Finding ranges of optimal transcript expression quantification in cases of non-identifiability

Cong Ma*[1], Hongyu Zheng*[1], and Carl Kingsford†[1]

[1]Computational Biology Department, School of Computer Science, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA

December 13, 2019

Abstract

Current expression quantification methods suffer from a fundamental but under-characterized type of error: the most likely estimates for transcript abundances are not unique. Probabilistic models are at the core of current quantification methods. The scenario where a probabilistic model has multiple optimal solutions is known as non-identifiability. In expression quantification, multiple configurations of transcript expression may be equally likely to generate the sequencing reads and the underlying true expression cannot be uniquely determined. It is still unknown from existing methods what the set of multiple solutions is or how far the equally optimal solutions are from each other. Such information is necessary for evaluating the reliability of analyses that are based on a single inferred expression vector and for extending analyses to take all optimal solutions into account. We propose methods to compute the range of optimal estimates for the expression of each transcript when the probabilistic model for expression inference is non-identifiable. The accuracy and identifiability of expression estimates depend on the completeness of input reference transcriptome, therefore our method also takes an assumed percentage of expression from combinations of known junctions into consideration. Applying our method on 16 Human Body Map samples and comparing with the single expression vector quantified by Salmon, we observe that the ranges of optimal abundances are on the same scale as Salmon's estimate. Analyzing the overlap of ranges of optima in differential expression (DE) detection reveals that the majority of predictions are reliable, but there are a few unreliable predictions for which switching to other optimal abundances may lead to similar expression between DE conditions. The source code can be found at https://github.com/Kingsford-Group/subgraphquant.

Keywords: expression quantification, alternative splicing, uncertainty, non-identifiability, differential expression.

1 Introduction

With the wide usage of gene or transcript expression data, we have seen improvement in probabilistic models of expression quantification [1–4], as well as characterization and evaluation of the errors of quantified expression [5–7]. However, there is an under-characterized type of estimation error, due to the non-uniqueness

*Equal contribution

†To whom correspondence should be addressed: carlk@cs.cmu.edu

of solutions to the probabilistic model. When multiple sets of expression of transcripts give rise to the sequencing library with equal probability, current methods only output one of the optimal solutions. It is usually unknown what the other optimal solutions are or how different they are from each other.

This type of estimation error, due to the non-uniqueness of optimal solutions, is fundamental and almost inevitable. The non-uniqueness of solutions to an optimization model is known as model non-identifiability under the assumption of infinite observations. Identifiability of a model is usually determined by the form of the objective function and the dimension of parameters in the optimization problem. For example, as shown in Figure 1, the toy example seeks a set of transcript abundances that best explain the splicing junction read counts in terms of ℓ_2 distances. The optimal abundances are non-identifiable. Given any optimal abundances, another optimal solution can be constructed by decreasing the abundances of transcripts $\{1-4-5, 2-4-6, 3-4-7\}$ by a fixed amount and then increasing the abundances of transcripts $\{1-4-7, 2-4-6, 3-4-5\}$ by the same amount, which leads to the same objective value. (The notation $1-4-5$ indicates the transcript that contains exon 1, exon 4, and exon 5.) The interdependence between parameters is one of the causes of the non-identifiability phenomenon.

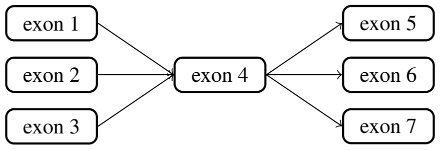

Figure 1: Example of nine transcripts (paths in the splice graph) using seven exons.

Assessing the potential error due to non-identifiability helps increase the accuracy of analyses that are based on transcript expression data. Expression data has been used in differential expression detection [8] to study gene function, or treated as features to predict disease and treatment outcome [9, 10]. Without considering the estimation error in expression, these analyses may miss the true signal, output false detections or make wrong predictions. Recent differential expression detection methods [11, 12] include estimation error in their models and show an improved accuracy of the detected differentially expressed transcripts. Integrating the multiple optimal estimates due to non-identifiability into consideration may further improve existing methods.

Previous work involving the non-identifiability of the expression model does not directly output all optimal abundances or provide estimates on potential estimation error. Lacroix et al. [13] and Hiller et al. [14] developed methods to identify the transcripts that have non-unique expression abundances but their methods do not provide information about the corresponding quantification error. Roberts and Pachter [15] incorporated the detection of non-identifiable abundances into their quantification method, and designed a tool to output the identifiability flag of each transcript along with its single expression estimate. Bernard et al. [16] proposed a network-flow-based model that has a similar premise and approach (Section 2.2, Section 2.3) to our work. The model is reparameterized using a network flow based on a fragment graph, then quantified with ℓ_1 regularization followed by a flow decomposition, without taking identifiability of transcript expression into consideration (as shown in our toy example, ℓ_1 regularization may not affect identifiability). Their proposed approach also only works for single-end, fixed-length reads, as phasing paths that include each other introduce ambiguity in the fragment graph and the resulting graph can no longer be quantified correctly. LeGault and Dewey [17] also introduced a network-flow-based model and specifically focused on the identifiability of the model, but the method requires additional assumptions for phasing reads, and does not provide further information about the multiple optimal abundances of transcripts.

Research on transcriptome assembly or third-generation sequencing with long reads has shown that the current transcript annotation database tends to be incomplete [18–21], and transcripts may be quantified erroneously because the unannotated transcripts responsible for the reads are not present in the reference. Therefore, it is important to quantify all optimal abundances under various reference completeness assumptions, especially under the incomplete reference assumption. Under such assumption, the parameter space includes the expressions of novel transcripts and the dimension is expanded, making the parameters more susceptible to non-identifiability.

In this work, we propose methods to construct the set of optimal estimates for transcript abundances under three assumptions about reference completeness given a probabilistic model. The current quantification probabilistic models are usually convex, thus the range of optima for each transcript is a closed interval. We use the phrase "range of optima" to refer to the set of optima projected to abundance of a single transcript. Assuming a complete reference, we propose a linear programming formulation to bound the range of optimal abundances for each transcript based on a phasing aware reparameterization of the model. Assuming that the reference transcriptomis incomplete but the splice graph is correct, we propose a set of max-flow-based algorithms on an adapted splice graph for reconstructing the set of optima. Finally, under the assumption that the reference is incomplete but a known, given percentage of expression is contributed by the reference, we derive the range of optima using an affine combination of the optima constructed in the previous two cases.

Using our method, we evaluate the distance between the single solution of the quantifier Salmon [4] to the other abundances in the range of optima on Human Body Map samples. We observe that most of the Salmon estimates are at the boundaries of the range, and the upper bounds of the range of optimal abundances tend to be within 10 fold of Salmon's estimates.

Applying our method on sequencing samples of MCF10 cell line, we use the range of optimal expression to evaluate the detected differentially expressed transcripts. Between the group of samples treated with and without epidermal growth factor (EGF), 257 transcripts are detected as differentially expressed based on the single estimates. The majority are reliable after considering the ranges of optima when assuming the reference transcriptome is somewhat complete and contributes to more than 40% of the expression, while there are still a few unreliable detections for which the two groups' ranges of optimal expression overlap.

2 Methods

2.1 Overview

Our goal is to calculate the range of optima for each transcript and to analyze the effect of non-identifiability on quantification. Our first step is to reparameterize the probabilistic model with a more identifiable parameter set, called path abundances. Assuming all reads originate from known transcripts (complete reference), we calculate the range of optima by a linear-programming-based reallocation process from known quantification results. We cover the reparameterization and reallocation process in Section 2.2.

On the other end of the completeness spectrum, we have the assumption that reads may also be generated from unannotated transcripts with existing splicing junctions but not present in reference transcriptome. Existing approaches are ill-suited for this setting, as the number of transcripts is too large to enumerate. We solve this problem in three steps. We describe how to derive the constraints governing the new set of parameters by first modeling the matching of phasing paths to transcripts as a multiple pattern matching problem, then defining a network flow over the matching automaton (Section 2.3). We next develop efficient inference algorithms for the resulting flow (Section 2.4) and calculate the range of optima via a new algorithmic framework called subgraph quantification (Section 2.5).

In Section 2.6, we investigate the rest of the reference completeness spectrum by proposing a hybrid model of read generation, and design a reliability check for differential expression calls against non-identifiability. Figure 2 provides an overview of different components of our methods.

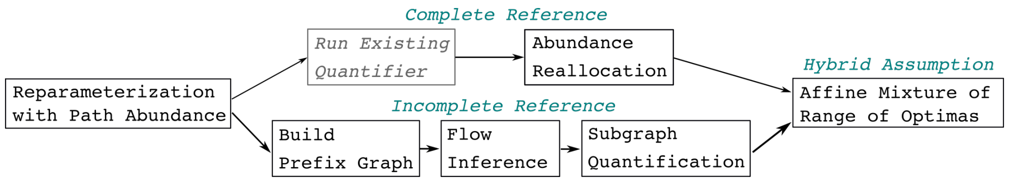

Figure 2: Steps to derive range of optima under different reference completeness assumptions. Grey blocks indicate steps built upon existing methods.

