RNA Transcript Assembly Using Inexact Flows

1st Lucia Williams
Gianforte School of Computing
Montana State University
Bozeman, USA
luciawilliams@montana.edu

2nd Gillian Reynolds
Gianforte School of Computing
Montana State University
Bozeman, USA
gillian.reynolds@student.montana.edu

3rd Brendan Mumey
Gianforte School of Computing
Montana State University
Bozeman, USA
brendan.mumey@montana.edu

***Abstract*—RNA-Seq technology allows for high-throughput, low cost measurement of gene expression. An important step in this process is the assembly of mRNA transcript short reads into full transcripts. The problem can be viewed as a flow decomposition problem in which the objective is to minimize the number of path flows needed to represent a given flow. In this work we relax the edge flow constraints to allow for some uncertainty in their measurement. We formulate this as the Inexact Flow Decomposition problem and propose an algorithmic strategy to solve it. In practice, real biological data has measurement errors and so experimentally-derived edge-weighted splice graphs are often not flows. The proposed method is the first approach to this problem that explicitly controls the error allowed on each edge in these graphs in order to achieve a flow. In an intermediate step, the method solves an exact flow decomposition instance; if a greedy method is used for this step, the overall running time is $O(|E|^2|V|^2 + |P|^3)$, where P is the solution found to the flow decomposition instance. Preliminary results on simulated biological data sets show that in many cases the ground truth paths can be recovered at approximately correct abundances, even with noisy input data.**

***Index Terms*—RNA sequencing, flow networks, transcript assembly, RNA splicing**

I. Introduction

RNA-Seq is a powerful sequencing technology that allows for both the discovery and quantification of expressed RNA transcripts [4]. One of its primary applications is as a low cost measurement of gene expression [31] under a variable set of conditions such as developmental stages, time, disease states, cell type and stimuli [11]. Alongside the identification of regulatory activity, RNA-Seq may also be used to identify small and long non-coding regulatory RNAs, infer gene structures and identify fusion and alternatively spliced transcripts [6], [15], [18], [20].

Each of these applications provides unparalleled insight into the complexity of the underlying gene regulation responsible for phenotypic diversity, and each comes with their own suite of computational challenges. One of the most significant challenges of RNA-seq analysis is the accurate reconstruction of transcript sequences from short-read sequencing technologies, which have dominated the sequencing market since the introduction of second generation sequencing technology.

Supported provided by US National Science Foundation grant DBI-1759522 and DBI-1661530.

978-1-7281-1867-3/19/$31.00 ©2019 IEEE

As such, much effort has been placed into the development of methods capable of transcript assembly from short reads. Two of the most popular tools are Cufflinks [29] and StringTie [24]. Cufflinks, the older of the two methods, represents the relationship between aligned reads using an overlap graph. This graph is decomposed into a set of transcripts using minimum path cover. Transcript abundances are then estimated using a maximum likelihood approach. StringTie, on the other hand, uses read data to construct a splice graph, and chooses both transcripts and their abundances using a heaviest-path approach.

In this work, we describe a new way of extracting transcripts and their abundances from a splice graph built from short read data. Assuming the splice graph is a flow, recent work [13], [27] has modeled this task as the Flow Decomposition problem, and sought efficient heuristics [27] and improved theoretical approaches such as fixed parameter tractable (FPT) algorithms [13] to solve it. In the splice graph model, nodes are subsequences of a gene (exons), and weights (flows) on edges indicate a measurement of the number of times that the source and target exons are observed consecutively in the sample. By recovering the minimum set of weighted paths to decompose the flow on the graph, these algorithms find a likely set of expressed transcripts, along with their weights. It is assumed that the smallest number of transcripts that could have generated the data is the best solution; this assumption was validated in [13]. Here, we extend this model to account for uncertainty in the sample measurements. There are numerous sources of uncertainty, biases and errors that plague each step of an RNA-Seq experiment. Prior to splice graph construction, these sources include sampling, library preparation, base-calling, read preprocessing and spliced read alignment [5], [8], [16], [17], [23], [32]. Because of these errors, a measured splice graph is unlikely to be a flow; thus, in order to uncover transcripts and their quantities from data, an error-correction model must be applied to generate a flow network, which can then be decomposed into paths (isoforms) and weights (abundances).

Rather than assign an exact weight to each edge, we assign an interval of possible weights, thereby accounting for possible errors in measurement. We then seek a minimal set of paths to explain these intervals. We model this as the *Inexact Flow Decomposition* (IFD) problem and formalize it in §III. Additionally, we give a method for applying IFD to find source

and sink nodes in the splicing graph. In general, IFD may be a more realistic model for many problems currently modeled as flow decomposition or similar problems, because it explicitly accounts for uncertainty in measurements.

II. Related Work

The Flow Decomposition problem has been studied since 2008, when it was introduced to model routing in telecommunication networks [30]. While many algorithms for producing a flow on a network generate a decomposition of the flow into paths in order to find the solution (e.g., the augmenting paths of the Ford-Fulkerson algorithm), the set of these paths is not generally minimal. It has been shown that Flow Decomposition is NP-hard [30], even when the flow values are chosen from a small set of possible values [12]. The only known approximation algorithm for Flow Decomposition is given in [22] and gives solutions within $L^{\log(f_{\max})}\log(f_{\max})$ of optimal, where L is the longest s-t path in the graph, and $f_{\max}$ is the maximum flow on any edge. A number of simple heuristics have been proposed [30], and while they perform poorly on carefully constructed inputs [12], the method of choosing and removing the s-t path with greatest flow ("Greedy-Width") was arguably the best-performing practical algorithm until 2016, when the "Catfish" algorithm [27] was published. The main idea of this method is to apply path-preserving transformations to the input graph in order to reduce an upper bound on the number of solutions needed; then Greedy-Width is run on the transformed graph. Catfish performs very well on simulated RNA transcript assembly data, where the true number of paths to be recovered is almost always between 1 and 10. More recently, a fixed-parameter tractable (FPT) algorithm was developed [13], that gives similar accuracy and running times on the same data sets used in [27], although it becomes impractical for larger inputs.

