Accurate Flow Decomposition via Robust Integer Linear Programming

Dias, Fernando H.C.; Tomescu, Alexandru I.

2024-09-13
Institute of Electrical and Electronics Engineers Inc.
http://hdl.handle.net/10138/589193
Dias, F H C & Tomescu, A I 2024, 'Accurate Flow Decomposition via Robust Integer Linear Programming', IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 21, no. 6, pp. 1955 - 1964. https://doi.org/10.1109/TCBB.2024.3433523

Accurate Flow Decomposition via Robust Integer Linear Programming

Fernando H. C. Dias, Alexandru I. Tomescu

Abstract—Minimum flow decomposition (MFD) is a common problem across various fields of Computer Science, where a flow is decomposed into a minimum set of weighted paths. However, in Bioinformatics applications, such as RNA transcript or quasi-species assembly, the flow is erroneous since it is obtained from noisy read coverages. Typical generalizations of the MFD problem to handle errors are based on least-squares formulations or modelling the erroneous flow values as ranges. All of these are thus focused on error handling at the level of individual edges. In this paper, we interpret the flow decomposition problem as a robust optimization problem and lift error-handling from individual edges to *solution paths*. As such, we introduce a new *minimum path-error flow decomposition* problem, for which we give an Integer Linear Programming formulation. Our experimental results reveal that our formulation can account for errors significantly better, by lowering the inaccuracy rate by 30–50% compared to previous error-handling formulations, with computational requirements that remain practical.

Index Terms—Flow decomposition, network flow, uncertainty, integer programming

1 Introduction

1.1 Background

In multiassembly problems, multiple genomic sequences in different abundances must be reconstructed from a set of *reads* sequenced from them [33]. Typical examples are the assembly of *RNA transcripts* (or *isoforms*) [17], [34], or the assembly of *viral quasi-species* [7], [29]. A standard approach to this problem first constructs a weighted graph from the reads. For example, in the case of RNA transcripts, one can align the RNA-seq reads to the reference genome and add a graph node for every inferred exon in the reference sequence. Each read spanning two exons leads to an edge between the corresponding nodes, directed from smaller to larger genomic positions, given some fixed orientation of the reference sequence (obtaining thus a *directed acyclic graph*, or *DAG*). Moreover, edges (or nodes, or both) have an associated weight, e.g. the average read coverage of the corresponding genomic positions. Such a graph is called a *splicing graph* [12], and it is used by many tools [13], [17], [23], [30], [31], [32], [34]. The multiassembly problem can then be formulated as finding a set of weighted paths (one path for each of the genomic sequences in the sequencing sample) whose superposition best "explains" the weights of the graph [26], [33]. See Figure 1 for an overview of the multiassembly problem for RNA transcripts.

If the data is perfect (e.g., no read alignment errors), then the edge weights form a *flow*, in the sense that the sum of the edge weights entering a node equals the sum of the edge weights exiting the same node, for every node different than a source or a sink. A *flow decomposition (FD)* is a set of weighted paths whose superposition matches such flow values, i.e., for every edge, the flow value of the edge equals the sum of the weights of the paths passing through it [1].

Previous research on perfect data noted that flow decompositions of *minimum size* (i.e., with a minimum number of paths, *minimum flow decompositions*, or *MFDs*) largely correspond to correct answers to multiassembly problems [2], [3], [10], [15], [22]. For example, on perfect RNA-seq data, MFDs correspond to the ground truth in over 95% of human genes [15], [22], as we also confirm in this paper. As such, a large amount of theoretical research focused on computing an MFD. While this problem is NP-hard [28] and hard to approximate [11], an exponential-factor approximation exists [18], as well as approximation algorithms for some variants where not all of the flow needs to be decomposed [11], or the flow values can also assume

• *Fernando H. C. Dias is with the System Analysis Laboratory in Department of Mathematics and Systems Analysis, Aalto Univeristy, Finland and Alexandru Tomescu is with the Department of Computer Science, University of Helsinki, Finland. E-mail: {fernando.dias@aalto.fi,alexandru.tomescu@helsinki.fi}*

Manuscript received May 31, 2024;

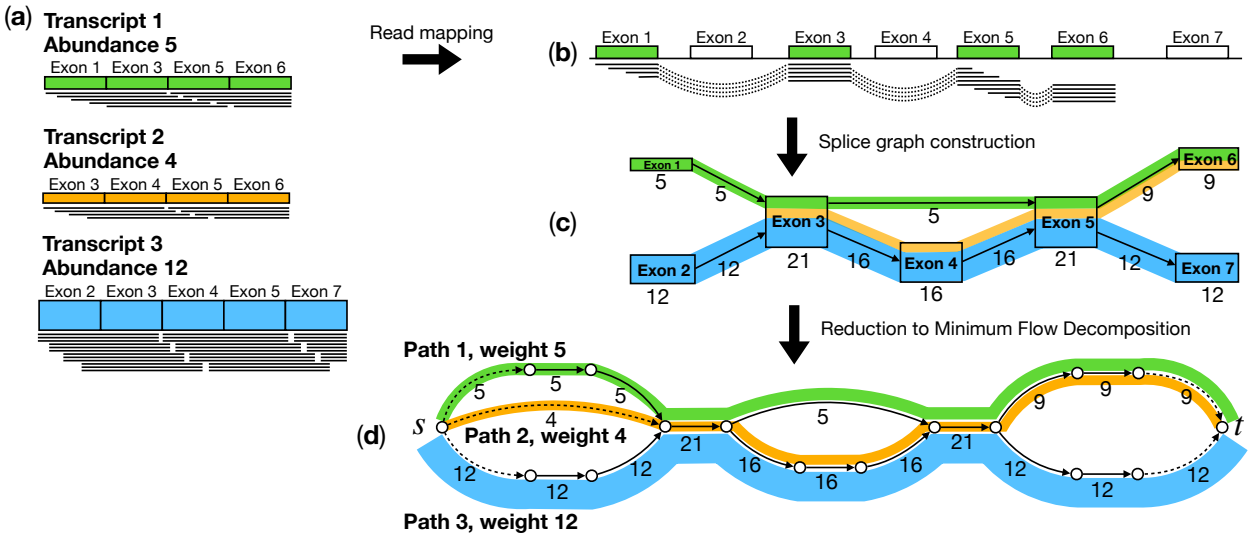


Fig. 1: Overview of the multiassembly problem modelled as a minimum flow decomposition problem. (a) Assume an RNA sample containing three transcripts, of abundance 5, 4 and 12, respectively. Via RNA-seq, we obtain a set of mixed reads in a number proportional to the abundance of each transcript. For illustration purposes, we draw these reads under each transcript, but in practice, we have no means of identifying the originating transcript of each read. For simplicity, we also draw exons as having equal lengths, even though their lengths vary in practice. (b) Assuming a genome reference, we can map the reads to the reference. Reads can map to one exon, two, or more exons. For simplicity, we show only alignments of the reads originating from the first transcript. (c) From all read alignments, one can construct a splicing graph: each exon corresponds to a node, and each read overlapping two exons corresponds to an edge between their corresponding nodes. Edges are always oriented according to the order of the exons in the reference genome; thus, the splicing graph is acyclic. Assuming perfect data and constant sequencing coverage, then each node and edge has an associated weight equal to the sum of the abundance of the transcripts passing through them. For example, Exon 4 appears in Transcripts 2 and 3, and thus its weight is $4+12=16$; likewise, Exon 5 appears in Transcripts 1, 2 and 3, and its weight is $5+4+12=21$. The multiassembly problem asks to reconstruct the three coloured paths in the splicing graph based only on the graph structure and its weights. (d) Under the same assumptions of perfect data, the multiassembly problem can be reduced to Problem 1 by constructing a flow network: we replace each exon with an edge having the same weight as the exon, and we add a global source s with edges to all resulting nodes having more out-going flow than in-coming flow (i.e. starts of transcripts), and a global sink t with edges from all resulting nodes having more in-coming flow than out-going flow (i.e. ends of transcripts). The three paths shown in the graph, of weights 5, 4 and 12, respectively, are a solution to Problem 1.

negative values [6]. Also, heuristic algorithms [23], [28], or a fixed-parameter tractable algorithm (in the size of the MFD) [15] exist. Recently, it was shown that there exists an Integer Linear Programming (ILP) formulation for MFD [8], which is the fastest practical and exact solution to the problem to date.