2.2 Reparameterized Transcript Quantification

We start with the core model of transcript quantification at the foundation of most modern methods [1–4]. Assume the paired-end reads from some RNA-seq experiment are error-free and uniquely aligned to a reference genome with possible gaps. We denote the set of reads as F, the set of transcripts as $\mathcal{T} = \{T_1, T_2, \cdots, T_n\}$ with corresponding length $l_1, l_2, \cdots, l_n$ and abundance (copies of molecules) $c_1, c_2, \cdots, c_n$. The transcripts per million (TPM) values are calculated by normalizing $\{c_i\}$ and multiplying by 10^6. Under the generative model, the probability of observing F is:

$$P(F \mid \mathcal{T}, c) = \prod_{f \in F} \sum_{i \in A(f)} P(T_i) P(f \mid T_i)$$

$A(f)$ is the set of transcripts onto which f can map. Let $D(l)$ be the distribution of generated fragment length. With absence of bias correction, $P(f \mid T_i)$ is proportional to $D(f) = D(l(f))$, where $l(f)$ denotes the fragment length inferred from mapping. Define the effective length for T_i as $\hat{l}_i = \sum_{j=1}^{l_i} \sum_{k=j}^{l_i} D(k - j + 1)$ (which can be interpreted as the total probability for T_i to generate a fragment), and $P(f \mid T_i) = D(f)/\hat{l}_i$. Probability of generating a fragment from T_i is assumed to be proportional to its abundance times its effective length, meaning $P(T_i) \propto c_i \hat{l}_i$. We discuss other definitions of effective length in Section S1.1 and refer to previous papers [1–4, 22] for more detailed explanation of the model. This leads to:

$$P(F \mid \mathcal{T}, c) = \prod_{f \in F} (\sum_{i \in A(f)} c_i) D(f) / (\sum_{T_i \in \mathcal{T}} c_i \hat{l}_i)$$

We now propose an alternative view of the probabilistic model with paths on splice graphs in order to derive a compact parameter set for the quantification problem.

Splice graphs are succinct representations of alternative splicing events in a gene. A splice graph is a directed acyclic graph where each vertex represents a full or partial exon and each edge either connects two adjacent partial exons or corresponds to a splicing junction. Two special vertices S and T represent start and end of transcripts instead of exons. We assume the splice graph is constructed so each transcript can be uniquely mapped to a $S - T$ path $p(T_i)$ on the graph, and the read library satisfies that each fragment f can be uniquely mapped to a (non $S - T$) path $p(f)$ on the graph. With this setup, $i \in A(f)$ if and only if $p(f)$ is a subpath of $p(T_i)$, or $p(f) \subset p(T_i)$.

We now define $c_p = \sum_{i:T_i \in \mathcal{T}, p \subset p(T_i)} c_i$ to be the total abundance of transcripts including path p, called a **path abundance**, and $\hat{l}_p = \sum_{i=1}^{l_k} \sum_{j=i}^{l_k} \mathbf{1}(p(T_k[i,j]) = p) D(j-i+1)$ called **path effective length**, where $T_k[i,j]$ is the fragment generated from transcript k from base i to base j and $\mathbf{1}(\cdot)$ is the indicator function. Intuitively, the path effective length is the total probability of sampling a fragment that maps exactly to the given path. Next, let $\mathcal{P}$ be the set of paths from the splice graph satisfying $\hat{l}_p > 0$. We can re-calculate the normalization term of the model by $\sum_{T_i \in \mathcal{T}} c_i \hat{l}_i = \sum_{p \in \mathcal{P}} c_p \hat{l}_p$ (proof in Section S1.2). The likelihood can now be rewritten as follows:

$$P(F \mid \mathcal{T}, c) = \prod_{f \in F} (\sum_{j:p(f) \subset p(T_j)} c_j) D_f / (\sum_{p \in \mathcal{P}} c_p \hat{l}_p) \propto \prod_{f \in F} c_{p(f)} / (\sum_{p \in \mathcal{P}} c_p \hat{l}_p)$$

As shown here, we reparameterize the model with c_p, the path abundance. When the transcriptome is complex, the dimension of c_p is lower than the number of transcripts, a direct manifestation of the non-identifiability problem. In practice, we further reduce the size of $\mathcal{P}$ by discarding long paths with small $\hat{l}_p$ and no mapped reads, as they contribute little to the likelihood function (See Section S1.3 for details). Our model also enables certain bias corrections as described in Section S1.4.

With the reparameterized model, we now derive the range of optimal expression values for each transcript assuming all reads are generated from known transcripts. The reference abundances $\{c_i^R\}$ are derived with existing tools, such as Salmon or kallisto [3]. We assume $\{c_i^R\}$ yields the optimal likelihood. If the model is non-identifiable, it means there are other sets of transcript abundances $\{c_i\}$ that yield exactly the same likelihood. As we have shown above, it suffices for $\{c_i\}$ to generate an identical set of path abundances as $\{c_i^R\}$ to be optimal. The set of transcript abundance that satisfies this condition and therefore indistinguishable from c^R is defined by the linear system $\sum_{i:p \subset p(T_i)} c_i = \sum_{i:p \subset p(T_i)} c_i^R, \forall p \in \mathcal{P}$ with nonnegative c_i. We can calculate the range of c_i in the indistinguishable set by maximizing or minimizing c_i in this system with linear programming.

2.3 Representing Path Abundance with Prefix Graphs

Assuming reads can be generated by novel transcripts, we cannot apply the reallocation trick as described last section because we do not know the optimal abundances. In this setup, optimizing with transcript abundances as parameters is prohibitively expensive, since the number of novel transcripts with known junctions can be exponential in the number of exons. In contrast, assuming most phasing paths contain few exons, the size of $\mathcal{P}$ grows polynomially in the number of exons. We now focus on representing c_p without individual c_i, to form an optimization problem with $\{c_p\}$ directly.

We first observe some properties of $\{c_p\}$. For example, if p is a subpath of p', $c_p \geq c_{p'}$ as any transcript containing p also contains p'. However, this is not sufficient to guarantee $\{c_p\}$ corresponds to a valid transcriptome. Our goal in this section is to find a low dimensional representation of $\{c_p\}$ while ensuring the existence of a corresponding transcriptome $\{c_i\}$.

The most natural idea is to use the splice graph flow to represent $\{c_p\}$. However, this does not work when p contains three or more exons. Informally, this can be solved by constructing higher-order splice graphs (as defined in LeGault and Dewey [17] for example), or fixed-order Markov models, but the size of the resulting graph grows exponentially fast and the resulting model may not be identifiable itself. We choose to "unroll" the graph just as needed (roughly corresponding to a variable-order Markov model). In a similar spirit, an alternative approach [16] constructs a fragment graph where each vertex corresponds to a valid phasing path and it is assumed that every transcript can be mapped uniquely to a path on the fragment path. However, as we discussed before, this assumption no longer holds when paired-end reads are taken into consideration. For the deep connection between our model and pattern matching problems, we explain our model as follows.

Our proposed method heavily relies on the analysis of a seemingly unrelated problem: Imagine we are given the set of paths $\mathcal{P}$, and for a transcript $T_i \in \mathcal{T}$, we want to determine the set of paths that are subpath of T_i, reading one exon of T_i at a time. We can treat this as an instance of multiple pattern matching, where the nodes in the splice graph (which are also the set of exons in the gene) constitute the alphabet, $\mathcal{P}$ is the set of patterns (short string to match against), and $\mathcal{T}$ is the set of texts (long strings to identify pattern in). In this setup, the Aho-Corasick algorithm [23] performs the matching quickly by using a finite state automaton (FSA) to maintain the longest suffix of current text that could extend and match a pattern in the future.

We are interested in the Aho-Corasick FSA constructed from $\mathcal{P}$ to solve the matching, and we further modify the finite state automaton as follows. Transitions between states of the FSA, called dictionary suffix links, indicate the next state of the FSA for all possible next exons. We do not need the links for all characters (exons), as we know $t \in \mathcal{T}$ is a $S - T$ path on the splice graph. If x is the last seen character, the next character y must be a successor of x in the splice graph, and we only generate the state transitions for this set of movements.

With an FSA, we can construct a directed graph from its states and transitions. Formally:

Definition 1 (Prefix Graph)**.** *Given splice graph G_S and set of splice graph paths $\mathcal{P}$ (assuming every single-vertex path is in $\mathcal{P}$), we construct the corresponding prefix graph G as follows:*

The vertices V of G are the splice graph paths p such that it is a prefix of $p'' \in \mathcal{P}$. For $p \in V$, let x be the last exon in p. For every y that is a successor of x in the splice graph, let p' be the longest $p' \in V$ that is a suffix of py (appending y to p). We then add an edge from p to p'.

The source and sink of G are the vertices corresponding to splice graph paths $[S]$ and $[T]$, where $[x]$ denotes a single-vertex path. The set $AS(p)$ is the set of vertices p' such that p is a suffix of p'.