A number of RNA-Seq assembly tools use flow decomposition as a core component of their methods. The first RNA-seq tool to utilize minimal path flow decomposition to solve the transcript assembly and quantification problem was Traph [28]; this work also considers error-correction in the experimentally-measured splice graph, by first determining a flow that best fits the data. Other tools such as Scallop [26] do not explicitly solve a flow decomposition problem, but do seek parsimonious sets of paths through splice graphs. A very recent tool called Ryuto [9] builds upon the methods developed by [28] and [26] and was able to reduce the number of false positive transcripts identified while finding more true transcripts when compared to Scallop.

III. Preliminaries

An *inexact flow network* is comprised of a network $G = (V, E)$, where V is a set of vertices and E is a multiset of directed edges; V contains a designated source vertex s and a designated sink vertex t. $|V| = n$ and $|E| = m$. Additionally, each edge $e \in E$ has an associated flow interval I_e. Flow intervals are either *bounded* or *unbounded*. Bounded flow intervals have the form $[l_e, u_e]$, where $0 \leq l_e \leq u_e$. Unbounded flow intervals have the form $[l_e, \infty)$, where $0 \leq l_e$. We use unbounded edges to identify a source and sink if they are not known; see §V-C. A path flow decomposition for an inexact flow network consists of set of s-t-paths $P = \{p_1, \ldots, p_k\}$ and associated positive path flow weights, $w = (w_1, \ldots, w_k)$ such that for each edge e

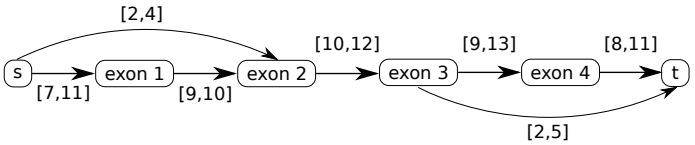
$$[f(e) \triangleq \sum_{i:e \in p_i} w_i] \in I_e. \tag{1}$$

We specify each path p_i by its ordered list of edges, where $p_i[k]$ refers to the k^{th} edge along p_i.

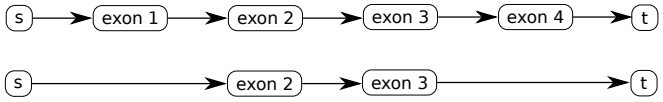
Any path flow decomposition of a splice graph G represents a set of transcripts and their abundances that could have generated the reads used to construct G.

In this work, we restrict attention to the case where the flow interval bounds $0 \leq l_e \leq u_e$ and the path weights $w_i > 0$ are integers. The *inexact flow decomposition* (IFD) problem is to find a path flow decomposition (P, w) such that $|P| = k$ is minimized. Among all solutions (P, w), where $|P| = k$, we seek those with minimum total flow $F = \sum_i w_i$. Figure 1 shows an example inexact flow network representing a gene with four exons, along with an optimal decomposition of the network into two paths and a non-optimal decomposition of the network into three paths.

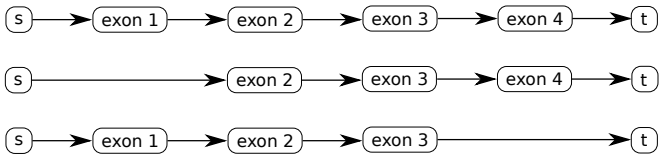
Lemma 1: Let (P^*, w^*) be an optimal solution to IFD. For each $p_i \in P^*$, there exists an edge $e \in p_i$ such that $f(e) = l_e$.



(a) An inexact flow network representing a gene with four exons.



(b) An example decomposition of the inexact flow network into two paths, representing two transcripts. Assigning the first transcript weight 10 and the second transcript weight 2 satisfies Equation 1.



(c) An example decomposition of the inexact flow network into three paths, representing three transcripts. Assigning the first transcript weight 9 and the second and third transcripts weight 1 satisfies Equation 1.

Fig. 1: An example inexact flow network and two possible decompositions of it.

Proof: Suppose there exists $p_i \in P$ such that for every edge $e \in p_i$, $f(e) > l_e$. Then the flow value w_i of p_i can be reduced by 1, contradicting the optimality of (P^*, w^*). ∎

The upper bound on the fixed-weight version of the problem, given in [30], applies to the interval version as well.

Lemma 2 (Upper bound on optimal solutions): $k^* \leq m - n + 2$.

Proof: First, note that (P^*, w^*) corresponds to some flow on the graph G. By Proposition 4 of [30], the paths in a minimal decomposition of an exact flow graph must be linearly independent. Then, the number of paths in an optimal decomposition of the flow graph is upper bounded by the number of linearly independent paths in the graph. As in the proof of Proposition 9 of [30], we let G^0 be G with an additional edge from t to s. Then, by Theorem 1, Chapter 2 of [3], the maximum number of independent cycles in G^0 is $(m + 1) - n + 1$. So, the total number of independent cycles in G is $m - n + 2$. Thus, $k^* = |P^*| \leq m - n + 2$. ∎

We note that this upper bound may be improved by excluding edges that will not be used in a solution to IFD.

Lemma 3: The maximum flow on any edge in an optimal path decomposition is bounded above by

$$B = (m - n + 2) \max_e l_e. \tag{2}$$

Proof: By Lemma 1, the flow on a path in any optimal path decomposition (P^*, w^*) equals some l_e value. By Lemma 2, there are at most $m - n + 2$ paths in P^*. Thus, $f(e) \leq (m - n + 2) \max_e l_e$ for any edge $e \in E$. ∎

Lemma 3 shows that we can replace unbounded flow intervals with bounded ones, so we can assume that all flow intervals are bounded in the input. For example, in the network shown in Fig. 1, $B = 30$, and would be even if any edges had infinite upper bounds.