1.2 Motivation

Since the weights of graphs constructed from real data generally do not form a flow (since e.g. the sequencing coverage is not uniform, see e.g. [21]), practical tools implement generalizations of the MFD problem that account for errors in the edge weights. The most popular one, *least-squares flow decomposition*, asks to find a set of weighted paths that minimise the sum of the squared difference between the weight of each edge and the sum of the weights of the paths passing through it (we call such difference the *superposition error* of an edge). Among all such sets of paths, one with the minimum number of paths is preferred. Many RNA transcript assembly tools use this model, see e.g. [5], [10], [17], [26], [27]. Another more recent variant (*minimum inexact flow decomposition*) is to assume that for each edge, we have an associated interval, and the sum of the weights of the paths passing through the edge must belong to this interval; analogously, among all such solutions, we prefer one of minimum size [30].

These two problems were first solved heuristically. For example, in the case of the former problem, the weights can be first "minimally corrected" so that they become a flow, and then this flow can be heuristically decomposed [5], [10], [17], [27]. For the latter problem, one can compute a "good flow" falling inside the edge intervals and then heuristically decompose this flow. Recently, exact ILP-based solutions for both problems were shown to exist [8], which are also fast in practice.

Even though practical tools may include additional information to the problem statements (e.g., long reads, as in [34], or pre-assembled contigs as in [2], [20]), models similar to least-squares and inexact flow have been central for dealing with weights not forming perfect flows. In this paper, for the first time, we experimentally evaluate the inaccuracy rate of *exact* solutions to the least-squares and minimum inexact flow decomposition models by implementing the exact ILP-based solutions from [8]. On a version of the dataset of [23] where we add errors, we show that the former model has an inaccuracy rate of 20–32% (depending on how stringent the evaluation metric is and how erroneous the edge weights are), while the latter one has an inaccuracy rate of 9–19%. As a baseline, on perfect flows, the inaccuracy rate is 4–5%. These results suggest a large gap between our computational models for erroneous and perfect flows.

1.3 Contributions

In this paper, we propose a new formulation for the flow decomposition problem for dealing with edge weights not forming a flow. We obtain this by interpreting the problem as a robust optimization problem [24], where it is assumed that some decision variables are prone to errors. A common approach is to model such errors as enlarged bounds for the variables [4]. One can interpret the minimum inexact flow decomposition problem as an instance of this approach. Another approach is to include the errors in the objective function. One can interpret the least-squares flow decomposition problem as an instance of this approach. However, these two previous approaches attach errors to *each edge*. From our point of view, this limits the robustness analysis to a *microscopic* management of errors focused on individual edges.

We propose a *macroscopic* management of errors by attaching an error to each *solution path* instead of each edge. In particular, for each solution path, we introduce an *error*, or *slack*, variable and define the *minimum path-error flow decomposition problem* as the problem of finding a set of weighted paths with associated error variables, such that the superposition difference of each edge is within the sum of the error variables of the paths using the edge. Overall, we would like to report a set of such paths minimizing the total path error. Moreover, among all such solutions, we prefer one with a minimum number of paths.

Intuitively, as opposed to the previous two formulations, if an edge belongs to multiple solution paths, then each such path contributes with an allowed error since it is more likely that the observed weight of that edge is noisier if it arises by summing up the read coverage of multiple genomic sequences. However, in the inexact flow formulation, the superimposed paths had to belong to a fixed interval, no matter how many paths passed through that edge. Likewise, in the least-squares formulation, the superposition error of an edge contributes the same amount to the objective function no matter how many solutions paths use the edge.

When the graph weights form a flow, this problem is equivalent to MFD; hence it is NP-hard. However, we show that finding a minimum path-error flow decomposition into k paths can be easily solved via an ILP formulation, with a quadratic number of variables and constraints in the graph size and the number of solution paths, by adapting the ILP formulations from [8]. In the above-mentioned experimental setting, our formulation achieves an inaccuracy rate of 5–11%, just a few percentage points behind the MFD formulation for perfect flows, and running on average in 3 seconds per input weighted graph, and, at the same, each instance was solved in under 7.5 minutes. In the most demanding evaluation scenario, we obtain a relative improvement in inaccuracy rate of more than 40% compared to the best previous formulation handling errors. Therefore, we hope that this problem formulation, enhanced with various additional information employed by practical multiassebly tools (such as long reads, as in [34], and also expressible in ILP [8]), could lie at the core of more accurate multiassembly tools.

The paper is organized as follows. We review the basic concepts of flow network and flow decomposition and define the problems addressed as well as previous methods attempted for flow decomposition under imperfect flow conditions in Sections 2.1 to 2.3. We next present our formulation in Section 3. Experimental results are provided in Section 4, and concluding remarks are discussed in Section 5.

1.4 Preliminary notions

A directed graph (or simply, a *graph*) is a tuple (V, E), with V the set of nodes, and $E \subseteq V \times V$ the set of

edges.[1] As mentioned in Section 1.1 (see also Figure 1), splicing graphs contain no cycles, i.e. are directed acyclic graphs (*DAGs*). Thus, in the rest of this paper we work only with DAGs. We also assume that the DAG has a single *source (s)*, namely a node without incoming edges, and a single *sink (t)*, namely a node without any outgoing edges. A path from s to t is called an *s-t path*. For a path P and an edge $(u, v) \in E$, we let $P(u, v) = 1$, if the edge (u, v) belongs to P, and $P(u, v) = 0$, otherwise.

We also assume to have a positive *flow value* $f_{uv} \in \mathbb{Z}^+$ associated to every edge $(u, v) \in E$, such that the following *flow conservation* condition holds:

$$\sum_{(u,v)\in E} f_{uv} = \sum_{(v,w)\in E} f_{vw}, \quad \forall v \in V \setminus \{s, t\}. \tag{1}$$

Given such a DAG (V, E), with associated flow values f, the tuple $G = (V, E, f)$ is said to be a *flow network*.

Given a flow network G, a set of k s-t paths $(\mathcal{P} = (P_1, \ldots, P_k))$ with associated weights $w = (w_1, \ldots, w_k)$, where each $w_i \in \mathbb{Z}^+$, is said to be a *k-flow decomposition* of G, if the following *flow superposition* condition holds:

$$\sum_{i\in\{1,\ldots,k\}} P_i(u, v)w_i = f_{uv}, \quad \forall (u, v) \in E. \tag{2}$$

The number k of s-t paths is called the *size* of the flow decomposition $\mathcal{P}$ and is also denoted $|\mathcal{P}|$. See Figure 1(d) for an example of a flow network and of a flow decomposition of it.