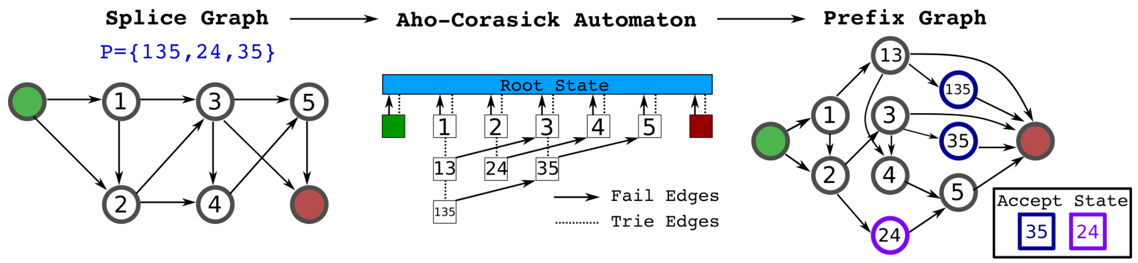

Figure 3: An example construction of Prefix Graph. The green and red nodes are S and T of the splice graph, and $[S]$ and $[T]$ of the prefix graph. We draw the trie plus fail edges for the Aho-Corasick automaton as it reduces cluttering, and one can derive the dictionary suffix link from both edge sets. The colored nodes in prefix graph are the states in $AS(35)$ and $AS(24)$, as indicated in figure.

Intuitively, the states of the automaton are the vertices of the graph, and are labeled with the suffix in consideration at that state. The (truncated) dictionary suffix links indicate the state transitions upon reading a character, and naturally serve as the edges of the graph. All transcripts start with S, end with T and there is no path in $\mathcal{P}$ containing either of them as they are not real exons, so there exist two vertices labeled $[S]$ and $[T]$. We call them the source and sink of the prefix graph, respectively. For $p \in \mathcal{P}$, $AS(p)$ denotes the set of states in FSA that recognizes p.

The resulting prefix graph has several interesting properties that resemble the splice graph it was constructed from. $[S]$ and $[T]$ work similarly to the source and sink of the splice graph, as every transcript can be mapped to an $[S]-[T]$ path on the prefix graph by feeding the transcript to the finite state automaton and recording the set of visited states, excluding the initial empty state (the first state after that is always $[S]$). Conversely, a $[S]-[T]$ path on the prefix graph can also be mapped back to a transcript, as it has to follow dictionary suffix links which by our construction can be mapped back to edges in the splice graph. The prefix graph is also a DAG: If there is a cycle in the prefix graph, it implies an exon would appear twice in a transcript.

A prefix graph flow is a bridge between the path abundance $\{c_p\}$ and the transcriptome $\{c_i\}$:

Theorem 1. *Every transcriptome can be mapped to and from a prefix graph flow. The path abundance is preserved during the mapping and can be calculated exactly from prefix graph flow:* $c_p = \sum_{s\in AS(p)} f_s$*, where* f_s *is the flow through vertex* s*.*

Proof. Using the path mapping between splice graph and prefix graph, we can map a transcriptome onto the prefix graph as a prefix graph flow, and reconstruct a transcriptome by decomposing the flow and map each $[S]-[T]$ path back to the splice graph as a transcript.

To prove the second part, let $\{c'_p\}$ be the path abundance calculated from prefix graph. We will show $\{c_p\} = \{c'_p\}$ for any finite flow decomposition of the prefix graph. Note that since no path appears twice for a transcript, if T_i contains p, it will be recognized by the FSA exactly once. This means the $[S]-[T]$ path that T_i maps to intersects with $AS(p)$ by exactly one vertex in this scenario, and it contributes the same to c'_p and c_p. If T_i does not contain p, it contributes to neither c'_p nor c_p. This holds for any transcript and any path, so the two definitions of path abundance coincide and are preserved in mapping from transcriptome to prefix graph flow. Since the prefix graph flow is preserved in flow decomposition, the path abundance is preserved too. □

This connection is important because while it is hard to check if some path abundance $\{c_p\}$ is valid, it is easy to check if a flow on the prefix graph is valid: The only conditions are nonnegative flow and flow balance. Such simplicity enables us to directly optimize over $\{c_p\}$ without using $\{c_i\}$ during optimization and reconstruct the transcriptome later, using the prefix graph flow as a proxy.

We close this subsection by noting that there are more efficient constructions of prefix graph (smaller graph for the same splice graph and $\mathcal{P}$) as described in Section S1.5. Under certain regularity constraints of $\mathcal{P}$, such construction can be proved to be optimal and identifiable for inference as long as the model is also identifiable with the path abundance.

2.4 Inference with Prefix Graph Flows

In this section, we infer optimal transcriptome abundance with incomplete reference assumption, by inference with prefix graph flows as variables. With no multimapped reads and a single gene, we can optimize $\sum_{f\in F}\log(c_{p(f)}/\sum_{p\in\mathcal{P}}(c_p\hat{l}_p))$, where $\{c_p\}$ is derived from a prefix graph flow and the flow values are the actual variables. This is a convex problem solvable with existing solvers.

With the presence of multimapped reads (to multiple locations or multiple paths), we can employ a standard EM approach, however optimizing the flow for the whole genome is harder. This is because we need to infer the flow for every prefix graph across the whole genome, and the instance becomes impractically large as we need to at least satisfy flow balance in every graph. Instead, we first calculate the allocated gene abundance $\sum_{p\in\mathcal{P}_g} c_p\hat{l}_p$ ($\mathcal{P}_g$ is the set of paths in gene g) for each gene, then optimize the flow for each gene individually. This results in the following localized EM:

$$\text{Global E-step (for all fragments } f\text{):}\quad z_{f,p} = c'_p D(f \mid p)/(\sum_{p'\in M(f)} c'_{p'} D(f \mid p'))$$

$$\text{Local M-step (for each gene } g\text{):}\quad c = \arg\max_c \sum_{p\in\mathcal{P}_g} (\sum_{f\in F} z'_{f,p}) \log c_p$$

$$\text{s.t.} \sum_{p\in\mathcal{P}_g} c_p\hat{l}_p = \sum_{p\in\mathcal{P}_g}\sum_{f\in F} z'_{f,p}/|F|$$

$M(f)$ is the set of paths f maps to, $D(f|p)$ is the fragment length probability if f maps to p, z is the allocation vector in EM, and z'/c' are values from previous iteration. For clarity, we also hide the flow constraints for $\{c_p\}$. Detailed derivations can be found in Section S1.6.

2.5 Subgraph Quantification

Even if one can reconstruct transcriptome from the prefix graph flows, there can be multiple reconstructions with different transcript abundances, corresponding to the fact that the model is non-identifiable with $\{c_i\}$ as parameters. We are interested in determining the lowest and highest abundance for any given transcript across all possible reconstructions from prefix graph flows.

As reconstructions from prefix graph flows are equivalent to flow decompositions, we can re-state the problem in graph theory language as subgraph quantification, defined as follows:

Definition 2 (Subgraph Quantification)**.** *Let G be a directed acyclic flow graph with flow balance, and $\mathcal{T} = \{T_i, c_i\}$ denote a flow decomposition of G with path T_i and weight c_i.*

The OR-QUANT *problem with subset of edges E' is to determine the range of $\sum_{i:|T_i\cap E'|>0} c_i$, that is, total weight of decomposed paths that intersect with E'.*

The AND-QUANT *problem with list of edge sets $\{E_k\}$ is to determine the range of $\sum_{i:|T_i\cap E_k|>0\forall k} c_i$, that is, total weight of decomposed paths that intersect with every E_k. $\{E_k\}$ needs to satisfy a well-ordering property: if a path visits $e_i \in E_i$ first and $e_j \in E'_j$ later, we require $i < j$.*

The problem of deriving the range of optimal transcript abundances reduces to an instance of AND-QUANT with G being the prefix graph flow and $\{E_i\} = p(T)$. We introduce OR-QUANT because it is a prerequisite for solving AND-QUANT in our framework, and potential interest of applying the framework to other problems. We now provide a quick sketch of our algorithms and defer the details and proofs to Section S1.7 and Section S1.8. Let $U = \text{MaxFlow}(G)$.

For OR-QUANT, the lower bound is obtained by removing E' from G, running a maxflow algorithm on the remaining graph G^- and returning $U - \text{MaxFlow}(G^-)$. The upper bound is obtained by changing endpoints of edges in E' to a new sink T', then running a maxflow from S to T'. For AND-QUANT, loosely speaking, we determine the set of edges that might be used for some path satisfying AND-QUANT condition, and claim if a path only uses these edges, it intersects each E_i and satisfies the condition. Denote this set of edges G_B, we return $U - \text{OR-QUANT}(G - G_B)$.

