1, \ldots, P'_k)). \tag{5}$$

From (5) and (4) we get

$$\begin{aligned}\mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,v_k}, (P_1, \ldots, P_k)) \geqslant \max\Big\{ & \\ & \mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,u,\ldots,v_k}, (P'_1, \ldots, P'_k)), \\ & \max_{j \in \{1,\ldots,\ell\}} \Big| w(u_{i_j}, v^*) - \sum_{\substack{t \in \{j,\ldots,\ell\} \\ \text{s.t. } u_{i_t} = u_{i_j}}} w_{i_t} \Big|, \max_{\substack{(w,z) \in E(G_{v_1,\ldots,v_k}) \setminus \\ \left(E(G_{v_1,\ldots,u,\ldots,v_k}) \cup new\_edges\right)}} w(w,z) \Big\}. \end{aligned} \tag{6}$$

We denote by $(P'_1, \ldots, P' \cup \{v\}, \ldots, P_k)$ the tuple $(P'_1, \ldots, P'_k)$ in which, for every $j \in \{1, \ldots, \ell\}$, we add the node v^* at the end of path P_{i_j}. Resorting again to the fact that the node v^* and the edges in new_edges do not belong to any path in $G_{v_1,\ldots,u,\ldots,v_k}$ ending in $\{v_1, \ldots, v_k\} \setminus \{v^*\}$, we get that the right-hand side of the inequality (6) is equal to $\mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,v_k}, (P'_1, \ldots, P' \cup \{v\}, \ldots, P'_k))$, thus:

$$\mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,v_k}, (P_1, \ldots, P_k)) \geqslant \mathsf{errMA\text{-}L}_\infty(G_{v_1,\ldots,v_k}, (P'_1, \ldots, P' \cup \{v\}, \ldots, P'_k)). \tag{7}$$

To conclude, if we enumerate through all $(v'_{i_1}, \ldots, v'_{i_\ell}) \in N^-(v^*)^\ell$, we will find the nodes $u_{i_1}, \ldots, u_{i_\ell}$ preceding v^* on the optimal paths $P_{i_1}, \ldots, P_{i_\ell}$, respectively. If we extend each optimal path ending in $v_1, \ldots, u, \ldots, v_k$ by the node v^*, we obtain paths $P'_1, \ldots, P' \cup \{v^*\}, \ldots, P'_k$, tuple whose cost is optimal, by (3) and (7). If we store in $\mathsf{solution}(v_1, \ldots, v_k)$ also the predecessors $u_1, \ldots, u_k$ of $v_1, \ldots, v_k$ on the optimal paths from some source, we can then trace back the k optimal paths from sources to sinks.

The time complexity bound follows from the fact that there are n^k tuples $(v_1, \ldots, v_k) \in V^k$, computing the sinks of $G_{v_1,\ldots,v_k}$ takes time linear in the number of edges of $G_{v_1,\ldots,v_k}$, which are $O(n^2)$, and there are at most Δ^k candidate predecessors $u_{i_1}, \ldots, u_{i_\ell}$ of v^* on $P_{i_1}, \ldots, P_{i_\ell}$ in the in-neighborhood of v^*; the current best cost to reach $v_1, \ldots, v_k$ from the sources can be computed from the cost $\mathsf{solution}(v_1, \ldots, u, \ldots, v_k)$ in time $O(k^2)$. □

We now show how we can transform the dynamic programming algorithm to obtain an additive approximation algorithm.

In order to reduce the dependency on W, we make sparse the weights up to W. Given $\varepsilon > 0$, let

$$W' := \{1, \varepsilon W, 2\varepsilon W, \ldots, W\}.$$

For every tuple $(w_1, \ldots, w_k) \in W^k$ of exact weights for the k paths, there exists a tuple of weights $(w'_1, \ldots, w'_k) \in (W')^k$ such that for every $1 \leqslant i \leqslant k$, $|w_i - w'_i| \leqslant \varepsilon W$.