Clearly a path flow decomposition to an IFD instance exists if and only there is a flow solution $\langle f(e) \rangle_{e \in E}$ that meets all flow interval constraints. Given an IFD instance in which the flow intervals are all bounded, we can use a standard approach based on maximum-flow that can find a flow solution that conforms to all lower and upper bounds on edge flows, if one exists. For completeness, we describe the approach: We define a new set of flow variables $\langle f'(e) \rangle$ satisfying

$$f(e) = l_e + f'(e). \tag{3}$$

We add a new edge (t, s) with $l_{(t,s)} = 0$ and $u_{(t,s)} = B$ and extend the conservation of flow requirement to s and t; thus (t, s) will have flow F in any flow solution. For each vertex $v \in V$ (including s and t),

$$\sum_{e \in \text{out}(v)} f(e) - \sum_{e \in \text{in}(v)} f(e) = 0. \tag{4}$$

By (3) and (4),

$$\sum_{e \in \text{out}(v)} f'(e) - \sum_{e \in \text{in}(v)} f'(e) = \sum_{e \in \text{in}(v)} l_e - \sum_{e \in \text{out}(v)} l_e \triangleq \text{exc}(v). \tag{5}$$

We create a flow network $G' = (V', E')$ where $V' = V \cup \{s', t'\}$; s' is a new source vertex and t' is a new sink vertex. We include all edges $e \in E$ into E' where the capacity of an edge $e \in E$ is set to $u_e - l_e$. For any $v \in V$ with $\text{exc}(v) > 0$, we add an edge (s', v) with capacity $\text{exc}(v)$. Likewise, for any $v \in V$ with $\text{exc}(v) < 0$, we add an edge (v, t') with capacity $-\text{exc}(v)$. Next, we find a maximum flow F^* in this network (e.g., by using the Edmonds-Karp algorithm [7]; we note the solution found will be integral since all constraints are integers).

Lemma 4: The maxflow solution $\langle f'(e') \rangle$ saturates all edges $(s', v) \in E'$ (equivalently, all edges $(v, t') \in E'$) if and only if there is a flow solution $\langle f(e) \rangle$ to the IFD instance.

Proof: It is easy to see that $\sum_{v \in V} \text{exc}(v) = 0$, since each edge $e \in E$ contributes l_e and $-l_e$ to the sum. Thus,

$$\sum_{v \in V : \text{exc}(v) > 0} \text{exc}(v) = \sum_{v \in V : \text{exc}(v) < 0} -\text{exc}(v), \tag{6}$$

and so the outgoing edges from s' are saturated if and only if the incoming edges to t' are saturated. If this occurs, by conservation of flow, (5) will be satisfied for all $v \in V$, and so defining $\langle f(e) \rangle$ by (3) provides valid flow in the original IFD network (since $0 \leq f'(e) \leq u_e - l_e$ for all $e \in E$ and the fact that (3) and (5) imply (4)). Conversely, a valid IFD flow $\langle f(e) \rangle$ gives rise to a maximum flow in G', $\langle f'(e) \rangle$ by defining $f'(e) = f(e) - l_e$ for $e \in E$ and saturating all edges incident to s' and t'. ∎

IV. Proposed IFD Algorithm

Lemmas 3 and 4 suggest the following strategy for solving the IFD problem: first compute upper bounds for all unbounded edge constraints, then find a valid integer flow solution, then decompose this flow into paths. We refine this basic approach with several additional heuristics and summarize it in Fig. 2.

1) Replace all unbounded flow intervals $[a, \infty)$ with bounded intervals $[a, B]$, where B is computed in Equation (2).
2) Find a valid integer flow $\langle f(e) \rangle$ for the bounded IFD instance (or report that no solution exists) by solving an associated maxflow problem.
3) *Heuristic 1:* Modify the flow solution $\langle f(e) \rangle$ to another feasible solution $\langle f'(e) \rangle$ that potentially requires fewer paths to decompose.
4) Decompose $\langle f'(e) \rangle$ using an existing method such as Greedy-Width or Catfish to obtain an initial path solution (P, w).
5) *Heuristic 2:* Refine (P, w) in order to reduce $|P|$; we search for opportunities to adjust the weights on a pair of paths such that a third path becomes unnecessary, or to splice paths such that one of the resulting paths can be eliminated.

Fig. 2: The proposed IFD algorithm.

We remark that this strategy is guaranteed to find a solution when one exists but that this solution is not necessarily optimal. Since IFD generalizes the standard flow-decomposition problem, it is at least as computationally hard; namely it is NP-hard even if all flow values are taken from $\{1, 2, 4\}$ [12] and also does not admit a PTAS [30]. The running time of each step is: Step 1: $O(m)$ time, Step 2: $O(n^2m)$ time (using Edmonds-Karp), Step 3: $O(n^2m)$ time per edge removed (see below), Step 4: Greedy-Width: $O(m + n \lg n)$ time (using a modification of Dijkstra's algorithm), Catfish: $O(m^2n^2|f'|)$ time, Step 5: $O(|P|^3)$ time. We next describe the heuristic steps above in further detail.

A. Heuristic 1: Flow simplification

It may be possible to modify the initial flow $\langle f(e) \rangle$ produced by Step 2. to create another feasible flow $\langle f'(e) \rangle$ that requires fewer paths to decompose. Lemma 3 provides an upper bound on k^* that depends linearly on the number of edges in the graph. Thus, if we can modify the flow to use fewer edges, we may reduce k^*. We define $C_f = \{e : l_e = 0, f(e) > 0\}$ as the set of candidate edges for elimination. To check if it is possible to modify the flow solution f so that $f(e) = 0$ for some $e \in C_f$, we make use of the previous maxflow instance and modify the upper bound constraint for e to be $u_e = 0$; again by the Lemma 4, we can find a feasible flow with this additional constraint if one exists. The full method is as follows:

1) Compute $C_f = \{e : l_e = 0, f(e) > 0\}$ and $Z_f = \{e' : f(e') = 0\}$.
2) Order C_f by increasing $f(e)$.
3) For each $e \in C_f$, check (using maxflow) if f' can be found that satisfies the IFD instance with additional constraints $f'(e) = 0$ and $f'(e') = 0$ for all $e' \in Z_f$. If yes, add e to Z_f and go to 1.