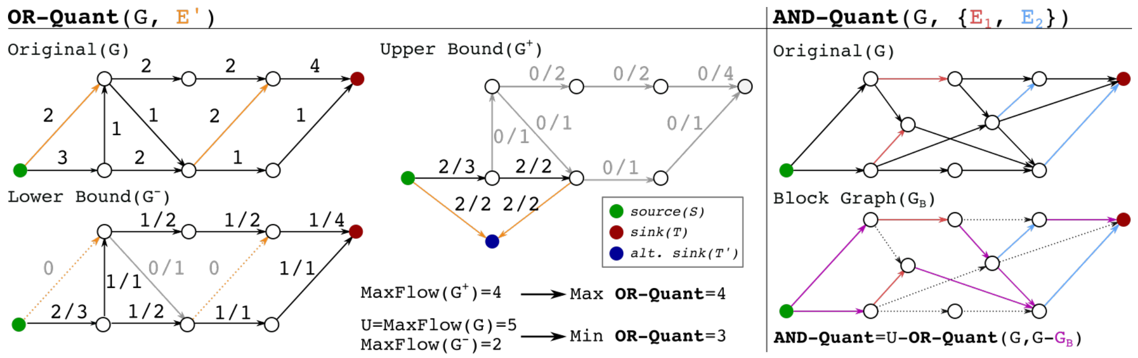

Figure 4: Algorithms for Subgraph Quantification (Section 2.5). Left side: Algorithm for OR-QUANT, showing G and the auxiliary graphs G^+ and G^- with maxflow. Right side: Algorithm for AND-QUANT, showing G and the block graph G_B. G_B consists of all colored / non-dashed edges in the image.

See Figure 4 for examples of both algorithms. Depending on the instance, there may exist other transcriptomes with the same set of path abundance but different prefix graph flow, meaning the prefix graph flow is non-identifiable in the model. We solve this concern in Section S1.9.

2.6 Structured Analysis of Differential Expression

We have discussed non-identifiability-aware transcript quantification under two assumptions (complete reference transcriptome and incomplete reference transcriptome as shown in Figure 2). In this section, we model the quantification problem under a hybrid assumption: Some reads are generated from the reference transcriptome while others are generated by combining known junctions. This model is more realistic than the model under the two extreme assumptions.

For each transcript T_i, let $[l_i^0, u_i^0]$ denote its expression calculated from the complete reference transcriptome assumption, and $[l_i^1, u_i^1]$ denote its expression calculated from incomplete reference transcriptome assumption. We use parameter λ to indicate the assumed portion of reads generated by combining known junctions. For $0 < \lambda < 1$, we define $[l_i^\lambda, u_i^\lambda] = [\lambda l_i^1 + (1-\lambda) l_i^0, \lambda u_i^1 + (1-\lambda) u_i^0]$, interpolating between the ranges for the extreme assumptions (We cover other possibilities in Section S1.10). The ranges can be useful for analyzing the effects of non-identifiability under milder assumptions, and we now show a usage of this framework in differential expression.

In differential expression analysis, for each transcript we receive two sets of abundance estimates $\{A_i\}$, $\{B_i\}$ under two conditions, and the aim is to determine if the transcript is expressed more in $\{A_i\}$. With fixed λ, we can instead calculate the ranges $\{[Al_i^\lambda, Au_i^\lambda]\}$ and $\{[Bl_i^\lambda, Bu_i^\lambda]\}$ as described above. If the mean of Al_i^λ is less than that of Bu_i^λ, we define this transcript to be a questionable call for differential expression. Intuitively, if λ portion of expression can be explained by unannotated transcripts, we cannot determine definitely if A_i is larger on average than B_i. This filtering of differential expression calls is very conservative (expect to filter out few calls), as most differential expression callers require much higher standards for a differential expression call.

3 Results

3.1 Point abundance estimation of Salmon tends to be near the boundaries of ranges of optima across 16 Human Body Map samples

Applying our method on Human Body Map RNA-seq samples, we characterize the distance of the single estimate of Salmon to the other values in the range of optima. The distance represents the potential quantification error due to the non-identifiability problem. The ranges of optima are calculated under both the complete reference assumption and incomplete reference assumption.

Salmon's estimates tend to be near either the smallest or the largest values of the range of optima rather than in the middle across all 16 Human Body Map samples (Figure S1A). The largest values in the range is on the same scale as and usually no greater than Salmon's estimate (Figure S1B). The patterns hold for both complete and incomplete reference assumption. A transcript determined to be lowly expressed under complete reference assumption may be expressed higher when unannotated transcripts are allowed to express. For these transcripts, Salmon's estimate is closer to the lower bound of the range of optima, as it is derived under complete reference assumption. Supplementary Figure S1 shows the subset of transcripts where the Salmon estimate is within the range of optima.

Under the incomplete reference assumption, Salmon's point estimation may be outside the range of optimal abundances (Supplementary Figure S2). This is because the optimal solution on a particular set of parameters (Salmon's estimates for reference transcripts) may not be optimal in an expanded parameter space (as we take novel transcripts into consideration).

3.2 Differentially expressed transcripts are generally reliable when assuming the reference contributes more than 40% to the expression

Applying our method on MCF10 cell line samples with and without EGF treatment (accession SRX1057418), we analyze the reliability of the detection of differentially expressed (DE) transcripts. The differential expression pipeline uses Salmon [4] for quantification, and then uses tximport [24] and DESeq2 [25] for detecting DE transcripts. This pipeline uses only Salmon's estimates and predicts 257 DE transcripts. We use the overlap between the mean range of optimal expression for the two conditions as an indicator for unreliable DE detection (as extended from Section 2.6). The overlap is defined as the size of the intersection over the size of the smaller interval of the two ranges of optima. The threshold of overlap to declare a unreliable DE detection is 25%.

We identify some examples of reliable and unreliable DE predictions. Transcript ENST00000010404.6 is considered to be an unreliable prediction by this criteria (Figure 5C). Assuming the reference transcriptome is complete, the expression of this transcript has a unique optimum. However, if there is a small amount of expression (10.9%) from the unannotated transcripts, the overlap exceeds 25%. Therefore, the accuracy of expression estimation and whether the transcript is differentially expressed is very vulnerable to unannotated transcript expression. This transcript belongs to gene *MGST1*, whose functions have not been clearly characterized but are inferred to involve mediating inflammation and protecting membranes [26].

Another example of unreliable DE detection is ENST00000396832.5, from gene *CSNK1E*. When there is a tiny amount (6.4%) of expression from unannotated transcripts, the ranges of optima across the two conditions overlap by more than 25%. Gene *CSNK1E* is a serine/threonine protein kinase [27], which is potentially stimulated by EGF [28]. However, the gene contains as many as 15 isoforms, and the effect of

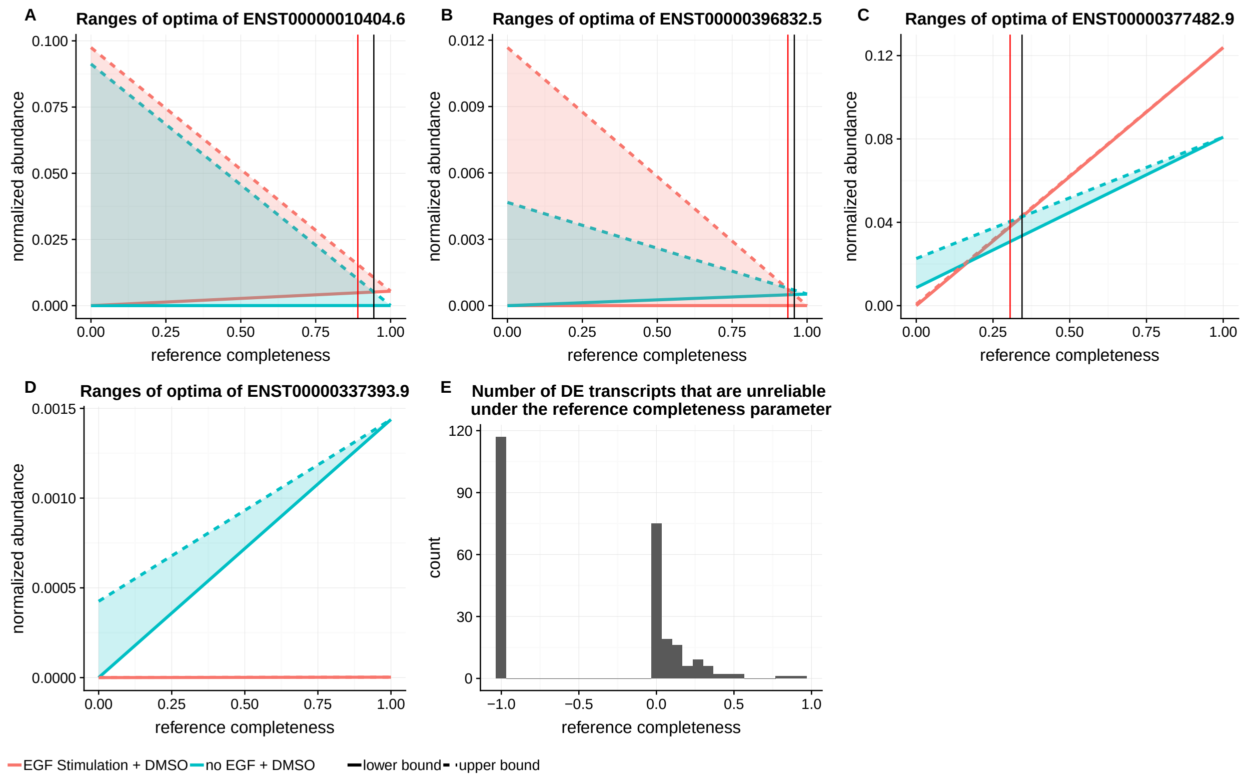

Figure 5: (A–D) Mean ranges of optimal abundances of DE groups. X axis is reference completeness parameter, which is the proportion of expression from the reference transcriptome. Y axis is the normalized abundances, where Salmon estimates are normalized into TPM for linear programming under complete reference assumption, and total flow in subgraph quantification is normalized to 10^6 for each sample. Black vertical lines indicate the reference completeness where the mean ranges of optima overlap. Red vertical lines indicate the reference completeness that the ranges have 25% overlap. (A) The potential unreliable DE transcript ENST00000010404.6. (B) The potential unreliable DE transcript ENST00000396832.5. (C) The reliable DE transcript ENST00000377482.9. (D) The reliable DE transcript ENST00000337393.9. (E) The histogram of the number of unreliable DE transcripts at each reference completeness parameter. Unreliability is defined as more than 25% overlap of the ranges of optima. -1.0 in the x axis indicates the overlap is no greater than 25% over all reference completeness parameter values.