The minimum flow decomposition problem [28] asks to find a flow decomposition of a given flow network of minimum size. Formally, it is defined as follows.

Problem 1 (Minimum Flow Decomposition)**.** *Given a flow network $G = (V, E, f)$, find a minimum-size set of s-t paths $\mathcal{P} = (P_1, \ldots, P_k)$ with associated weights $(w_1, \ldots, w_k)$, with each $w_i \in \mathbb{Z}^+$, namely, a solution to the following model:*

Minimise $|\mathcal{P}|$
Subject to:
Flow Superposition Condition (2).

Problem 1 admits an Integer Linear Programming formulation, as shown in [8], which works by iterating over k in increasing order and checking whether a flow decomposition of size k exists. It is a standard result that problem always has a feasible solution, see e.g. [1]. For example, can obtain a decomposition of at most $|E|$ paths, by iteratively considering the edge e of smallest flow value $f(e)$, some path P passing through e, and subtracting P with weight $f(e)$ from the flow f.

1. Since $E \subseteq V \times V$, there can be no *parallel* edges with the same endpoints. However, there is a trivial observation that there is no loss of generality because, for flow decomposition problems, such edges can be subdivided by introducing a new node in the middle of each parallel edge.

2 Previous Approaches for Handling Errors

In this section, we review three state-of-the-art approaches for decomposing graphs where the values associated to the edges are arbitrary integers greater than 0, and thus do not satisfy flow conservation (recall Section 1.2).

2.1 Bounded-error Formulation

Let (V, E) be a DAG with unique source s and unique sink t, with *imperfect flow values* $f_{uv} \in \mathbb{Z}^+$ associated to each edge $(u, v) \in E$ (i.e., not necessarily satisfying the flow conservation condition). The tuple $G = (V, E, f)$ is called an *imperfect flow network* [8]. Moreover, assume that we are also given an *error bound* B, which intuitively represents the bound by which the Flow Superposition Condition (2) can be relaxed; see Figure 2a for an illustration.

Formally, we would like to find a minimum-size set of s-t paths $(P_1, \ldots, P_k)$ with associated weights $(w_1, \ldots, w_k)$ such that the following *imperfect flow superposition* condition, with error bound B, holds:

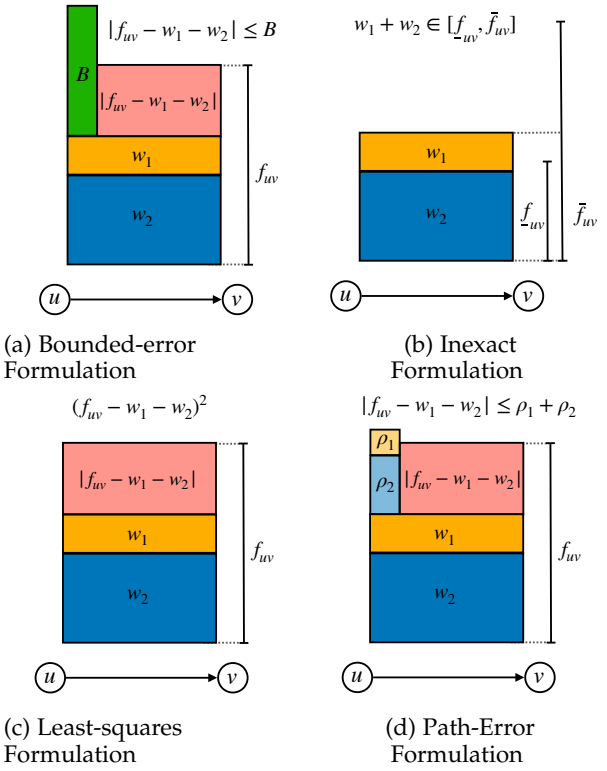
$$\left| f_{uv} - \sum_{i\in\{1,\ldots,k\}} P_i(u, v)w_i \right| \leq B, \quad \forall (u, v) \in E. \tag{3}$$

We call the left-hand term of the above inequality *superposition error*. Formally, we have the following problem.

Problem 2 (Minimum imperfect flow decomposition (bounded-error))**.** *Given an imperfect flow network $G = (V, E, f)$, and an error bound $B \geq 0$, find a set of s-t paths $\mathcal{P} = (P_1, \ldots, P_k)$ with associated weights $(w_1, \ldots, w_k)$, with each $w_i \in \mathbb{Z}^+$, that are a solution to the following model:*

Minimise $|\mathcal{P}|$
Subject to:
Imperfect Flow Superposition Condition with error bound B (3).

Problem 2 is clearly NP-hard, since Problem 1 can be reduced to it by setting $B = 0$. It also admits an



(a) Bounded-error Formulation

(b) Inexact Formulation

(c) Least-squares Formulation

(d) Path-Error Formulation

Fig. 2: Illustration of the behaviour expected for all methods present in this manuscript. In orange and blue, we show the weights w_1 and w_2 of paths P_1 and P_2, respectively, which we assume to pass through the edge (u, v). In red, we show the superposition error, namely, the absolute difference $|f_{uv} - w_1 - w_2|$. Each method uses different techniques to control the amount of superposition error. In Figure 2a, an error bound B (in green) limits the superposition error; in Figure 2b, superposition error is allowed as long as $w_1 + w_2$ falls in a given range of the edge (u, v); in Figure 2c, a minimization of least-squares of the superposition errors is applied. Finally, in the path-error formulation of Figure 2d, the superposition error is not controlled by a global bound or by the edge (u, v) (as in Figures 2a and 2b), but by the sum of paths slacks $\rho_1 + \rho_2$ of the paths P_1 and P_2, in light orange and light blue, respectively.

Integer Linear Programming formulation [8], which analogously checks for the existence of a set of k weighed paths in increasing order on k. Note that in Problem 2, based on the value allocated to the constant B, a feasible solution might not exist.

2.2 Inexact Formulation

The above bounded-error formulation, where the error bound is the same for all edges, has been generalized by [30] to allow for an *error range* specific for each edge. More specifically, we can consider that for each edge $(u, v) \in E$; we have associated two values $\underline{f}_{uv} \leq \overline{f}_{uv}$. Such a tuple $G = (V, E, \underline{f}_{uv}, \overline{f}_{uv})$ is then called an *inexact flow network* [30].

We now ask for a minimum-size set of s-t paths $(P_1, \ldots, P_k)$ with associated weights $(w_1, \ldots, w_k)$ such that the sum of the reported path weights of each edge is in the range $[\underline{f}_{uv}, \overline{f}_{uv}]$; see Figure 2b for an illustration. More precisely, we have the following *inexact flow superposition* condition:

$$\underline{f}_{uv} \leq \sum_{i \in \{1,\ldots,k\}} P_i(u, v) w_i \leq \overline{f}_{uv}, \quad \forall (u, v) \in E. \quad (4)$$

Formally, we have the following problem.

Problem 3 (Minimum inexact flow decomposition)**.** *Given an inexact flow network $G = (V, E, \underline{f}_{uv}, \overline{f}_{uv})$, find a set of s-t paths $\mathcal{P} = (P_1, \ldots, P_k)$ with associated weights $(w_1, \ldots, w_k)$, with each $w_i \in \mathbb{Z}^+$, that are a solution to the following model:*

Minimise $|\mathcal{P}|$

Subject to:

Inexact Flow Superposition Condition (4).