If C is the value of the objective function for the optimal paths of weight $(w_1, \ldots, w_k)$, and C' is the value of the objective function for the optimal paths of weight $(w'_1, \ldots, w'_k)$, then it holds that

$$C - \varepsilon W \leqslant C' \leqslant C + \varepsilon W.$$

Therefore, in order to find an εW-additive approximation of the optimal solution for Problem MA-L_∞, we follow the proof of Theorem 1, but whenever we enumerate over k-tuples in W^k, we now enumerate through k-tuples in $(W')^k$. Therefore, the following corollary holds:

Corollary 2. *If the optimal solution for problem MA-L_∞ has cost OPT, then for every $\varepsilon > 0$, in time $O((\frac{1}{\varepsilon}W)^k(n^2+\Delta^k)n^k)$, where W is the maximum weight of an edge, we can find a solution of cost OPT', such that $OPT - \varepsilon W \leqslant OPT' \leqslant OPT + \varepsilon W$.*

3 MA-ratio

Since in Problem MA-ratio all edges of the graph must be covered by at least one of the k paths, if the graph admits a solution, then it must have a simple structure. This leads to a simpler version of the dynamic programming algorithm shown in Section 2, so that the optimal tuple of paths with a given tuple of weights can be computed in time $O(k^k n + m)$.

We start by defining the *rank* of a vertex x in a DAG G as the length of a longest directed path from x to a sink of G. The rank of every vertex of a DAG can be computed in time $O(m)$, by doing a topological sort of the vertices of G, and then assigning rank 0 to the sinks, and

$$rank(x) = 1 + \max_{y \in N^+(x)} rank(y),$$

to all vertices x processed in topological order, where $N^+(x)$ denotes the out-neighborhood of x.

Given a DAG G, let $\tilde{G}$ denote the DAG obtained from G as follows. For every edge (u, v) such that $rank(u) > rank(v)+1$, subdivide (u, v) into as many edges as there are ranks between $rank(u)$ and $rank(v)$. Stated formally, remove edge (u, v), and add new nodes $w_1, \ldots, w_{rank(u)-rank(v)-1}$, and edges (u, w_1), (w_1, w_2), $\ldots$,$(w_{rank(u)-rank(v)-1}, v)$. Observe that the endpoints of every edge of $\tilde{G}$ will have consecutive ranks.

The following lemma places a bound on the number of vertices of each rank in $\tilde{G}$, when G can be covered by k paths.

Lemma 1. *Let s_r be the number of sources of rank smaller than r in a DAG. Given a DAG G without isolated vertices, if the edges of G can be covered by k paths, then for every $r \geqslant 0$ there are at most $k - s_r$ vertices of rank r in $\tilde{G}$.*

Proof. First observe that there can be no directed path between two vertices of the same rank, by the definition of rank.

Assume, for a contradiction, that there exists a set X of at least $k - s_r + 1$ vertices of $\tilde{G}$ having the same rank r. Since each of the s_r sources of rank strictly smaller than r has to be covered by a distinct path not passing through X, by the definition of rank, the vertices of X can be covered by at most $k - s_r$ paths.

Since G has no isolated vertices, then $\tilde{G}$ has no isolated vertices, and thus every vertex x in X has at least one in-coming or out-going edge e_x. Edge e_x can be covered only by a directed path passing through x. There can be no directed path passing through two vertices in X; therefore, we have that each of the at least $k - s_r + 1$ vertices of X has to be covered by a distinct path, a contradiction. □

We show first an exact dynamic programming algorithm for Problem MA-ratio and then explain what changes are needed to obtain a $(1+\varepsilon)$-approximation algorithm.

Theorem 2. *Problem MA-ratio can be solved in time $O(W^k k^k n + m)$, where W is the maximum weight of an edge on the input DAG G with n vertices and m edges.*

Proof. Given an input DAG G, we construct the graph $\tilde{G}$; denote by r_{max} the maximum rank of $\tilde{G}$. We further transform $\tilde{G}$ by making it such that all its sources have rank r_{max}. For each source s of rank $r_s < r_{max}$, we can add a directed path of length $r_{max} - r_s$ ending in s, whose starting point, thus, has

rank r_{max}, and can take care in the algorithm that its edges are not taken into account in the objective function. Since there are at most k sources, and by Lemma 1, the resulting graph has at most k vertices at each rank.