EGF on each individual one may not be quantified accurately.

In contrast, transcripts ENST00000377482.9 from gene *ERRFI1* and transcript ENST00000337393.9 from *ZNF644* are highly reliable DE calls under our standard. The ranges of optima between DE conditions are different enough that the DE call is robust to a wide range of reference completeness parameter. Only when the reference transcriptome is very incomplete (only contributing less than 34.5% to the expression) do the ranges of optima of transcript ENST00000377482.9 overlap. Transcript ENST00000337393.9 from *ZNF644* does not have overlapping ranges of optima regardless of the reference completeness parameter. *ERRFI1* is known to be up-regulated in cell growth [29], which is a direct effect of EGF stimulation.

When more expression is from unannotated splice graph paths, the ranges of optima tend to be wider and more likely to overlap with each other. However, the majority of DE predictions are reliable when the reference transcripts are relatively complete and contribute more than 40% to the expression. A few DE predictions are unreliable and sensitive to a small proportion of expression from unannotated transcripts.

4 Discussion and Conclusion

In this work, we proposed approaches to compute the range of equally optimal transcript abundances in the presence of non-identifiability. The ranges of optima help evaluate the accuracy of the single solution from current quantifiers. Analyzing the reliability of differential expression detection using the ranges of optima on MCF10 cell lines samples, we observe that most predictions are reliable. Specifically, the ranges of optimal abundances between the case and control groups of the predicted transcripts do not overlap unless the expression of unannotated transcripts is greater than 60% of the whole expression. However, we still identify two unreliable predictions of which the ranges of optima between conditions largely overlap even when the reference transcriptome expression is assumed to be around 90% of the whole expression.

The reference completeness parameter λ is unknown in our model, and we address this by investigating the ranges of optima under varying λ. However, determining the best λ that fits the dataset as an indicator for reference trustfulness is an interesting question in itself, and we believe transcript assembly or related methods might be useful for choosing the correct λ value for each dataset.

The non-identifiability problem in expression quantification is partly due to the contrast between the complex splicing structure of the transcriptome and short length of observed fragments in RNA-seq. Recent developments of full-length transcript sequencing might be able to close this complexity gap by providing longer range phasing information. However, it suffers from problems such as low coverage and high error rate. It is still open whether this technology is appropriate for quantification under different assumptions and how the current expression quantification methods, including this work, should be adapted to work with full-length transcript sequencing data.

Acknowledgements

This research is funded in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4554 to C.K., by the US National Science Foundation (CCF-1256087, CCF-1319998) and by the US National Institutes of Health (R01GM122935). This work was partially funded by The Shurl and Kay Curci Foundation. This project is funded, in part, under a grant (#4100070287) with the Pennsylvania Department of Health. The Department specifically disclaims responsibility for any analyses, interpretations or conclusions. We would also like to thank Natalie Sauerwald and Dr. Jose Lugo-Martinez for insightful comments on the manuscript.

Disclosure Statement

C.K. is a co-founder of Ocean Genomics, Inc.

References

[1] Bo Li and Colin N Dewey. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics*, 12(1):323, 2011.

[2] James Hensman, Panagiotis Papastamoulis, Peter Glaus, Antti Honkela, and Magnus Rattray. Fast and accurate approximate inference of transcript expression from RNA-seq data. *Bioinformatics*, 31(24): 3881–3889, 2015.

[3] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, 34(5):525–527, 2016.

[4] Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 14(4):417–419, 2017.

[5] Sahar Al Seesi, Yvette Temate Tiagueu, Alexander Zelikovsky, and Ion I. Măndoiu. Bootstrap-based differential gene expression analysis for RNA-Seq data with and without replicates. *BMC Genomics*, 15(8):S2, Nov 2014.

[6] Christelle Robert and Mick Watson. Errors in RNA-Seq quantification affect genes of relevance to human disease. *Genome Biology*, 16(1):177, 2015.

[7] Charlotte Soneson, Michael I Love, Rob Patro, Shobbir Hussain, Dheeraj Malhotra, and Mark D Robinson. A junction coverage compatibility score to quantify the reliability of transcript abundance estimates and annotation catalogs. *Life Science Alliance*, 2(1), 2019. doi: 10.26508/lsa.201800175.

[8] Juliana Costa-Silva, Douglas Domingues, and Fabricio Martins Lopes. RNA-Seq differential expression analysis: An extended review and a software tool. *PLoS ONE*, 12(12):e0190152, 2017.

[9] Katherine A Hoadley, Christina Yau, Denise M Wolf, Andrew D Cherniack, David Tamborero, Sam Ng, Max DM Leiserson, Beifang Niu, Michael D McLellan, Vladislav Uzunangelov, et al. Multiplatform analysis of 12 cancer types reveals molecular classification within and across tissues of origin. *Cell*, 158(4):929–944, 2014.

[10] Ignasi Morán, İldem Akerman, Martijn van de Bunt, Ruiyu Xie, Marion Benazra, Takao Nammo, Luis Arnes, Nikolina Nakić, Javier García-Hurtado, Santiago Rodríguez-Seguí, et al. Human β cell transcriptome analysis uncovers lncRNAs that are tissue-specific, dynamically regulated, and abnormally expressed in type 2 diabetes. *Cell Metabolism*, 16(4):435–448, 2012.

[11] Harold Pimentel, Nicolas L Bray, Suzette Puente, Páll Melsted, and Lior Pachter. Differential analysis of RNA-seq incorporating quantification uncertainty. *Nature Methods*, 14(7):687, 2017.

[12] Anqi Zhu, Avi Srivastava, Joseph G Ibrahim, Rob Patro, and Michael I Love. Nonparametric expression analysis using inferential replicate counts. *Nucleic Acids Research*, 47(18):e105–e105, 08 2019.

[13] Vincent Lacroix, Michael Sammeth, Roderic Guigo, and Anne Bergeron. Exact transcriptome reconstruction from short sequence reads. In *International Workshop on Algorithms in Bioinformatics*, pages 50–63. Springer, 2008.

[14] David Hiller, Hui Jiang, Weihong Xu, and Wing Hung Wong. Identifiability of isoform deconvolution from junction arrays and RNA-Seq. *Bioinformatics*, 25(23):3056–3059, 2009.

[15] Adam Roberts and Lior Pachter. Streaming fragment assignment for real-time analysis of sequencing experiments. *Nature Methods*, 10(1):71, 2013.

[16] 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.

[17] Laura H LeGault and Colin N Dewey. Inference of alternative splicing from RNA-Seq data with probabilistic splice graphs. *Bioinformatics*, 29(18):2300–2310, 2013.

[18] Bo Wang, Elizabeth Tseng, Michael Regulski, Tyson A Clark, Ting Hon, Yinping Jiao, Zhenyuan Lu, Andrew Olson, Joshua C Stein, and Doreen Ware. Unveiling the complexity of the maize transcriptome by single-molecule long-read sequencing. *Nature Communications*, 7:11708, 2016.

[19] Richard I Kuo, Elizabeth Tseng, Lel Eory, Ian R Paton, Alan L Archibald, and David W Burt. Normalized long read RNA sequencing in chicken reveals transcriptome complexity similar to human. *BMC Genomics*, 18(1):323, 2017.

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

[21] Mihaela Pertea, Alaina Shumate, Geo Pertea, Ales Varabyou, Florian P Breitwieser, Yu-Chi Chang, Anil K Madugundu, Akhilesh Pandey, and Steven L Salzberg. CHESS: a new human gene catalog curated from thousands of large-scale RNA sequencing experiments reveals extensive transcriptional noise. *Genome Biology*, 19(1):208, 2018.

[22] Lior Pachter. Models for transcript quantification from RNA-Seq. *arXiv preprint arXiv:1104.3889*, 2011.

[23] Alfred V Aho and Margaret J Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, 1975.

[24] Charlotte Soneson, Michael I Love, and Mark D Robinson. Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences. *F1000Research*, 4, 2015.

[25] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12):550, 2014.

[26] Kim D Pruitt, Tatiana Tatusova, Garth R Brown, and Donna R Maglott. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. *Nucleic Acids Research*, 40(D1): D130–D135, 2011.

[27] Nuala A O'Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Research*, 44(D1):D733–D745, 2015.

[28] Natalie G Ahn and EG Krebs. Evidence for an epidermal growth factor-stimulated protein kinase cascade in Swiss 3T3 cells. Activation of serine peptide kinase activity by myelin basic protein kinases in vitro. *Journal of Biological Chemistry*, 265(20):11495–11501, 1990.