Also Problem 3 is NP-hard, being a generalization of Problems 1 and 2. Likewise, it also admits an Integer Linear Programming formulation [8], which analogously checks for the existence of a set of k weighed paths in increasing order on k. As in the case Problem 2, some input might not admit feasible solutions due to the range bounds.

2.3 Least-squares Formulation

In the previous formulations, the superposition error was bounded by either a global error bound B or by a finite range $[\underline{f}_{uv}, \overline{f}_{uv}]$. Again, suppose we assume an imperfect flow network. In that case, an alternative is to minimise the sum of all superposition errors. For example, as in a popular least-squares formulation used by [17], minimising:

$$\sum_{(u,v) \in E} \left(f_{uv} - \sum_{i \in \{1,\ldots,k\}} P_i(u, v) w_i \right)^2. \quad (5)$$

Moreover, over all such sets of s-t paths $(P_1, \ldots, P_k)$ minimising (5), we prefer one of minimum size k. Formally, we have the following problem.

Problem 4 (Least-squares flow decomposition)**.** *Given an imperfect flow network $G = (V, E, f)$, find a minimum-size set of s-t $\mathcal{P} = (P_1, \ldots, P_k)$ with associated weights*

$(w_1, \ldots, w_k)$, with each $w_i \in \mathbb{Z}^+$, that are a solution to the following model:

Minimise $|\mathcal{P}|$

Subject to:

$$\text{Minimise} \quad \sum_{(u,v)\in E} \left(f_{uv} - \sum_{i\in\{1,\ldots,k\}} P_i(u,v) w_i \right)^2 .$$

Problem 4 is also NP-hard [26] and can be solved via Integer Quadratic Programming (IQP), see [8] for details. This formulation works by trying all k values until $|E|$ and taking the set of k weighted paths having the least-squared error.

3 Minimum Path-Error Flow Decomposition

3.1 Problem Definition

Assume we have an imperfect flow network and consider the superposition error of an edge (u, v). In the bounded and inexact formulations, the superposition error was controlled either by a global error bound or by a range associated to (u, v). In both cases, the bound or the range of (u, v) stays the same no matter the number of s-t paths of the decomposition traversing (u, v). Moreover, individual edges control the errors in all three previous formulations.

The main idea of our new path-error formulation is to move the error control mechanism from individual edges to the flow decomposition paths. As such, we can introduce a non-negative *slack*, or *error*, variable ρ_i for each s-t path P_i and require that the superposition error of (u, v) be at most the sum of the slacks of the s-t paths using (u, v). Thus, since the slack of the path is now associated with the path (and not independently controlled by individual edges), the fact that a path P_i leads to a substantial superposition error in some edges is overall penalised by increasing the needed slack ρ_i of P_i. Formally, we have the following *robust flow superposition* constraint, for each edge $(u, v) \in E$:

$$\left| f_{uv} - \sum_{i\in\{1,\ldots,k\}} P_i(u,v) w_i \right| \leq \sum_{i\in\{1,\ldots,k\}} \rho_i P_i(u,v). \tag{6}$$

Since the path slacks now control the superposition errors, we can ask for a set of paths of minimum total slack, that is, minimising:

$$\min \sum_{i\in\{1,\ldots,k\}} \rho_i. \tag{7}$$

And among all sets of paths of minimum total slack, we prefer one of minimum-size. Formally, we have the following problem (see also Figure 3 for an example).

Problem 5 (Minimum Path-Error Flow Decomposition)**.** *Given an imperfect network $G = (V, E, f)$, find a minimum-size set of s-t paths $\mathcal{P} = (P_1, \ldots, P_k)$ and associated weights $(w_1, \ldots, w_k)$ and slacks $(\rho_1, \ldots, \rho_k)$, with each $w_i \in \mathbb{Z}^+$, $\rho_i \in \mathbb{Z}^+ \cup \{0\}$, that are a solution to the following model:*

Minimise $|\mathcal{P}|$

Subject to:

Minimise $\sum_{i\in\{1,\ldots,k\}} \rho_i$

Subject to:

Robust Flow Superposition Condition (6).

3.2 Integer Linear Programming Formulation

In this section we give an ILP formulation for the following sub-problem, where we fix before-hand the number k of paths that are required to have minimum path error:

Problem 6 (Minimum Path-Error k-Flow Decomposition)**.** *Given an imperfect network $G = (V, E, f)$ and a positive integer k, find a set of k s-t paths $\mathcal{P} = (P_1, \ldots, P_k)$ and associated weights $(w_1, \ldots, w_k)$ and slacks $(\rho_1, \ldots, \rho_k)$, with each $w_i \in \mathbb{Z}^+$, $\rho_i \in \mathbb{Z}^+ \cup \{0\}$, that are a solution to the following model:*

Minimise $\sum_{i\in\{1,\ldots,k\}} \rho_i$

Subject to:

Robust Flow Superposition Condition (6).

To find a solution for Problem 5, one can solve Problem 6 for all k values in increasing order, and select the k value minimising the sum of path slacks. The following lemma gives a loose upper bound on such value of k, and we leave open whether a tighter upper bound exists (e.g. polynomial in the graph size):

Lemma 1. *Let $\mathcal{P} = (P_1, \ldots, P_k)$, with associated weights $(w_1, \ldots, w_k)$ and slacks $(\rho_1, \ldots, \rho_k)$, respectively, be a minimum solution to Problem 5. Then it holds that $k \leq \sum_{i=1}^{k} w_i$.*

Proof. This holds trivially, since by definition each integer weight w_i is positive, and thus $\sum_{i=1}^{k} w_i \geq k$. □

Our ILP formulation for Problem 6 follows the same general structure given by [8] for the MFD problem, which in the present case is as follows: formulate k s-t paths, then model the robust flow superposition constraint, and minimise the sum of path slacks.

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3433523

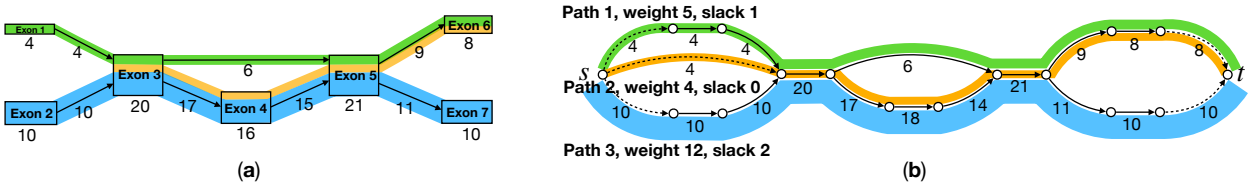


Fig. 3: Example of Minimum Path-Error k-Flow Decomposition. (a) Consider the splicing graph from Figure 1(c), with the difference that the sequencing coverage is not uniform across the transcripts. While the flow network constructed from this graph (in Figure 1(d)) satisfied flow conservation at every node (except s and t), this is no longer the case for the graph in (b). The paths shown in (b) are a solution to Problem 5 (with $k = 3$), and in particular, the Robust Flow Superposition Condition (6) holds for every edge. For example, the edge of weight 14 is traversed by Paths 2 and 3, of weights 4 and 12, and slacks 0 and 2, respectively, and it holds that $|14 - (4 + 12)| \leq 0 + 2$.