B. Heuristic 2: Reducing the number of paths

We next describe several heuristics for reducing the number of paths needed in the IFD solution. If the edge flow interval bounds are somewhat slack, these opportunities are often found (see §VI).

a) Rebalancing: Consider a triple of paths p_i, p_j and p_k that overlap as shown in Fig. 3. It is possible that the flows on p_i and p_j can be adjusted such that p_k can be eliminated. For each pair of paths p_i, p_j, we compute the following:

$$\begin{aligned} \text{fwd}(i,j) &\triangleq \operatorname*{argmax}_{l \geq 0} p_i[1 \ldots l] = p_j[1 \ldots l] \\ \text{rev}(i,j) &\triangleq \operatorname*{argmax}_{l \geq 0} p_i[\text{len}(p_i) - l + 1 \ldots \text{len}(p_i)] \\ &= p_j[\text{len}(p_j) - l + 1 \ldots \text{len}(p_j)] \end{aligned} \tag{7}$$

Next, for all triple of paths p_i, p_j, p_k, we define:

$$\text{lap}(i,j,k) \triangleq \text{fwd}(i,k) + \text{rev}(j,k) - \text{len}(p_k). \tag{8}$$

If $lap(i,j,k) > 0$, then paths p_i, p_j and p_k are as shown in Fig. 3. Let

$$f_{-ijk}(e) \triangleq \sum_{p_l \in P \setminus \{p_i, p_j, p_k\}, e \in p_l} w_l \tag{9}$$

be the current flow on edge e, ignoring paths p_i, p_j and p_k.

We first check if the flows w_i and w_j on paths p_i and p_j can be adjusted so that all affected edge constraints remain satisfied. Then w_i and w_j must satisfy the following constraints:

$$\begin{aligned} w_i &\in [l_e - f_{-ijk}(e), u_e - f_{-ijk}(e)], \quad \forall e \in p_i \cap \overline{p_j}, \\ w_j &\in [l_e - f_{-ijk}(e), u_e - f_{-ijk}(e)], \quad \forall e \in \overline{p_i} \cap p_j, \\ w_i + w_j &\in [l_e - f_{-ijk}(e), u_e - f_{-ijk}(e)], \quad \forall e \in p_i \cap p_j. \end{aligned}$$

Let

$$\begin{aligned} L_{w_i} &= \max_{e \in p_i \cap \overline{p_j}} \; l_e - f_{-ijk}(e), \\ U_{w_i} &= \min_{e \in p_i \cap \overline{p_j}} \; u_e - f_{-ijk}(e), \\ L_{w_j} &= \max_{e \in \overline{p_i} \cap p_j} \; l_e - f_{-ijk}(e), \\ U_{w_j} &= \min_{e \in \overline{p_i} \cap p_j} \; u_e - f_{-ijk}(e), \\ L_{w_i+w_j} &= \max_{e \in p_i \cap p_j} \; l_e - f_{-ijk}(e), \\ U_{w_i+w_j} &= \min_{e \in p_i \cap p_j} \; u_e - f_{-ijk}(e). \end{aligned}$$

Then, the above constraints are equivalent to the following:

$$\begin{aligned} w_i &\in [L_{w_i}, U_{w_i}], \\ w_j &\in [L_{w_j}, U_{w_j}], \\ w_i + w_j &\in [L_{w_i+w_j}, U_{w_i+w_j}]. \end{aligned} \tag{10}$$

There exists an integer solution w_i, w_j that satisfies each of the constraints in (10) if and only if $L_{w_i} \leq U_{w_i}$, $L_{w_j} \leq U_{w_j}$, $L_{w_i+w_j} \leq U_{w_i} + U_{w_j}$, and $L_{w_i} + L_{w_j} \leq U_{w_i+w_j}$. If all these inequalities hold, then among feasible solutions w', w_k we choose one that maximizes the distance to any constraint with the idea that this may improve the chance of finding subsequent path reductions.

b) Splice and Merge: If it is not possible to eliminate p_k by modifying the flow on p_i and p_j, then we next consider the option of splicing p_i and p_j in the region they overlap. The resulting path splice generates a path that agrees with p_k (and so can be merged with it) and a new path p' that is defined by

$$\begin{aligned} p' \triangleq\; & p_j[1 \ldots \text{len}(p_j) - \text{rev}(j,k) + \text{lap}(i,j,k)] \\ & + p_i[(\text{fwd}(i,k) + 1 \ldots \text{len}(p_i)]. \end{aligned} \tag{11}$$

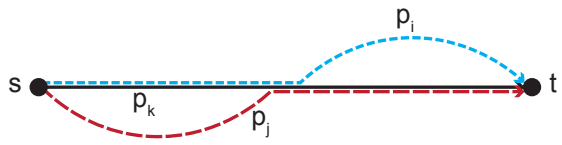


Fig. 3: Path reduction opportunity: We may be able to (1) adjust the flows on p_i and p_j so as to eliminate the need for p_k, or (2) splice p_i and p_j in the region they overlap so that one of the resulting paths coincides with p_k and can be merged; this again requires checking if path flows can be found that satisfy all edge flow constraints.

Again, we would like to see if integers flow for paths p' and p_k can be found so that all edge flow constraints are satisfied. The formulation of this problem is identical to (10), so we apply the same method as above.