[29] Maresa Wick, Christiane Bürger, Martin Funk, and Rolf Müller. Identification of a novel mitogen-inducible gene (mig-6): regulation during G1 progression and differentiation. *Experimental Cell Research*, 219(2):527–535, 1995.

S1 Omitted Proofs and Discussion

S1.1 Defining the Effective Length of Transcripts

Recall in Section 2.2 we define the effective length for transcript T_i as $\hat{l}_i = \sum_{j=1}^{l_i} \sum_{k=j}^{l_i} D(k-j+1)$. In most works, $\hat{l}_i$ is instead defined as $l_i - \mu(T_i)$, the actual length of transcript l_i subtract the truncated mean of D, and the truncated mean is defined as $\mu(T_i) = (\sum_{j=1}^{l(i)} jD(j))/(\sum_{k=1}^{l(i)} D_k)$. The following lemma establishes the connection between the two definitions.

Lemma S1. $\hat{l}_i = \sum_{j=1}^{l_i} \sum_{k=j}^{l_i} D(k-j+1) = (\sum_{t=1}^{l(i)} D_t)(l_i + 1 - \mu(T_i))$.

Proof.

$$\begin{aligned}
\hat{l}_i &= \sum_{j=1}^{l_i} \sum_{k=j}^{l_i} D(k-j+1) \\
&= \sum_{t=1}^{l_i} D(t)(l_i + 1 - t) \\
&= (l_i + 1) \sum_{t=1}^{l_i} D(t) - \sum_{t=1}^{l_i} tD(t) \\
&= (\sum_{t=1}^{l_i} D(t))(l_i + 1 - \frac{\sum_{t=1}^{l_i} tD(t)}{\sum_{t=1}^{l_i} D(t)}) \\
&= (\sum_{t=1}^{l_i} D(t))(l_i + 1 - \mu(T_i))
\end{aligned}$$

This completes the proof. □

Ignoring the difference between l_i and $l_i + 1$, the two definitions differ by a multiplicative factor of $\sum_{t=1}^{l(i)} D(t)$. This factor is exactly 1 if the transcript is longer than any possible reads, and very close to 1 as long as the transcript is longer than most reads, which is indeed the case for most transcripts in an RNA-seq experiment. We choose the doubly-sum formula for $\hat{l}_i$ for its strong connection to a generative model. Additionally, it is also more natural to state $P(T_i) \propto c_i \hat{l}_i$ with $\hat{l}_i$ defined this way: c_i is the copy of molecules for transcript T_i, and assuming the probability of a observing a fragment f from transcript T_i is proportional to $c_i D(f \mid T_i)$, we can naturally get $P(T_i) \propto \sum_{f \in T_i} c_i D(f \mid T_i) = c_i \hat{l}_i$.

S1.2 Re-calculating Normalization Constant

In this section, we prove the following equation, which is necessary for the reparameterization process described in Section 2.2:

$$\begin{aligned}\sum_{T_i \in \mathcal{T}} \hat{l}_i c_i &= \sum_{T_i \in \mathcal{T}} \sum_{j=1}^{l_i} \sum_{k=j}^{l_i} D(k-j+1) c_i \\ &= \sum_{p \in \mathcal{P}} \sum_{i,j,k : p(T_i[j,k]) = p} D(k-j+1) c_i \\ &= \sum_{p \in \mathcal{P}} \Big(\sum_{j,k : \exists i, p(T_i[j,k]) = p} D(k-j+1) \Big) \Big(\sum_{i : p \subset p(T_i)} c_i \Big) \\ &= \sum_{p \in \mathcal{P}} \hat{l}_p c_p \end{aligned}$$

The third equation holds because the sum of $D(k-j)$ across any transcripts containing path p is the same, as shift in reference does not change $D(k-j)$.

S1.3 Trimming Set of Phasing Paths

Recall the likelihood function under our reparameterized model: $P(F \mid \mathcal{T}, c) \propto \prod_{f \in F} c_{p(f)} / (\sum_{p \in \mathcal{P}} c_p \hat{l}_p)$. Paths $p \in \mathcal{P}$ with no mapped reads do not contribute to $\prod_{f \in F} c_{p(f)}$, as there are no $f \in F$ such that $p(f) = p$. It does play a role in calculating the normalization constant $\sum_{p \in \mathcal{P}} c_p \hat{l}_p$. However, since we only remove paths with very low $\hat{l}_p$, the contribution of $c_p \hat{l}_p$ for this set is intuitively small. This removal thus causes small underestimation of the normalization constant, and in turn small overestimation of transcript abundances (and path abundances).

If there is a removed path with large c_p when optimized under the full model, it means in the inferred transcriptome there is a lot of reads mapped to path p, even though zero reads mapped in the sequencing library. This mostly happens if there is a dominant transcript with incredibly high abundance, and p is part of the transcript. For such things to happen, the transcript must have huge amount of reads mappable, and the fact that no a single read mapped to path p indicates $\hat{l}_p$ is incredibly small, or the modeling might be faulty.

This trimming is necessary as the fragment length distribution $D(l)$ usually has a long tail when inferred from experiments, due to smoothing and potential mapping errors, leading to a lot of extremely long paths that are near impossible to sample a read from. This trimming might have some effect on our analysis of non-identifiability later from a theoretical perspective, as intuitively by trimming down $\mathcal{P}$ we are increasing the co-dimension of $\{c_p\}$ in $\{c_i\}$ leading to wider ranges of optima. However, as we described above, it is near impossible to sample a read from such paths, unless we have infinite amount of data, which is not the case in practice. In other words, the trimming can be seen as a modification to the original likelihood model to account for the fact that we cannot generate infinitely large sequencing libraries.

S1.4 Path Dependent Bias Correction

Our proposed model is also capable of bias correction, but not to the full extent as done in Salmon. We define the affinity $A_p(j,k)$ as the unnormalized likelihood of generating a read pair mapped to path p from position

j to k in some coordinate. This is the analogue for $P(f_j \mid t_i)$ in Salmon model, excluding the sequence specific terms (which can also be integrated into our model directly, but they will not appear in effective length calculation). For non-bias-corrected model, we simply have $A_p(j,k) = D(k-j+1)$. The bias correction that can be applied directly are then those calculated from the genomic sequence in between the paired-end alignment, including certain motif-based correction and GC-content-based correction. To adapt the likelihood model with bias correction, the definition of transcript and path effective length is changed as follows:

$$\hat{l}_i = \sum_{j=1}^{l_i} \sum_{k=j}^{l_i} A_{p(T_i[j,k])}(j,k)$$

$$\hat{l}_p = \sum_{j=1}^{l_i} \sum_{k=j}^{l_i} A_p(j,k)\mathbf{1}(p(T_i[j,k]) = p), \forall p \in p(T_i)$$

Note that $\hat{l}_p$ is the same for any T_i and we assume the coordinate when calculating A_p coincides with the internal coordinate of T_i. The definition of path abundance remains unchanged, and the inference and subgraph quantification processes are the same.

Admittedly, there is no way to know the exact location of the read pair within the transcript after reparameterization with path abundance, so transcript-specific bias correction cannot be done in an exact manner. Nonetheless, since the splice graph is known in full, approximate bias correction is possible. In our experiments, let $A_i(j,k)$ denote the affinity value calculated from a full bias correction model, for the fragment generated from base j to k on transcript T_i. Let $\hat{r}_i$ be the reference abundance of transcript T_i. Then we let $A_p(j,k) = (\sum_{i:p\subset p(T_i)} \hat{r}_i A_i(j',k'))/(\sum_{i:p\subset p(T_i)} \hat{r}_i)$, where j' and k' are the coordinates of the path sequence in reference coordinate of T_i. If $A_i(j,k)$ is independent of transcript i but only dependent on the path p, we have $A_p(j,k) = A_i(j',k')$ for any transcript i containing path p. Otherwise, we average the affinity value calculated from known transcripts on this loci, weighted by their reference abundance. For simplicity, we use Salmon outputs as reference abundance, but other approaches, including iterative process of estimating $A_i(j,k)$ and c_p alternatively, are possible.

One limiting factor for bias correction is that calculating $\hat{l}_p$ can be expensive as we need to calculate this for every possible fragments. To speed up this process, we can use a simplified form for $A_p(j,k)$ so $\hat{l}_p$ has a closed-form solution (for example, do not allow bias correction when calculating effective length, similar to existing approaches to calculate effective length of transcripts), sample from possible $A_p(j,k)$ when the number of fragments from a particular path is large, or precompute the values for fixed genome and bias correction model. These approaches may result in slightly inaccurate path effective length, and it is still an open question how it affects downstream procedures.

S1.5 Compact Aho-Corasick Automaton and Compact Prefix Graph

In this section, we formally define the ingredients of prefix graph flows and prove several claims in the main text in a more rigorous manner based on a more compact construction.