Indeed, to model each path P_i, $i \in \{1, \ldots, k\}$, we introduce a set of binary variables x_{uvi} for each edge $(u, v) \in E$. If $x_{uvi} = 1$, the edge (u, v) will be part of P_i, and if $x_{uvi} = 0$, then it is not. Every path P_i is required to start at s (i.e., to have exactly one variable x_{svi} set to 1, for some edge (s, v)) and end at t (to have exactly one variable x_{vti} set to 1, for some edge (v, t)). Any other node of the graph requires equal in- and out-degree in terms of the binary variables x_{uvi} (i.e., 0 or 1). More formally, we add the following constraints for every $v \in V$:

$$\sum_{(u,v)\in E} x_{uvi} - \sum_{(v,u)\in E} x_{vui} = \begin{cases} 0, & \text{if } v \in V \setminus \{s, t\}, \\ 1, & \text{if } v = t, \\ -1, & \text{if } v = s. \end{cases} \tag{8}$$

See [8], [25] for a more detailed explanation of why this is a correct formulation of an s-t path in a DAG.

Alongside these path constraints, we must impose condition (6). By replacing $P_i(u, v)$ with x_{uvi}, the following constraint is obtained:

$$\left| f_{uv} - \sum_{i\in\{1,\ldots,k\}} x_{uvi} w_i \right| \leq \sum_{i\in\{1,\ldots,k\}} \rho_i x_{uvi}, \forall (u, v) \in E. \tag{9}$$

The above constraint contains non-linear terms ($x_{uvi}w_i$ and $\rho_i x_{uvi}$) that standard linear solvers cannot solve. However, using basic linearisation techniques, these terms can be replaced by a set of linear constraints. The product $x_{uvi}w_i$ is a product of two decision variables, one binary and one integer. It can be replaced by a new integer variable $\phi_{uvi} \in \mathbb{Z}^+ \cup \{0\}$, such that:

$$\phi_{uvi} = x_{uvi} w_i, \quad \forall (u, v) \in E, \forall i \in \{1, \ldots, k\}. \tag{10}$$

To guarantee equivalence in this substitution of variables, the following additional constraints are necessary. For all $(u, v) \in E$, and $i \in \{1, \ldots, k\}$:

$$\phi_{uvi} \leq M_1 x_{uvi}, \tag{11a}$$
$$\phi_{uvi} \leq w_i, \tag{11b}$$
$$\phi_{uvi} \geq w_i - (1 - x_{uvi}) M_1, \tag{11c}$$

where M_1 is a sufficiently large constant so that the inequality in (11a) is always satisfied. Correctness can be seen as follows. If $x_{uvi} = 1$, then constraint (11a) holds by the choice of M. Constraint (11b) sets $\phi_{uvi} \leq w_i$, and constraint (11c) sets $\phi_{uvi} \geq w_i$; thus $\phi_{uvi} = w_i = x_{uvi}w_i$. If $x_{uvi} = 0$, then constraint (11a) sets $\phi_{uvi} \leq 0$. Since $\phi_{uvi} \geq 0$, we have $\phi_{uvi} = 0 = x_{uvi}w_i$, and the other two constraints are satisfied.

The terms $\rho_i x_{uvi}$ can be linearised using the same technique by introducing the additional variable $\gamma_{uvi} \in \mathbb{Z}^+ \cup \{0\}$ and ensuring that $\gamma_{uvi} = \rho_i x_{uvi}$. For this case, we will choose (possibly) another significantly large constant M_2 that functions as above. For all $(u, v) \in E$, and $i \in \{1, \ldots, k\}$, the resulting constraints are:

$$\gamma_{uvi} \leq M_2 x_{uvi}, \tag{12a}$$
$$\gamma_{uvi} \leq \rho_i, \tag{12b}$$
$$\gamma_{uvi} \geq \rho_i - (1 - x_{uvi}) M_2. \tag{12c}$$

Remark: The values of M_1 and M_2 are independent.

Finally, the absolute value function in the left-hand of the constraint in Equation (6) can also be expanded in the two separate constraints, for all $(u, v) \in E$:

$$f_{uv} - \sum_{i\in\{1,\ldots,k\}} \phi_{uvi} \leq \sum_{i\in\{1,\ldots,k\}} \gamma_{uvi}, \tag{13a}$$
$$f_{uv} - \sum_{i\in\{1,\ldots,k\}} \phi_{uvi} \geq - \sum_{i\in\{1,\ldots,k\}} \gamma_{uvi}, \tag{13b}$$

This complete ILP model can be found in the

Supplementary Material (see Section I). As we observe in Section 4, running this ILP for larger k values is significantly more expensive. Thus, as a heuristic to find the smallest number of paths attaining minimum path error, we solve this ILP for increasing values of k, and report the solution for that k value such that the minimum path error for $k+1$ does not change. On 1000 random instances in our dataset, this k value was also the smallest value in the range $[1, |E|]$ with minimum path error, but we leave it as open problem studying whether this holds in general.

4 Experiments Design

In this section, we discuss the solvers implemented in this paper in Section 4.1, the experiments and datasets used to test all formulations in Section 4.2, as well the metrics used to evaluate the formulations, and finally present all results in tables in Section 4.3.

4.1 Solvers

We implemented all three previous formulations and tested them against our new formulation. We denote the bounded-error formulation (Problem 2) as Bounded; the inexact formulation (Problem 3) as Inexact; the least-squares formulation (Problem 4) as LeastSquares; the minimum path-error formulation (Problem 5) as MinPathError. We also run the ILP formulation from [8] (Problem 1), which we denote as MFD. Our implementations were done using the GUROBI Python API under default settings, and allocating only one thread to the GUROBI solver. We ran our experiments on a personal computer with 16 GB of RAM and an Apple M1 processor at 2.9 GHz (using the ARM-native `macOS 64-bit universal2` Gurobi package). The runtimes of our ILP implementations include the linear scan in increasing order on k.

4.2 Datasets and metrics

In our experiments, we start from a human transcriptomics dataset generated by [22] containing perfect splice graphs for each human gene. This was built using publicly available RNA transcripts from the Sequence Read Archive with quantification using the tool Salmon [19][2]. In this file, we used all 40870 instances available. Each input consists of a perfect flow in an acyclic splice graph for each human gene, together with its annotated RNA transcripts (i.e., the ground truth s-t paths in the graph) and simulated expression values (i.e., ground truth path weights). This dataset was used in [15], [22] to measure the accuracy of the Minimum Flow Decomposition problem. Additionally, to obtain non-trivial instances, we remove all input graphs that are a single path. The inputs for MFD tests are the perfect inputs from this original dataset. To obtain imperfect flow networks, we introduce errors or noise in the flow values of the edges, as follows. We first assume that the read coverage distribution follows a Poisson distribution, as commonly assumed, see e.g. [5], [17]. For each edge (u,v) with perfect flow value f^*_{uv}, we consider the Poisson distribution $\text{Pois}(f^*_{uv})$. For each allowed error range $\epsilon \in \{0.25, 0.5, 1\}$, we compute the $(0.5-\epsilon/2)*100$-th percentile and $(0.5+\epsilon/2)*100$-th percentile of this distribution. To get an input for the Inexact formulation, we use these two values as the two flow bounds $\underline{f}_{uv}$ and $\overline{f}_{uv}$, respectively, of edge (u,v). While this might create infeasible instances, we observed none in our case. To get an input to be used for the other formulations, for each edge (u,v), we set its imperfect flow value f_{uv} by taking a random sample from the distribution $\text{Pois}(f^*_{uv})$, but restricted to the range between these two percentiles (and normalized to the mass of the distribution in this range); note that $\epsilon = 1$ is equal to sampling from the full distribution. The data generation procedure can be found in the https://github.com/algbio/RobustFlowDecomposition, and the new dataset can be found in https://doi.org/10.5281/zenodo.10775004. For the experiments regarding the bounded-error formulation, the value B was chosen based on the parameter $\epsilon \in \{0.25, 0.5, 1\}$ of the dataset, namely $B = (1+\epsilon)\max\{f_{uv} : (u,v) \in E\}$.