For every rank $r \geqslant 0$ of $\tilde{G}$, and for every tuple $(v_1, \ldots, v_k)$ of vertices of rank r of $\tilde{G}$, and every k-tuple $(w_1, \ldots, w_k) \in W^k$ of weights for the k paths, we store a table $\mathsf{solution}_r$, such that

$$\mathsf{solution}_r(v_1, \ldots, v_k, w_1, \ldots, w_k) := \min_{\substack{\text{paths } P_1, \ldots, P_k \text{ in } G, \\ \text{each } P_i \text{ is from a source to } v_i, \text{ of weight } w_i}} \mathsf{errMA\text{-}r}(\tilde{G}_{v_1,\ldots,v_k}, (P_1, \ldots, P_k)),$$

and $\mathsf{solution}_r(v_1, \ldots, v_k, w_1, \ldots, w_k) := \infty$ if no k paths from the sources to $v_1, \ldots, v_k$, respectively, exist.

The solution will be obtained by taking the minimum over all k-tuples of sinks, and all k-tuples of weights in W^k. If this value is ∞, then no solution exists. In tables $\mathsf{solution}_r$, we can also store the predecessors of the k-tuples of endpoints on the optimal paths, in order to retrieve these paths.

We initialize the table $\mathsf{solution}_{r_{max}}$ with 1, for every k-tuple of vertices of rank r_{max} and every tuple of weights in W^k. For every rank i, $r_{max} > i \geqslant 1$ in decreasing order, we initialize the table $\mathsf{solution}_i$ with ∞ entries, and compute it as follows.

Observe that the set E_i of edges between ranks $i+1$ and i has cardinality at most k, from the fact that G is acyclic and by the definition of rank, as in the proof of Lemma 1. We enumerate through all the ways of assigning the edges in E_i to the k paths, that is, through all k-tuples $(e_1, \ldots, e_k) \in E_i^k$ where $e \in E_i$ equals some e_j if e is assigned to the jth path. Note that this assignment induces an assignment $(u_1, \ldots, u_k)$ of the vertices of rank $i+1$ to the k paths, and an assignment $(v_1, \ldots, v_k)$ of the vertices of rank i to the k paths.

We also enumerate through all k-tuples $(w_1, \ldots, w_k) \in W^k$ of weights for the k paths. For each edge $e \in E_i$ we can then compute its coverage by looking at to which paths it belongs in the assignment $(e_1, \ldots, e_k)$, and which are the weights of the corresponding paths in the assignment $(w_1, \ldots, w_k)$. We update $\mathsf{solution}_i(v_1, \ldots, v_k, w_1, \ldots, w_k)$ with the maximum between $\mathsf{solution}_{i+1}(u_1, \ldots, u_k, w_1, \ldots w_k)$, and the ratios between the weight of the edges in E_i and their coverage with the assignment $(e_1, \ldots, e_k)$ with weights $(w_1, \ldots, w_k)$.

Graph $\tilde{G}$ can be constructed in time $O(m)$. The update at each rank takes time $O(W^k k^k)$; since the maximum rank in $\tilde{G}$ is at most the maximum rank in G, which is n, then the entire procedure takes time $O(W^k k^k n + m)$. □

In order to reduce the dependency on W^k, we make sparse the weights W. Given $\varepsilon > 0$, let

$$W' := \{1, (1+\varepsilon), (1+\varepsilon)^2, \ldots, (1+\varepsilon)^{\lfloor \log_{1+\varepsilon} W \rfloor}\}.$$

For every tuple $(w_1, \ldots, w_k) \in W^k$ of exact weights for the k paths, there exists a tuple of weights $(w'_1, \ldots, w'_k) \in (W')^k$ such that for every $1 \leqslant i \leqslant k$,

$$w_i \leqslant w'_i \leqslant (1+\varepsilon) w_i.$$

If C is the value of the objective function for the optimal paths of weight $(w_1, \ldots, w_k)$, and C' is the value of the objective function for the optimal paths of weight $(w'_1, \ldots, w'_k)$, then it holds that

$$\frac{1}{1+\varepsilon} C \leqslant C' \leqslant (1+\varepsilon) C.$$

Therefore, in order to find a $(1+\varepsilon)$-approximation of the optimal solution for Problem MA-ratio, we follow the proof of Theorem 2, but whenever we enumerate over k-tuples in W^k, we now enumerate through k-tuples in $(W')^k$. Therefore, the following corollary holds:

Corollary 3. *If the optimal solution for problem MA-ratio has cost OPT, then for every $\varepsilon > 0$, in time $O((\log_{1+\varepsilon} W)^k k^k n + m)$, where W is the maximum weight of an edge, we can find a solution of cost OPT', such that $1/(1+\varepsilon)OPT \leqslant OPT' \leqslant (1+\varepsilon)OPT$. Thus, when k is bounded, Problem MA-ratio admits an FPTAS, since the running time is then polynomial in n, $1/\varepsilon$ and $\log W$.*

4 MA-cover

In this section we prove that the MA-cover problem is **NP**-hard via a reduction from a variant of the Subset Sum Problem, defined below.

Definition 1 (k Subset Sum). *Given a set of n integers $A = \{a_1, a_2, \ldots, a_n\}$ and two integers B and k, decide if there is a subset of A with precisely k elements that sum to B.*

Notice that the k Subset Sum Problem is more general than the Subset Sum Problem, thus, it is also NP-hard.

Theorem 3. *Problem MA-cover is NP-hard.*

Proof. Given an instance $(A = a_1, ..., a_n, B, k)$ of the k Subset Sum Problem, we construct the DAG $G_{A,B,k}$ as follows:

1. we add a directed path $b_1, ..., b_{n+2}$ (of length $n+1$), where each edge has weight B.
2. we add nodes $a_1, ..., a_n$ and, for every $i \in \{1, \ldots, n\}$, we add an edge (b_{n+2}, a_i) of weight a_i.

We prove that a k Subset Sum instance $(A = a_1, ..., a_n, B, k)$ is a "yes" instance if and only if $G_{A,B,k}$ admits k paths which satisfy exactly $n+1+k$ edges.

Assume that we have $a'_1, \ldots a'_k \subseteq \{a_1, \ldots, a_n\}$ such that $a'_1 + a'_2 + \cdots + a'_k = B$. We select the k paths to be from b_1 to the vertices in $G_{A,B,k}$ corresponding to $a'_1, a'_2, \ldots, a'_k$. Thus, the k paths cover the edges between b_{n+2} and $a'_1, a'_2, \ldots, a'_k$ and, moreover, since the sum of the weights of the k paths is precisely B, then

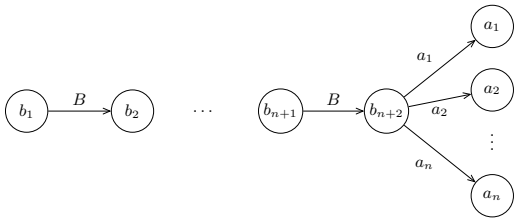

Fig. 1. The reduction from the k Subset Sum Problem to the MA-cover Problem.

they cover all of the edges on the path between b_1 and b_{n+2}. Therefore, we cover $n + 1 + k$ edges and the direct implication follows.

We now prove the reverse implication. Observe that the $n + 1$ edges on the directed path $b_1, \ldots, b_{n+2}$ are either all satisfied, or none is satisfied, as they have the same coverage. Since we have $n + 1 + k$ satisfied edges, and there are only n other edges in the graph, we have that the edges between b vertices are all satisfied. Since b_1 is the unique source and all the k paths must start in b_1, the sum of the weights of the k paths is B.

Let $(b_{n+2}, a'_1), \ldots, (b_{n+2}, a'_k)$ be the remaining edges satisfied. Since we have exactly k paths, this implies that among the edges (b_{n+2}, a_i), the edges (b_{n+2}, a'_1), $\ldots, (b_{n+2}, a'_k)$ are the only edges used by a path. Since the sinks of the graph are $a_1, \ldots, a_n$, this means that the sum $a'_1 + \cdots + a'_k$ is precisely the sum of the weights of the k paths, which by the above observation is B. Therefore, $\{a'_1, \ldots, a'_k\}$ is a solution for instance (A, B, k). □