Definition S3 (Aho-Corasick CFSA for phasing path matching)**.** *The Compact Aho-Corasick Finite-State Automaton (CFSA) is built with the following setup:*

- *The alphabet is the set of all vertices in splice graph.*

- *The patterns to match against is the union of (1) all paths of length 1, and (2) all path in $\mathcal{P}$ with last exon removed.*
- *The dictionary suffix link $d[s, a]$, where s is a state in the FSA and a is a character (vertex in splice graph in our case), is calculated if and only if there is an edge from last vertex in s to a.*
- *Additionally, each suffix link is labelled with sa, the path represented by s attached by a, which by our construction is still a valid path in the splice graph.*
- *A path p in $\mathcal{P}$ is said to be recognized at a suffix link with label s', if p is a suffix of s'.*

Intuitively, we wasted some space in the original prefix graph by only using vertex-level information when the edge-level information are also available. The "compact" in the name of this CFSA is for the fact that not all dictionary suffix links are calculated as in usual Aho-Corasick FSAs, and the fact that pattern recognition happens over transitions (suffix links), not states. We construct the prefix graph in the same way as in the main text, extracting states of the CFSA as vertices and dictionary suffix links as edges, excluding the initial state where no suffix is matched. We call this new graph a **compact prefix graph**. For the rest of this section, we use $\{c_s\}$ to denote a flow on the compact prefix graph.

Lemma S2 (Properties of Compact Prefix Graph)**.** *Given a compact prefix graph and a flow $\{c_s\}$ mapped from a transcriptome $\{T_i, c_i\}$:*

- *The compact prefix graph is a DAG.*
- *There is a one-to-one mapping between $S - T$ paths in splice graph and $[S] - [T]$ paths in compact prefix graph.*
- *The value $\{c_p\}$ can be derived from the compact prefix graph alone, and is preserved during mapping between compact prefix graph flows and transcriptomes (similar to the main text): $c_p = \sum_{i: p \subset p(T_i)} c_i$.*

Proof.

- We label each node in the prefix graph by the last vertex of their corresponding CFSA state. If an edge goes from u to v (two nodes in the compact prefix graph), there is an edge from the label of u to the label of v (two nodes in the splice graph). This means we can sort the nodes of the compact prefix graph by using the label and the topological order of the splice graph, breaking ties arbitrarily, and the result will be a valid topological order for the compact prefix graph.
- For a $[S] - [T]$ path of the compact prefix graph, we can generate the path on splice graph by writing down the label for each node (as defined above, last vertex in the representing string). The resulting path is valid on the splice graph. On the other hand, we can generate a path on the compact prefix graph by feeding the transcript to the CFSA and list the visited states in CFSA.
- We need to show that exactly one edge in the set $AS(p)$ is visited if a transcript T matches p. This is the transition right before the last exon in p is visited, so it happens exactly once as no exon can be visited twice. The rest of the proof is identical to the original prefix graph proof.

□

There are many other methods to achieve the same end result of representing $\{c_p\}$. For example, simply have a variable c_i for each possible transcript, representing the transcriptome in its original form, and calculate the path abundance by definition. However, the compact prefix graph formulation is the simplest possible in certain sense, as shown in the following theorem:

Theorem S2 (Compact Prefix Graph Flow can be Minimal Sufficient)**.** *With fixed splice graph and path set $\mathcal{P}$, if $\mathcal{P}$ satisfy the condition that for each $p \in \mathcal{P}$ any prefix of p is also in $\mathcal{P}$, there is a one-to-one mapping between feasible set of $\{c_p \mid p \in \mathcal{P}\}$ and feasible set of $\{c_s\}$ where s iterates over the set of edges of compact prefix graph.*

Proof. The mapping from a compact prefix graph flow to $\{c_p\}$ is direct and unique. For the other direction, assuming the theorem is false, there exists two sets of $\{c_s\}$, compact prefix graph flow with identical $\{c_p\}$. Since each c_p is sum of several flow values, the difference between the sets $\{\Delta c_s\}$ satisfies the following:

$$\sum_{s \in AS(p)} \Delta c_s = 0, \forall p \in \mathcal{P}$$

We now prove that $\Delta c_s = 0$ for all s, using induction on the size of s (length of the splice graph path it represents), which we denote k. The induction hypothesis for k is that $\Delta c_s = 0$ for all edge s with size no less than k. For the base case where $k > \max_s |s|$, there are no edges with size k or larger, so the hypothesis holds trivially. Assume this holds for $k+1$. For each edge s' with size k, denote the splice graph path it represents as p'. By the construction of CFSA, it is a prefix of some $p'' \in \mathcal{P}$, and by the setup of the theorem $p' \in \mathcal{P}$, which means there is a corresponding equation $\sum_{s \in AS(p')} \Delta c_s = 0$. The set $AS(p')$ contains the edge s' (by definition) and some edge whose represented path is longer than s', and for all edges in the latter category, the induction hypothesis says their $\Delta c_s = 0$. This means $\Delta c_{s'}$ for the edge we are looking at is also zero. □

S1.6 Localized Expectation-Maximization

In this section, we rewrite the optimization target in a slightly different form: we optimize $\sum_{f \in F} \log c_{p(f)}$ satisfying $\sum_{p \in \mathcal{P}} c_p \hat{l}_p = 1$. The resulting maximum likelihood will be the same in both formulations, but c_p might be scaled by a constant between the optimal solutions. We also exclusively use the compact prefix graph defined in Section S1.5, and call it prefix graph for brevity.

We first look at multimapped reads. Let $M(f)$ denote the set of paths that a read pair f can map to, $D(f \mid p)$ be the fragment length probability when read pair is mapped to path p, the objective function becomes:

$$\log P(F \mid T, c) = \sum_{f \in F} \log\Big(\sum_{p \in M(f)} c_p D(f \mid p)\Big)$$

and the constraints being the same as before. First, with a standard EM argument, we can define allocation variable z and perform the following EM step:

$$\text{E-step:} z_{f,p} = c'_p D(f \mid p) / \Big(\sum_{p' \in M(f)} c'_{p'} D(f \mid p')\Big)$$

$$\text{M-step:} c = \arg\max_c \sum_{p \in \mathcal{P}} \Big(\sum_{f \in F} z'_{f,p}\Big) \log c_p$$

c' and z' denote values from last iteration. In the M-step the variable c also needs to satisfy aforementioned prefix graph constraints, including $\sum_{p \in \mathcal{P}} c_p \hat{l}_p = 1$.

Next, we adapt our EM algorithm to optimize over multiple genes at once. Let g denote a gene and $\mathcal{P}_g$ denote the set of paths belonging to g. The target function during the M-step can be decomposed by genes, but all genes together have to satisfy the normalization constraint $\sum_{p\in\mathcal{P}} c_p\hat{l}_p = 1$. If the per-gene factor $c_g = \sum_{p\in\mathcal{P}_g} c_p\hat{l}_p$ can be known in advance, the optimization can be done on each gene. The following lemma shows this is indeed achievable, without knowing c_g:

Lemma S3. *For an optimization instance on gene g with per-gene factor c_g, let c_p^* be the optimal solution when c_g is set to 1. Then, $c_p = c_g \cdot c_p^*$ is optimal for original problem.*

Proof. We show that if $\{c_p\}$ is optimal for per-gene factor c_g, then $\{c_p/c_g\}$ is optimal for normalization coefficient 1. First, the flow balance constraint is scale-invariant so it holds for $\{c_p/c_g\}$. The normalization constraint also holds by definition. We now look at the objective function:

$$\begin{aligned}\sum_{p\in\mathcal{P}_g}\Big(\sum_{f\in F} z'_{f,p}\log c_p\Big) &= \sum_{p\in\mathcal{P}_g}\Big(\sum_{f\in F} z'_{f,p}(\log(c_p/c_g)+\log c_g)\Big)\\ &= \sum_{p\in\mathcal{P}_g}\Big(\sum_{f\in F} z'_{f,p}\log(c_p/c_g)\Big) + \text{Const}\end{aligned}$$

where the Const term is constant to c_p. The variable term is the target function with normalization coefficient 1 and c_p replaced with c_p/c_g, so maximization of the two sets of variables are equivalent. □

We now plug $c_p = c_g c_p^*$ back to the log-likelihood function and solve for c_g, which yields $c_g \propto \sum_{p\in\mathcal{P}_g}\sum_{f\in F} z'_{f,p}$ by taking derivative of Lagrangian while treating c_p^* as constant. This gives the localized EM process as follows:

$$\begin{aligned}\text{Global E-step:}\quad & z_{f,p} = c'_p D(f|p)/\Big(\sum_{p'\in M(f)} c'_{p'} D(f\mid p')\Big)\\ \text{Local M-step:}\quad & c = \arg\max_c \sum_{p\in\mathcal{P}_g}\Big(\sum_{f\in F} z'_{f,p}\Big)\log c_p\\ & \text{s.t.} \sum_{p\in\mathcal{P}_g} c_p\hat{l}_p = \sum_{p\in\mathcal{P}_g}\sum_{f\in F} z'_{f,p}/|F|, \forall g\end{aligned}$$

Note that sum of all $z'_{f,p}$ over the genome is exactly $|F|$. Again in the M-step, the variable satisfies the prefix graph constraint.

S1.7 Algorithms for OR-Quant

We start by defining the flow realizations, which are essentially path decompositions, and the partial versions of them. Recall $U = \textsc{MaxFlow}(G)$.