As part of our evaluation procedure, we consider three metrics of increasing complexity to indicate if the set of weighted paths output for an input graph by a formulation is a *failure*:

- M1: the superposition of the weighted paths (i.e., the flow induced from the solution) do not fully match the superposition of the ground truth paths (i.e., the original perfect flow);
- M2: the output paths (as sequences of edges) are not exactly the same as the ground truth paths;
- M3: the output paths (as sequences of edges) are not the same as the ground truth paths, or some path has the different weight as its corresponding ground truth path.

For each formulation, we compute the *inaccuracy rate* as the proportion of all input graphs whose output is classified as *failure*, under each metric. This corre-

2. The full dataset from [22] is available at https://zenodo.org/record/1460998. We use the file `rnaseq/salmon/sparse_quant_SRR020730.graph`

sponds to $1 - accuracy$, where *accuracy* analogously computed under M2 and M3 were previously used in [15], [22]. We also include the first metric to evaluate how well the formulations can recover the original perfect flow from its erroneous variant (before being split into paths).

4.3 Results

In Table 1, we show the inaccuracy rate computed over the entire dataset. For baseline values, in this table, we also show the results for MFD. Note that, by definition, the inaccuracy rate of M1 for MFD is 0% since its input flow is perfect, and M1 does not measure how well MFD decomposes the flow into paths. Therefore such values are not reported in the table. Among the formulations handling errors, we observe that MinPathError is the best performing under M1 (around 40% better than Inexact), meaning that it can more accurately reconstruct the ground truth flow from its erroneous version.

Next, we discuss only metrics M2 and M3, which also take into account how well also the ground truth paths (and their weights) are reconstructed. Under these two metrics, MFD gives a correct answer in 96%–95% of the inputs, respectively. As errors in the flow values are introduced (i.e. with the increase of ϵ), the formulations handling errors show a general decline from these baseline values attained by MFD. Among the previous formulations, the best-performing one is Inexact. MinPathError has a relative improvement over Inexact ranging between 33–47%, depending on the metric and the level of errors. For example, at a moderate level of errors ($\epsilon = 0.5$), MinPathError is only 3–4 percentage points behind MFD (recall that MFD assumes error-free flows), whereas Inexact is 8–12 percentage points behind MFD. For the largest level of errors, MinPathError presents a 6 percentage points incrase in inaccuracy rate from the MFD, whereas this increase is significantly higher for Inexact. Although Bounded and LeastSquares start with competitive performances with small errors, they drastically under perform for moderate and large errors.

In Table 2, we further separate these inaccuracy rate results by grouping the inputs based on the number of ground truth paths. Once again, the inaccuracy rates for MFD under M1 are 0%, by definition. Therefore such values were not reported in the table. As expected, the values of the metrics for all formulations get worse with the increase in the number of ground truth paths. However, compared to Inexact, we observe that the results of MinPathError are more stable with the increase of this number. For example, on large-error levels (when we sample from the full Poisson distribution), the inaccuracy rate of MinPathError remains below 0.18 in all cases, while that of Inexact reaches around 0.30. This increase is further accentuated for the remaining formulations, i.e., reaching up to 0.40 for both Bounded and LeastSquares.

In Table 3, we show the runtimes of all four formulations. As expected, MFD is the fastest overall, followed closely by Inexact. Thus, Inexact is the best among the previous formulations handling errors, being the fastest and most accurate. MinPathError is about 4 times slower than Inexact, having more variables and including additional auxiliaries due to non-linear components, and over all graphs MinPathError finishes within 18 hours. However, we can expect that practical tools implementing our model run each input graph in parallel, since this is trivially possible. In this case, the wall-clock time can be roughly estimated as the running times we reported divided by the number of CPU threads available. We also observe that the runtime of MinPathError grows exponentially with the number of ground truth paths, but nevertheless, across all levels of errors, each instance takes at most 400 seconds, which is still practically feasible. However, the payoff for this larger computational requirement is the significantly smaller inaccuracy rate, as discussed above.

In our experiments, we also we show the precision and recall for three abundance levels across all instances used as input (see Table 1 in Supplementary Material). For this calculation, we first define the true positive (if a path that is reported as a solution is also present as part of ground truth), false negative (if a path that is reported as a solution is not present as part of the ground truth) and false negative (if a path is presented in the ground truth but is not reported as a part of the solution). Those values are calculated in light of $M2$ and $M3$, considering that $M1$ only consider the correct superposition. With such values in mind, precision is calculated as the ratio of true positive overall positives (true and false), and recall is the ratio of true positive overall reported (true positive and false negative). Based on the results, it is evident to conclude that, in both Inexact and MinPathError, the precision is lower than recall, signifying that the number of false positives outweighs the number of false negatives. In addition, with lower abundance, the values of precision and recall are higher, highlighting that with more abundance, there is less robustness in the predicted decomposition. Finally, with more paths in the decomposition, both values of precision and recall tend to decrease. The main difference between Inexact and MinPathError is that, in the latter, the decrease in precision and recall is slower than in the former. This concludes that both methods have

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3433523

TABLE 1: Proportion of input graphs that are classified as *failure* under each metric (inaccuracy rate). For MinPathError, we also list in parentheses the relative improvement of the values of each metric over the corresponding values of best previous formulation handling errors, namely Inexact.

Model	M1	M2	M3
MFD	-	0.04	0.05

	ϵ = 0.25			ϵ = 0.5			ϵ = 1.00		
Model	M1	M2	M3	M1	M2	M3	M1	M2	M3
Bounded	0.11	0.15	0.20	0.15	0.17	0.23	0.18	0.21	0.24
Inexact	0.09	0.11	0.15	0.10	0.12	0.17	0.14	0.15	0.19
LeastSquares	0.20	0.22	0.26	0.20	0.24	0.25	0.23	0.28	0.32
MinPathError	0.05 (-44.45%)	0.07 (-36.40%)	0.08 (-46.70%)	0.06 (-40.00%)	0.07 (-41.70%)	0.09 (-47.05%)	0.08 (-42.85%)	0.10 (-33.34%)	0.11 (-42.10%)

very similar behaviour, with Inexact outperforming in consistency and robustness.

It is also important to recall that both Inexact and Bounded are solved iteratively with increasing k until a feasible solution is found. At the same time, LeastSquares requires *all* possible values of k to be tested, and then an optimal solution is found. Moreover, MinPathError stops when two consecutive iterations do not show improvement in the total path error. The reported runtime considers all iterations executed for each method, which partly explains the higher runtime of the two latter formulations.