We implement checking these path reduction options in a subroutine, try-reduce(i, j, k). The full heuristic is a follows: while there is an unchecked triple (i, j, k), call try-reduce(i, j, k), if it finds a path-reduction operation among paths p_i, p_j, and p_k it performs it; thus reducing $|P|$ by one. Remove unchecked triples that involve the eliminated paths and create new triples to check involving any new or modified paths against current set of paths. Finally, we update the fwd and rev tables along rows and columns corresponding to each new path created by splicing.

c) Other Pairwise Opportunities: The final path-reduction method is to consider just pairs of paths p_i and p_j that have some overlap. We check if rebalancing is possible by adjusting their path flows, such that the flow on one of the paths can be reduced to zero and thus eliminated. We also check if splicing the paths at a vertex where they overlap and then rebalancing allows the flow on one of the paths to be reduced to zero. This is also checked using the same approach as above.

V. Using IFD for RNA Transcript Assembly

As mentioned, there are variety of ways that the edge weights in the splice graph can contain errors, and so it is likely that real RNA read data sets will not yield a flow in the splice graph. We consider two approaches to model these errors and determine an IFD instance to solve in order to obtain a set of paths (assembled transcripts) and weights (abundances).

A. Confidence-based Flow Intervals

Suppose that the probability distribution of measured edge weights in the splice graph is known, e.g. if the true flow value on an edge is x, the measured flow value is distributed according to some probability distribution D_x. For simplicity, we assume that for each true flow value x, the measured edge weight m is independently sampled from $\mathcal{N}(x, (\epsilon x)^2)$, the Gaussian distribution with mean x and standard deviation ϵx; and then m is rounded to the nearest integer. The resulting edge weights will no longer be a flow (with high probability), and so existing flow decomposition methods cannot be directly applied. To use IFD, we need to choose flow intervals for each measured edge weight m. We numerically compute the posterior distribution (assuming a uniform prior), i.e., for a given m what is the probability that m was sampled from $\mathcal{N}(x, (\epsilon x)^2)$. We then compute a confidence intervals for each m value, e.g., 95%. For example, if $[l_{95}(m), u_{95}(m)]$ is the computed 95% confidence interval for a measured edge weight m, the true flow value falls in this interval with probability 0.95. Higher confidence intervals are larger, and so more likely to yield a feasible IFD solution, but intervals that are too large may allow too much flexibility and reduce solution fidelity if too few paths are found. We explore this in § VI-B.

B. Minimum Cost Flow Intervals

We adapt the method of Tomsecu *et al.* [28], that considers the problem of finding a flow that provides the best fit to the experimental abundance data. A fitting-error function $c_e()$ is specified for each edge $e \in E$. Common functions include linear or sum-of-squares error. The objective is then to determine a flow f on G such that $\sum_{e \in E} c_e(|a(e) - f(e)|)$ is minimized. Let this flow be called f_{fit}. For each edge $e \in R$, let

$$\Delta(e) = |a(e) - f_{\text{fit}}(e)|. \tag{12}$$

We then use the interval $[a(e) - \Delta(e), a(e) + \Delta(e)]$ as the flow constraint interval for edge e in the IFD instance. Note that if each $c_e()$ is monotonic then any flow solution to the IFD will have the same fitting-error as f_{fit}. Some intervals found in this way have $l_e = u_e$, and so we also consider adding some slack to these edges to promote the use of Heuristic 2. See §VI-C.

Tomsecu *et al.* find f_{fit} by solving an associated minimum-cost flow problem. The construction below is a slight simplification of their approach: We suppose that a splice graph $G = (V, E)$ is given as input and that for each edge e, $a(e)$ is the experimentally determined abundance of transcripts that traverse e. Source and sink nodes in G are also specified. For convenience, we first add a new node x to G and edges (x, s) and (t, x) to E, for each source s and sink t. We let $a(x, s) = \sum_{e \in \text{out}(s)} a(e) - \sum_{e \in \text{in}(s)} a(e)$ and $a(t, x) = \sum_{e \in \text{in}(t)} a(e) - \sum_{e \in \text{out}(t)} a(e)$. The node x will serve to ensure that a balanced flow is achieved; we require that the flow out of all sources equals the flow into all sinks. We construct a supplemental flow network $G_{\text{sup}} = (V_{\text{sup}}, E_{\text{sup}})$, where $V_{\text{sup}} = V \cup \{s', t', x\}$. For each $e \in E$, including the newly created edges above, we add an edge e_{sup} to E_{sup} and let $f(e) = a(e) + f_{\text{sup}}(e)$; $f_{\text{sup}}(e)$ is the amount by which $a(e)$ must be supplemented (either positively or negatively) in order to achieve an optimal flow. The cost of sending flow on e_{sup} is given by $c_e(|f_{\text{sup}}(e)|)$ and e_{sup} is provided infinite capacity. For each $v \in V \cup \{x\}$, we compute

$$\text{a-exc}(v) \triangleq \sum_{e \in \text{in}(v)} a(e) - \sum_{e \in \text{out}(v)} a(e).$$

In order to achieve conservation of flow at all vertices in the final flow solution, we must have that

$$\sum_{e \in \text{out}(v)} f_{\text{sup}}(e) - \sum_{e \in \text{in}(v)} f_{\text{sup}} = \text{a-exc}(v), \tag{13}$$

for all $v \in V \cup \{x\}$. To achieve this at minimum cost, we create following min-cost flow instance: If a-exc$(v) > 0$, add an edge (s', v) with capacity a-exc$(v) > 0$ and cost 0. Likewise, if a-exc$(v) < 0$, add an edge (v, t') with capacity $-$a-exc$(v) > 0$ and cost 0. The flow requirement is to send

$$A \triangleq \sum_{v : \text{a-exc}(v) > 0} \text{a-exc}(v)$$

units of flow from s' to t'. We observe that A is the capacity of both the cut $(s', V_{\text{sup}} \setminus s')$ and the cut $(V_{\text{sup}} \setminus t', t')$, thus all outgoing edges of s' and incoming edges of t' must be

saturated to achieve the desired flow. If this flow is achieved, it is easy to see that (13) will hold at each $v \in V \cup \{x\}$ and so f will be a flow in G. As shown in [28], f minimizes the fitting-error.