5 Conclusions and future work

The recent RNA-Seq technology allows for new high-throughput ways for transcript identification and quantification. This has attracted great attention, many tools having been developed recently. However, most of them work by exhaustively enumerating all paths in a splicing graph, and then estimating their fitness. In [18, 17] we formalized these problems by giving a polynomial-time algorithm, and a hardness result, if we require a fixed number of transcripts. In this work we investigated three new objective functions, and provided exact and approximations algorithms for them. We plan to implement these algorithms as an extension of our tool Traph [18, 17]. We also plan to apply these multi-assembly methods to other biological settings, such as assembling metagenomics reads into the different genomes present in a mixed sample [13], or in the multi-assembly of viral quasi-species. Also, an interesting research direction is to study approximation algorithms for Problem MA-cover, or to analyse the complexity of Problems MA-L_∞, MA-ratio and MA-cover when k is not fixed.

Acknowledgement This work was partially supported by Academy of Finland under grant 250345 (CoECGR).

References

1. Au, K.F., Jiang, H., Lin, L., Xing, Y., Wong, W.H.: Detection of splice junctions from paired-end RNA-seq data by SpliceMap. Nucleic Acids Res **38**(14) (August 2010) 4570–4578
2. Birol, I., et al.: De novo transcriptome assembly with ABySS. Bioinformatics **25**(21) (November 2009) 2872–2877
3. Brett, D., et al.: Alternative splicing and genome complexity. Nature Genetics **30**(1) (December 2001) 29–30
4. Feng, J., et al.: Inference of isoforms from short sequence reads. In Berger, B., ed.: RECOMB. Volume 6044 of LNCS, Springer (2010) 138–157
5. Guttman, M., et al.: 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
6. Heber, S., et al.: Splicing graphs and EST assembly problem. Bioinformatics **18**(suppl 1) (2002) S181–S188
7. Li, J.J., et al.: Sparse linear modeling of next-generation mRNA sequencing (RNA-Seq) data for isoform discovery and abundance estimation. Proc. of the National Academy of Sciences **108**(50) (2011) 19867–19872
8. Li, W., et al.: IsoLasso: a LASSO regression approach to RNA-Seq based transcriptome assembly. J. Comput. Biol. **18**(11) (2011) 1693–1707
9. Lin, Y.Y., et al.: CLIIQ: Accurate Comparative Detection and Quantification of Expressed Isoforms in a Population. In: Proc. WABI 2012. Volume 7534 of LNCS, Springer (2012) 178–189
10. Maniatis, T., Tasic, B.: Alternative pre-mRNA splicing and proteome expansion in metazoans. Nature **418**(6894) (2002) 236–243
11. McIntyre, L., et al.: RNA-seq: technical variability and sampling. BMC Genomics **12**(1) (June 2011) 293+
12. Mortazavi, A., et al.: Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nature Methods **5** (2008) 621–628
13. 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)
14. Ozsolak, F., Milos, P.M.: RNA sequencing: advances, challenges and opportunities. Nature reviews. Genetics **12**(2) (February 2011) 87–98
15. Pepke, S., Wold, B., Mortazavi, A.: Computation for ChIP-seq and RNA-seq studies. Nature methods **6**(11) (2009) s22–s32
16. Sammeth, M., Foissac, S., Guigó, R.: A General Definition and Nomenclature for Alternative Splicing Events. PLoS Comput Biol **4**(8) (August 2008) e1000147+
17. Tomescu, A.I., Kuosmanen, A., Rizzi, R., Mäkinen, V.: A Novel Combinatorial Method for Estimating Transcript Expression with RNA-Seq: Bounding the Number of Paths. Accepted at the 13th Workshop on Algorithms in Bioinformatics, WABI 2013
18. Tomescu, A.I., Kuosmanen, A., Rizzi, R., Mäkinen, V.: A Novel Min-Cost Flow Method for Estimating Transcript Expression with RNA-Seq. BMC Bioinformatics **14**(Suppl 5) (2013) S15 presented at RECOMB-Seq 2013, Beijing, China.
19. Trapnell, C., Pachter, L., Salzberg, S.L.: TopHat: discovering splice junctions with RNA-Seq. Bioinformatics **25**(9) (2009) 1105–1111