5 Conclusion

By interpreting the imperfect flow decomposition problem as a robust optimization problem, we shifted the management of errors from the microscopic focus on the edges of the graph to the macroscopic focus on the weights of the solution paths. This led to a significant increase in accuracy compared to the previous error-handling formulations, even on a large amount of errors. This approach is less accurate when compared only to error-free problem instances. This decrease in inaccuracy rate comes at the cost of an increased running time; however, despite the NP-hardness of the problem, this is still feasible in practice, with any instance finishing in at most 7.5 minutes, while the average runtime over the entire dataset is much smaller.

In this paper, we focused only on giving a flow decomposition formulation that is better at handling errors in coverage values. In particular, we considered the results of MFD on perfect flows to be a gold standard. However, in practical Bioinformatics tools, additional constraints can be added to restrict the solution space further. For example, it is known that if one is also given a set of paths that must appear as subpath in at least one solution path (*subpath constraints*, arising from long reads aligned to the graph), then the accuracy further increases [31]. Such subpath constraints have been added on top of the ILP formulation for MFD, see [8]. They can also be trivially added to our ILP from Section 3 by analogously requiring that they appear as subpaths also in our solution paths. We leave this problem for future work since, as already mentioned, this paper focuses on better handling erroneous flows. Overall, given the performance and flexibility of all such ILP formulations, they can be at the core of future RNA assemblers.

Although our formulation (for a given number k of paths) requires a larger number of variables and constraints to express uncertainty, its size is still asymptotically quadratic, similar to the previous ILP formulations for MFD. Considering the further development of ILP solvers, this is likely to be only a minor limitation in downstream applications. Currently, the main drawback is the dependency of k. An improvement of the overall formulation is to obtain a direct estimation of the number k of paths of each decomposition which would allow for avoiding multiple iterations. Finally, we look forward to extending different ILP formulations for this problem involving the stochastic formulation [16], uncertainty in cyclic graphs [9], and safety [14].

Acknowledgment

This work was partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 851093, SAFEBIO), partially by the Research Council of Finland (grants No. 322595, 352821, 346968, 358744).

REFERENCES

[1] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows*. Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts, 1988.

[2] Jasmijn A Baaijens, Leen Stougie, and Alexander Schönhuth. Strain-aware assembly of genomes from mixed samples using flow variation graphs. In *International Conference on Research in Computational Molecular Biology*, pages 221–222. Springer, 2020.

[3] Jasmijn A Baaijens, Bastiaan Van der Roest, Johannes Köster, Leen Stougie, and Alexander Schönhuth. Full-length de novo viral quasispecies assembly through variation graph construction. *Bioinformatics*, 35(24):5086–5094, 2019.

[4] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming*, 88:411–424, 2000.

[5] Elsa Bernard, Laurent Jacob, Julien Mairal, and Jean-Philippe Vert. Efficient rna isoform identification and quantification from rna-seq data with network flows. *Bioinformatics*, 30(17):2447–2455, 2014.

[6] Manuel Cáceres, Massimo Cairo, Andreas Grigorjew, Shahbaz Khan, Brendan Mumey, Romeo Rizzi, Alexandru I. Tomescu, and Lucia Williams. Width helps and hinders splitting flows. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[7] Jiao Chen, Yingchao Zhao, and Yanni Sun. De novo haplotype reconstruction in viral quasispecies using paired-end read guided path finding. *Bioinformatics*, 34(17):2927–2935, 2018.

[8] Fernando H. C. Dias, Lucia Williams, Brendan Mumey, and Alexandru I. Tomescu. Fast, Flexible, and Exact Minimum Flow Decompositions via ILP. In *RECOMB 2022 - 26th Annual International Conference on Research in Computational Molecular Biology*, volume 13278 of *Lecture Notes in Computer Science*, pages 230–245. Springer, 2022.

[9] Fernando HC Dias, Lucia Williams, Brendan Mumey, and Alexandru I Tomescu. Minimum flow decomposition in graphs with cycles using integer linear programming. *arXiv preprint arXiv:2209.00042*, 2022.

[10] Thomas Gatter and Peter F Stadler. Ryūtō: network-flow based transcriptome reconstruction. *BMC bioinformatics*, 20(1):1–14, 2019.

[11] Tzvika Hartman, Avinatan Hassidim, Haim Kaplan, Danny Raz, and Michal Segalov. How to split a flow? In *2012 Proceedings IEEE INFOCOM*, pages 828–836. IEEE, 2012.

[12] Steffen Heber, Max Alekseyev, Sing-Hoi Sze, Haixu Tang, and Pavel A Pevzner. Splicing graphs and est assembly problem. *Bioinformatics*, 18(suppl_1):S181–S188, 2002.

[13] Rodney Hull, Mzwandile Mbele, Tshepiso Makhafola, Chindo Hicks, Shao-Ming Wang, Rui Manuel Reis, Ravi Mehrotra, Zilungile Mkhize-Kwitshana, Gibson Kibiki, David O Bates, et al. Cervical cancer in low and middle-income countries. *Oncology letters*, 20(3):2058–2074, 2020.

[14] Shahbaz Khan, Milla Kortelainen, Manuel Cáceres, Lucia Williams, and Alexandru I Tomescu. Safety and completeness in flow decompositions for RNA assembly. In *International Conference on Research in Computational Molecular Biology*, pages 177–192. Springer, 2022.

[15] Kyle Kloster, Philipp Kuinke, Michael P O'Brien, Felix Reidl, Fernando Sánchez Villaamil, Blair D Sullivan, and Andrew van der Poel. A practical fpt algorithm for flow decomposition and transcript assembly. In *2018 Proceedings of the Twentieth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 75–86. SIAM, 2018.

[16] Can Li and Ignacio E Grossmann. A review of stochastic programming methods for optimization of process systems under uncertainty. *Frontiers in Chemical Engineering*, 2:34, 2021.

[17] Wei Li, Jianxing Feng, and Tao Jiang. IsoLasso: a LASSO regression approach to RNA-Seq based transcriptome assembly. *Journal of Computational Biology*, 18(11):1693–1707, 2011.

[18] Brendan Mumey, Samareh Shahmohammadi, Kathryn McManus, and Sean Yaw. Parity balancing path flow decomposition and routing. In *2015 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2015.

[19] Rob Patro, Geet Duggal, and Carl Kingsford. Salmon: accurate, versatile and ultrafast quantification from RNA-seq data using lightweight-alignment. *BioRxiv*, page 021592, 2015.

[20] Mihaela Pertea, Geo M Pertea, Corina M Antonescu, Tsung-Cheng Chang, Joshua T Mendell, and Steven L Salzberg. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nature biotechnology*, 33(3):290–295, 2015.

[21] Andrey D Prjibelski, Alla Mikheenko, Anoushka Joglekar, Alexander Smetanin, Julien Jarroux, Alla L Lapidus, and Hagen U Tilgner. Accurate isoform discovery with IsoQuant using long reads. *Nature Biotechnology*, 41:915–918, 2023.

[22] Mingfu Shao and Carl Kingsford. Accurate assembly of transcripts through phase-preserving graph decomposition. *Nature biotechnology*, 35(12):1167–1169, 2017.

[23] Mingfu Shao and Carl Kingsford. Theory and a heuristic for the minimum path flow decomposition problem. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(2):658–670, 2017.

[24] Allen L Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations research*, 21(5):1154–1157, 1973.

[25] Leonardo Taccari. Integer programming formulations for the elementary shortest path problem. *European Journal of Operational Research*, 252(1):122–130, 2016.