C. Identifying Sources and Sinks

Existing RNA transcript prediction methods based on flow decomposition, such as Traph [28], Catfish [25] and Toboggan [14] assume that the source and sink nodes are known in the flow network. These nodes correspond to exons that appear at the beginning or end of transcripts. While some start/end exons may be clear (e.g., nodes for which there are only outgoing/incoming edges with measured weights), this is not a completely reliable method if, for example, an exon appears at the beginning of one transcript and in the middle of another. We propose a simple approach to identify sources and sinks using IFD: create a new source node s^* and sink node t^* and then, for all $v \in V$, add edges (s^*, v) and (v, t^*) to the network, with associated flow constraint intervals of $[0, \infty)$. Then, all v such that $f(s^*, v) > 0$ in the IFD solution are sources, and all v such that $f(v, t^*) > 0$ in the IFD solution are sinks.

VI. Experimental Results

We tested our proposed IFD algorithm on splice graphs constructed from simulated human transcriptomes, drawing from the extensive test corpus used in two previous studies [13], [27]. This data consists of flow graphs generated from simulated transcriptomes from three different species: human, mouse, and zebrafish. Transcripts (paths) and their abundances (weights) were simulated for each species using Flux-Simulator [10], and these were superimposed to construct splice graphs. (Reads were not used.) For this study, we considered only the human data set, which had 524,212 splice graph instances with at least two paths. These nontrivial instances averaged 18.8 nodes, 22.6 edges, and 2.8 paths. We simulated errors on the weights of the splice graph edges; see §VI-A.

Because these reads were simulated from real gene data, the simulated paths are considered to be a ground truth minimal decomposition of paths for the resulting flow network. While it may not always be the case that the simulated paths are in fact a minimal decomposition (there could be a decomposition with a smaller number of paths, using a different set of paths than those that generated the flow graph), it was found in [13] that in most cases (at least 99%), the simulated paths are indeed a minimal decomposition for the network.

In order to measure the quality of the paths and weights predicted by our algorithm, we compute the *weighted Jaccard similarity (WJS)* between the groundtruth set of paths P^* and the predicted set of paths P. For each IFD instance, we arbitrarily order the set of all paths in the union of P^* and P and create vectors X^* and X containing the corresponding weights. If X^* and X both have length n, then the weighted Jaccard similarity between them is defined to be

$$\text{WJS}(X^*, X) = \frac{\sum_{i=1}^{n} \min X^*[i], X[i]}{\sum_{i=1}^{n} \max X^*[i], X[i]}.$$

$\text{WJS}(X^*, X) \in [0, 1]$, with $\text{WJS}(X^*, X) = 1$ only when the predicted paths and their weights exactly match the groundtruth paths and their weights.

We implemented the IFD algorithm described in §IV in Python, making use of some graph data structures and processing functionality developed as part of Toboggan [14], and using Google's OR-Tools [1] to solve *min-cost flow* and *maxflow* instances. For simplicity and scalability, the Greedy-Width algorithm [30] was used to construct the initial path decomposition and as in [12] we use a modified version of Dijkstra's algorithm to find bottleneck paths. The Python code and link to the data used to generate the results reported in this section can be found at https://github.com/msu-alglab/ifd.

A. Simulating Data Errors

As discussed in §I, there are variety of ways that splice graph edge weights can contain errors. To create data with such errors, we assume that for each true flow weight x, the actual measured edge weight m is independently sampled from $\mathcal{N}(x, (\epsilon x)^2)$, the Gaussian distribution with mean x and standard deviation ϵx; we assume the sample m is then rounded to the nearest integer. For this experiment we fixed $\epsilon = 0.05$. The resulting edge weights will no longer be a flow (with high probability) and so existing flow decomposition methods cannot be directly applied. We investigate both confidence interval and minimum cost flow interval approaches described in §V-A and §V-B, respectively, in order to determine path decomposition solutions.

B. Confidence Interval Experiment

To create IFD instances from flow graphs with errors, we need to choose flow intervals for each measured edge weight m. To use the method from §V-A, we numerically compute the posterior distribution (assuming a uniform prior), i.e., for a given m what is the probability that m was sampled from $\mathcal{N}(x, (\epsilon x)^2)$. For example, if $[l_{95}(m), u_{95}(m)]$ is the computed 95% confidence interval for a measured edge weight m, the true flow value falls in this interval with probability 0.95.

Our results are reported in Table I, by confidence interval. Because not all instances are solvable at every confidence level, and instances with larger $|P^*|$ are generally more difficult to solve, we report the average WJS and $|P|$ weighted according to the distribution of $|P^*|$ in the full data set. The best WJS is found when using a 95% confidence interval. Surprisingly, the fewest paths are found using the one of the smaller tested confidence intervals, 85%. One possible reason for this is that the graphs with error that are solvable with smaller intervals must have perturbed weights that are close to the true weights, making them "easier" to solve. Additionally, for those graph instances which have smaller solutions for lower confidence intervals, it is often the case

that the initial solution found by Greedy-Width is smaller for the lower confidence intervals than higher confidence intervals. This suggests that a substituting a more accurate method such as Catfish may reduce the number of paths found at higher intervals. However, our ultimate goal is accurate reconstruction of isoforms and abundances, which we measure with WJS. Wider intervals are necessary to find solutions for most instances. For example, a 90% confidence interval finds solutions in less than half of our test cases. This is because the test cases tend to have many edges (22.6 on average), meaning that there are many opportunities for non-overlapping intervals to be created in a graph, even when we expect that each interval contains the true weight 90% of the time.

Though not reported in Table I, we also note that, as expected, Heuristic 2a (rebalancing) and Heuristic 2b (splice and merge) occur more frequently as the interval size increases. At all confidence levels tested, rebalancing occurred much more frequently than splicing and merging. For example, at a 95% interval, the weighted average of rebalances per solvable graph instance was 0.96, while the weighted average of splice and merges per solvable graph instance was 0.34.

TABLE I: Experimental results using a simulated human transcriptomic data set with simulated errors. For varying confidence intervals, we report the percentage of inputs for which the IFD instance was solvable. For solvable instances we also report the weighted average of weighted Jaccard similarity scores and number of predicted paths.

confidence interval	IFD solvable	weighted avg. WJS	weighted avg. $\lvert P\rvert$
99%	99.9%	0.810	2.673
95%	83.3%	**0.822**	2.768
90%	46.6%	0.803	2.813
85%	36.4%	0.798	**2.643**
80%	32.8%	0.792	2.752

C. Minimum Cost Flow Interval Experiment

We next test the IFD algorithm using the minimum cost flow method for generating intervals, as described in §V-B. We use a linear fitting-error function for simplicity. After generating intervals, we proceed with the five steps outlined in Fig. 2. We refer to this method as *mc-IFD*. Because the intervals are generated as $[a(e) - \Delta(e), a(e) + \Delta(e)]$, they always include the best fit flow for the measured graph according to the specified fitting-error function. Thus, every IFD instance created in this way is solvable. We compare the quality of solutions found by our algorithm to the quality of solutions found by setting the best-fit flow and then running Greedy-Width, as in [28], which we refer to as the *mc-fd* method.

The minimum cost flow method for finding intervals generates many small intervals. In fact, on this data set, almost every path found by Greedy-Width contains at least one edge with $l_e = u_e$. Because there is so little slack in some of the intervals, no splice or rebalance opportunities are found in this data set, so our results are nearly identical to those found by the mc-fd method. The only differences come from the differences in initial flow solution found by our maxflow method, described in §III, and the min-cost flow used in the mc-fd method. We also tested a widening heuristic for such zero-width intervals according to the following rules: if $l_e = u_e = 0$, set $l_e = 0, u_e = 5$. If $l_e = u_e > 0$, reassign l_e to be $0.8l_e$ and u_e to be $1.2u_e$, both rounded to the nearest integer. We refer to this as the *mc-IFD+* method.

Our results are reported in Table II. We see that, by adding some slack in the zero-width intervals, we are able to reduce the number of paths found, though at the expense of a very slightly lower WJS. Overall, the min-cost flow method for finding intervals yields higher WJS than the confidence interval method we test in the previous subsection, but requires more paths.

TABLE II: Experimental results using a simulated human transcriptomic data set with simulated errors. We compare the weighted Jaccard similarity and number of paths found by applying Greedy-Width to the minimum-cost flow found as in [28] (mc-fd) with our IFD approach (mc-IFD). We also try widening width-zero intervals in our IFD approach (mc-IFD+).

method	avg. WJS	avg. $\lvert P\rvert$
mc-fd	0.902	4.387
mc-IFD	0.902	4.387
mc-IFD+	0.882	4.120

D. Unknown Sources/Sinks Experiment

To test the ability of our method to recover the ground truth paths and their abundances even when the source and sink nodes are unknown, we remove the given source node and then proceed as described in §V-C: we create new source and sink s^* and v^*, add edges from s^* to all other nodes and to t^* from all other nodes with intervals $[0, \infty)$. Then, we apply the confidence-based method for creating intervals. Results are reported in Table III. For both the 95% and 85% confidence levels (the best confidence levels to maximize WJS or minimize $|P|$ respectively in Table I), nearly all IFD instances tested contained a flow ($> 99\%$). Despite the true starts and ends of transcripts being unknown, the IFD method is able to achieve relatively high WJS.

TABLE III: Experimental results using IFD to recover paths and weights in a splice graph where the sources and sinks are unknown.

method	weighted avg. WJS	weighted avg. $\lvert P\rvert$
95%	0.783	3.17
85%	0.770	4.33

VII. Discussion

In this work we proposed the Inexact Flow Decomposition problem and an algorithmic strategy for solving it based on first finding a feasible flow, possibly modifying this flow, then finding an initial path decomposition and refining that using path splicing and merging. To our knowledge, ours is the first algorithm for path decomposition to first find an initial solution

and then attempt to improve it. Other heuristics might be effective for reducing the initial number of paths needed for an IFD instance. We remark that it is possible to adapt the dynamic-programming based FPT scheme of [13] to solve the IFD problem on directed acyclic graphs (the only step that needs modification is to incorporate the interval constraints into the integer linear systems that are checked for feasibility, as path solutions are found in each sub-problem). The IFD problem may be of interest in other contexts such as transportation route prediction [21] or telecommunications and computer network routing [12], [30], where flow measurements might contain errors or be inexact.

Our experimental results suggest that the IFD model has promise for isoform assembly and quantification. In the presence of random errors on splice graph edges, our heuristic algorithms predict paths and weights with high weighted Jaccard similarity to the groundtruth paths and edges. However, there are a number of future directions that would further the IFD approach. The first of these would be the development of biologically-based uncertainty values for the edge weights in the graph. While many methods tend to account for these biases in a static form, newer approaches by [19] and [2] have developed methods that use observed characteristics within the data to identify and correct for specific biases in the data in a sample-specific manner. However, [19] is written specifically for isoform quantification rather than splice isoform assembly and [2] only accounts for read-mapping biases. A second extension is the benchmarking of the IFD approach on both simulated and real RNA-Seq short read data. This was outside of the scope of this particular investigation as the focus was on the development and assessment of the IFD algorithm on splice graphs, but it is an important next step in validating the usefulness of IFD for transcript assembly.