[26] Alexandru I Tomescu, Travis Gagie, Alexandru Popa, Romeo Rizzi, Anna Kuosmanen, and Veli Mäkinen. Explaining a weighted dag with few paths for solving genome-guided multi-assembly. *IEEE/ACM transactions on computational biology and bioinformatics*, 12(6):1345–1354, 2015.

[27] Alexandru I Tomescu, Anna Kuosmanen, Romeo Rizzi, and Veli Mäkinen. A novel min-cost flow method for estimating transcript expression with RNA-Seq. In *BMC bioinformatics*, volume 14, pages S15:1–S15:10. Springer, 2013.

[28] B. Vatinlen, F. Chauvet, P. Chrétienne, and P. Mahey. Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths. *European Journal of Operational Research*, 185(3):1390–1401, 2008.

[29] Kelly Westbrooks, Irina Astrovskaya, David Campo, Yury Khudyakov, Piotr Berman, and Alex Zelikovsky. HCV quasispecies assembly using network flows. In *International Symposium on Bioinformatics Research and Applications*, pages 159–170. Springer, 2008.

[30] Lucia Williams, Gillian Reynolds, and Brendan Mumey. RNA Transcript Assembly Using Inexact Flows. In *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1907–1914. IEEE, 2019.

[31] Lucia Williams, Alexandru Tomescu, Brendan Marshall Mumey, et al. Flow decomposition with subpath constraints. In *21st International Workshop on Algorithms in Bioinformatics (WABI 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[32] Charlotte J Wright, Christopher WJ Smith, and Chris D

Jiggins. Alternative splicing as a source of phenotypic diversity. *Nature Reviews Genetics*, 23(11):697–710, 2022.

[33] Yi Xing, Alissa Resch, and Christopher Lee. The multi-assembly problem: reconstructing multiple transcript isoforms from est fragment mixtures. *Genome research*, 14(3):426–441, 2004.

[34] Qimin Zhang, Qian Shi, and Mingfu Shao. Scallop2 enables accurate assembly of multiple-end rna-seq data. *bioRxiv*, 2021.

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3433523

TABLE 2: Proportion of input graphs that are classified as *failure* under each metric (inaccuracy rate). Input graphs are partitioned by the number of ground truth paths (column "G.t. paths"). For MinPathError, we also list the relative improvement of the values of the metric over the corresponding values of best previous formulation, namely Inexact.

		ϵ = 0		
	G.t. paths	M1	M2	M3
MFD	2-5	-	0.03	0.05
	6-10	-	0.04	0.07
	11-15	-	0.05	0.08
	16-20	-	0.07	0.08
	21-26	-	0.08	0.09

		ϵ = 0.25			ϵ = 0.5			ϵ = 1.00		
	G.t. paths	M1	M2	M3	M1	M2	M3	M1	M2	M3
Bounded	2-5	0.12	0.14	0.18	0.16	0.17	0.19	0.23	0.26	0.28
	6-10	0.14	0.17	0.20	0.17	0.18	0.20	0.25	0.27	0.28
	11-15	0.20	0.21	0.28	0.19	0.22	0.27	0.32	0.33	0.35
	16-20	0.21	0.23	0.30	0.24	0.26	0.29	0.34	0.35	0.38
	21-26	0.26	0.27	0.32	0.27	0.28	0.32	0.35	0.37	0.41
Inexact	2-5	0.10	0.12	0.13	0.11	0.12	0.14	0.15	0.16	0.18
	6-10	0.11	0.13	0.15	0.13	0.14	0.15	0.17	0.18	0.20
	11-15	0.13	0.14	0.16	0.15	0.15	0.16	0.20	0.21	0.24
	16-20	0.14	0.15	0.17	0.17	0.16	0.17	0.22	0.23	0.27
	21-26	0.16	0.17	0.19	0.18	0.19	0.24	0.26	0.28	0.30
LeastSquares	2-5	0.20	0.22	0.24	0.25	0.26	0.29	0.27	0.28	0.30
	6-10	0.22	0.23	0.25	0.26	0.27	0.31	0.28	0.29	0.33
	11-15	0.23	0.24	0.27	0.27	0.28	0.32	0.30	0.31	0.31
	16-20	0.24	0.25	0.28	0.28	0.29	0.33	0.32	0.35	0.37
	21-26	0.25	0.27	0.30	0.29	0.30	0.34	0.36	0.38	0.40
MinPathError	2-5	0.06 (-40.00%)	0.07 (-41.70%)	0.08 (-38.46%)	0.08 (-27.30%)	0.09 (-25.00%)	0.10 (-28.57%)	0.09 (-40.00%)	0.10 (-37.50%)	0.11 (–38.90%)
	6-10	0.07 (-36.40%)	0.08 (-38.46%)	0.09 (-40.00%)	0.09 (-30.76%)	0.10 (-28.57%)	0.11 (-26.70%)	0.10 (-41.17%)	0.12 (-33.40%)	0.13 (-35.00%)
	11-15	0.08 (-38.46%)	0.10 (-28.57%)	0.11 (-31.25%)	0.11 (-26.67%)	0.12 (-20.00%)	0.13 (-18.75%)	0.12 (-40.00%)	0.14 (-33.34%)	0.15 (-37.50%)
	16-20	0.10 (-28.57%)	0.11 (-26.70%)	0.12 (-29.41%)	0.13 (-23.52%)	0.14 (-12.50%)	0.15 (-11.76%)	0.16 (-27.30%)	0.16 (-30.43%)	0.17 (-37.04%)
	21-26	0.12 (-25.00%)	0.13 (-23.52%)	0.14 (-26.31%)	0.13 (-27.78%)	0.14 (-26.32%)	0.15 (-37.50%)	0.16 (-38.46%)	0.16 (-42.85%)	0.18 (-40.00%)

This article has been accepted for publication in IEEE/ACM Transactions on Computational Biology and Bioinformatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TCBB.2024.3433523

TABLE 3: Running times for all formulations tested and implemented in this paper. The "Average Time" columns contain in parentheses the standard deviations of the running times.

$\epsilon = 0$			
Model	#-instances	Time (h)	Average Time (s)
MFD	40870	3.94	0.51 (0.43)

			$\epsilon = 0.25$		$\epsilon = 0.5$		$\epsilon = 1.00$	
Model			Time (h)	Average Time (s)	Time (h)	Average Time (s)	Time (h)	Average Time (s)
Bounded			5.23	0.46 (0.42)	6.68	0.58 (0.61)	7.24	0.63 (0.42)
Inexact			3.67	0.32 (0.61)	4.32	0.38 (0.79)	5.14	0.45 (0.36)
LeastSquares			7.56	0.66 (0.73)	8.28	0.72 (0.68)	8.44	0.74 (0.39)
	G.t. paths	#-instances						
MinPathError	2-5	36471	8.06	0.81 (0.50)	9.12	0.90 (0.61)	9.78	0.96 (0.64)
	6-10	4041	2.53	2.25 (1.28)	3.13	2.78 (1.52)	3.55	3.16 (2.58)
	11-15	295	1.16	14.2 (4.13)	1.72	20.9 (5.41)	2.14	26.1 (6.13)
	16-20	46	0.92	72.1 (35.5)	1.25	97.8 (47.6)	1.64	128 (50.1)
	21-26	17	0.78	165 (50.9)	0.94	199 (83.1)	1.08	228 (124)
	All	40870	13.45	1.28	16.16	1.57	18.19	2.31