REFERENCES

[1] Google's OR-Tools, developers.google.com/optimization.

[2] D. Aguiar, L.-F. Cheng, B. Dumitrascu, F. Mordelet, A. Pai, and B. Engelhardt. Bayesian nonparametric discovery of isoforms and individual specific quantification. *Nature Communications*, 9, 2018.

[3] C. Berge. *Graphs*. 1984.

[4] A. Conesa, P. Madrigal, S. Tarazona, D. Gomez-Cabrero, A. Cervera, A. McPherson, M. Szcześniak, D. Gaffney, L. Elo, X. Zhang, et al. A survey of best practices for RNA-Seq data analysis. *Genome Biology*, 17(1):13, 2016.

[5] C. Del Fabbro, S. Scalabrin, M. Morgante, and F. Giorgi. An extensive evaluation of read trimming effects on Illumina NGS data analysis. *PloS One*, 8(12):e85024, 2013.

[6] L. Ding, E. Rath, and Y. Bai. Comparison of alternative splicing junction detection tools using RNA-Seq data. *Current Genomics*, 18(3):268–277, 2017.

[7] J. Edmonds and R. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[8] P. Engström, T. Steijger, B. Sipos, G. Grant, A. Kahles, T. Alioto, J. Behr, P. Bertone, R. Bohnert, D. Campagna, et al. Systematic evaluation of spliced alignment programs for RNA-Seq data. *Nature Methods*, 10(12):1185, 2013.

[9] T. Gatter and P. Stadler. Ryūtō: network-flow based transcriptome reconstruction. *BMC Bioinformatics*, 20(1):190, 2019.

[10] T. Griebel, B. Zacher, P. Ribeca, E. Raineri, V. Lacroix, R. Guigó, and M. Sammeth. Modelling and simulating generic RNA-Seq experiments with the flux simulator. *Nucleic Acids Research*, 40(20):10073–10083, 2012.

[11] Y. Han, S. Gao, K. Muegge, W. Zhang, and B. Zhou. Advanced applications of RNA sequencing and challenges. *Bioinformatics and Biology Insights*, 9:BBI–S28991, 2015.

[12] T. Hartman, A. Hassidim, H. Kaplan, D. Raz, and M. Segalov. How to split a flow? In *INFOCOM, 2012 Proceedings IEEE*, pages 828–836. IEEE, 2012.

[13] K. Kloster, P. Kuinke, M. O'Brien, F. Reidl, F. Sanchez Villaamil, B. Sullivan, and A. 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.

[14] K. Kloster, P. Kuinke, M. O'Brien, F. Reidl, F. Sanchez Villaamil, B. Sullivan, and A. van der Poel. Toboggan: Version 1.0, June 2017.

[15] S. Kumar, A. Vo, F. Qin, and H. Li. Comparative assessment of methods for the fusion transcripts detection from RNA-Seq data. *Scientific Reports*, 6:21597, 2016.

[16] N. Lahens, I. Kavakli, R. Zhang, K. Hayer, M. Black, H. Dueck, A. Pizarro, J. Kim, R. Irizarry, R. Thomas, et al. IVT-seq reveals extreme bias in RNA sequencing. *Genome Biology*, 15(6):R86, 2014.

[17] L. Manley, D. Ma, and S. Levine. Monitoring error rates in Illumina sequencing. *Journal of Biomolecular Techniques*, 27(4):125, 2016.

[18] J. Mattick and J. Rinn. Discovery and annotation of long noncoding RNAs. *Nature Structural & Molecular Biology*, 22(1):5, 2015.

[19] A. McDermaid, X. Chen, Y. Zhang, J. Xie, C. Wang, and Q. Ma. GeneQC: A quality control tool for gene expression estimation based on RNA-sequencing reads mapping. *BioRxiv*, page 266445, 2018.

[20] J. Mehta. Sequencing small RNA: introduction and data analysis fundamentals. In *RNA Mapping*, pages 93–103. Springer, 2014.

[21] S. Micka and B. Mumey. The minimum road trips problem. In *Fall Workshop on Computational Geometry*, pages 1–4, 2017.

[22] B. Mumey, S. Shahmohammadi, K. McManus, and S. Yaw. Parity balancing path flow decomposition and routing. In *Globecom Workshops (GC Wkshps), 2015 IEEE*, pages 1–6. IEEE, 2015.

[23] S. Parekh, C. Ziegenhain, B. Vieth, W. Enard, and I. Hellmann. The impact of amplification on differential expression analyses by RNA-Seq. *Scientific Reports*, 6:25533, 2016.

[24] M. Pertea, G. Pertea, C. Antonescu, T.-C. Chang, J. Mendell, and S. Salzberg. StringTie enables improved reconstruction of a transcriptome from RNA-Seq reads. *Nature Biotechnology*, 33(3):290, 2015.

[25] M. Shao and C. Kingsford. Catfish. https://github.com/Kingsford-Group/catfish.

[26] M. Shao and C. Kingsford. Scallop enables accurate assembly of transcripts through phasing-preserving graph decomposition. *BioRxiv*, page 123612, 2017.

[27] M. Shao and C. 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.

[28] A.. Tomescu, A. Kuosmanen, R. Rizzi, and V. Mäkinen. A novel min-cost flow method for estimating transcript expression with RNA-Seq. In *BMC Bioinformatics*, volume 14, page S15. BioMed Central, 2013.

[29] C. Trapnell, B. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. Van Baren, S. Salzberg, B. Wold, and L. Pachter. Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature biotechnology*, 28(5):511, 2010.

[30] 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.

[31] Z. Wang, M. Gerstein, and M. Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57, 2009.

[32] C. Williams, A. Baccarella, J. Parrish, and C. Kim. Trimming of sequence reads alters RNA-Seq gene expression estimates. *BMC Bioinformatics*, 17(1):103, 2016.