Definition S4 (Flow Realizations)**.** *A realization $R = \{(T_i, c_i)\}$ on a graph G is a set of $S-T$ paths T_i, each with an abundance c_i, satisfying $\sum_{i:e\in T_i} c_i = f_e$ for every edge e.*

A partial realization is a subset of a (full) realization that by definition satisfies $\sum_{i:e\in T_i} c_i \le f_e$ for every edge e. Flow of a partial realization is defined as $\sum c_i$.

As mentioned in main text, the following lemma is well known, and we present it here without proof.

Lemma S4 (Weighted Decomposition of Flow Graphs)**.** *A flow graph is a directed acyclic graph with edge weight that satisfies for each node except S and T, sum of its incoming edge weight equals sum of its* *outgoing edge weight. Every flow graph can be decomposed into finite number of $S - T$ paths $\{T_i\}$ with weight $\{c_i\}$ where $\sum_{i:e \in T_i} c_i = f_e$ holds for every edge e.*

With inferred flows, prefix graph G is a flow graph that can be decomposed into a finite number of weighted paths. OR-QUANT problem asks for the minimum and maximum flow of a partial realization of which the paths go through any edge in E'. We now state our algorithms and prove their correctness.

Theorem S3 (Diff-Flow)**.** *The auxiliary graph G^- is built with edge set $E - E'$.*

$Z = U - \text{MAXFLOW}(G^-)$ is tight lower bound for OR-QUANT *of E'.*

Proof. We prove the theorem in two parts: First, we show there is a realization r of G such that $Q(r) = U - \text{MAXFLOW}(G^-)$, then we show any realization satisfies $Q(r) \geq U - \text{MAXFLOW}(G^-)$.

The key observation is that all $S - T$ paths that do not include any edge in E' are represented as full paths in G^-. For the first part, fix a maximum flow of G^- as F^-, we first build a realization r^- from F^-, then build a realization r^+ from the graph $G - F^-$. We have $Q(r^-) = 0$, as no path in r^- contain any edge in E'. The flow of r^+ is exactly $Z = U - \text{MAXFLOW}(G^-)$, so $Q(r^+) \leq Z$ and equation holds if every path in r^+ intersect with E^-. If some path in r^+ does not intersect with E', F^- is not a maxflow of G^-, because we can add that path to F^- to improve the flow, leading to a contradiction. This means every path in r^+ intersects with E', so $q(r^+) = U - \text{MAXFLOW}(G^-)$ and a full realization $r = r^+ + r^-$ have the desired property.

Similarly for the second part, for any realization r we split it into two parts: r^+ containing set of paths that intersect with E', and r^- containing everything else. r^- is a flow on G^-, so its flow cannot exceed $\text{MAXFLOW}(G^-)$. This means $Q(r) = Q(r^+) = U - Q(r^-) \geq Q - \text{MAXFLOW}(G^-) = Z$. □

Theorem S4 (Split-Flow)**.** *The auxiliary graph G^* is built by adding a vertex T' in G and for each edge in E' changing its destination to T'. T' replaces T as the sink of flows.*

$Y = \text{MAXFLOW}(G^*)$ is tight upper bound for OR-QUANT *of E'.*

Proof. We prove the theorem in a similar style: First we show there is a realization r of G such that $Q(r) = \text{MAXFLOW}(G^*)$, then we show any realization satisfies $Q(r) \leq \text{MAXFLOW}(G^*)$.

For the first part, we first build realization r^* from a maximum flow of G^*. However, r^* is not a valid realization on G, because G^* and G have different sets of edges. Our goal is to get a realization r^+ of G that is a "reconstruction" of r^* on G.

For convenience, we define the following:

Path Mapping for G^*. A $S - T$ path on G is mapped to a $S - T'$ path on G^* by finding first edge of the path that is in E', change the destination of that edge to T', and discard the path after that edge (so the path ends at T'). The original path is guaranteed to intersect with E'.

Similarly, a $S - T'$ path on G^* is mapped to a path on G by moving the destination of last edge (that was T') back to its original node before the transformation. We assume a label is kept on each edge that ends at T', so multiple edges can exist between a node and T'. The resulting path is guaranteed to intersect with E', but is not a complete $S - T$ path.

We apply an induction argument, formally defined as follows:

Flow Reconstruction Instance. An instance for flow reconstruction has two inputs: (F, r^*). F is a flow on G (they share the same topology but not flow values), and r^* is a partial realization on F^*, where F^* is constructed in the same way as G^* by moving endpoints of edge in E' to a new node T'. The output of the instance is r^+, a partial realization of F, satisfying the condition that (1) each path in r^+ intersects with E', and (2) if we map each path in r^+ back to F^*, resulting realization r^{*+} has the same flow as r^*.

The size of a flow reconstruction instance is defined as the size of r^* (number of paths), not to be confused with the flow of the instance. Our goal is to solve the instance for any size. The base case is when the size is zero meaning r^* is empty, and an empty r^+ can be returned. Now assuming the instance can be solved for size less than k, we solve the instance for size k.

Induction Algorithm. Pick the path (P, c) in r^* whose second-to-last node (right before T') is the last among all paths in topological ordering of G. Denote the last edge of this path, mapped back to F, as (u, v). P without last edge is then a path from S to u.

We next generate an arbitrary realization of F and use it to construct the a partial realization that roughly matches (P, c). To do this, we first discard paths that do not contain edge (u, v) in the realization. Then, for every remaining path in the partial realization, we change its path before u to match P and scale the weight by c/w where w is the flow of edge (u, v) on F. Note that $c/w \leq 1$, because r^* is a partial realization on F^*, meaning the flow of P does not exceed the flow on the edge (u, T') that was mapped from (u, v), which has capacity exactly w.

Denote the scaled partial realization as r^0. r^0 has total flow c, is a valid partial realization of F, and every path in r^0 has P (mapping its last edge back to G as (u, v)) as prefix. We continue mapping the paths by recursively calling the procedure on (F', r'^*), where F' is F with flow of r^0 subtracted (as $c/w \leq 1$, F' is still a valid flow graph), and r'^* is r^* with (P, c) removed.

Proof of Induction Correctness. The new instance to be called is an instance with size $k - 1$, since one path in r^* is removed. We first prove the recursive call is valid, that is, r'^* is indeed a partial realization of F'^*.

If this call is invalid, it means some edge in F'^* has less flow than the cumulative flow from r'^*. Since r^* is a partial realization of F^* and r'^* is a subset of r^*, this can only happen to some edge that is in a path in r^0 (only for these edges F'^* have less flow than F^*). By construction of r^0, this edge can only be either some edge in P (including the edge (u, v)), or some edge whose starting point is later than v in topological order of G. For the former case, exactly c flow is subtracted on this edge from F to F', and it is the same from r^* to r'^* for the path is removed with weight c. For the latter case, note that such edge have nonnegative flow in F'^* but zero flow in r'^*, otherwise this means some path in r^* uses that edge, which contradicts with our selection criteria of (P, c).

Given the recursive call is valid, we can prove the correctness of the algorithm by showing that r^+ indeed satisfies the two output conditions. For (1), each path in r^+ contains some edge (u, v) as defined in the induction algorithm, which is in E. For (2), note that at each recursion call, the realization r^0 mapped to F^* matches the flow of one path (P, c) in r^*, so the condition is maintained in recursion. This finishes the correctness proof for the induction.

Completing the Proof. With the induction algorithm, we can reconstruct a partial realization r^+ of G from r^* that is realization of a maxflow on G^*, then similar to last proof, construct r^- from G minus the flow of r^+. By construction of r^+, we have $Q(r^+) = \textsc{MaxFlow}(G^*)$. If some path in r^- intersect with E', we can map that path to G^* and improve the maxflow that yields r^*, contradiction. This means $Q(r^-) = 0$ and $Q(r) = \textsc{MaxFlow}(G^*)$.

On the other hand, for a realization r of G, we can write $r = r^+ + r^-$ where r^+ contains transcripts that intersect with E', and r^- otherwise. We can map each path in r^+ to G^* and resulting realization r^* is a flow on G^*, so $Q(r)$ which is total flow of r^+ cannot exceed $\textsc{MaxFlow}(G^*)$. This completes the whole proof. □

S1.8 Algorithms for AND-Quant

We start by discarding any edges $e \in E_l$ such that there are no $S - T$ path t satisfying $e \in t$ and $|t \cap E_k| > 0, \forall k$, and assume the remaining edge set is not empty. By definition, removal of these edges will not change the answer to AND-Quant.

Definition S5 (Start/End-Sets and Natural Order)**.** *Define $U_i = \{u : (u, v) \in E_i\}$, $V_i = \{v : (u, v) \in E_i\}$ and for convenience, $V_0 = \{S\}, U_{m+1} = \{T\}$.*

Since G is a DAG, we also define the natural order $u \leq v$ if there is a directed path starting at u and ending at v. We define $x \to V_i$ as the condition that there is some $v_i \in V_i$ such that $x \leq v_i$ (intuitively, x can reach V_i). Same goes for U_i and the other direction.

Theorem S5 (Block-Flow)**.** *Define B_i, the i^{th} block subgraph, the set of edges (u, v) that satisfies $V_{j-1} \to u$ and $v \to U_j$, for $1 \leq i \leq m + 1$. The full block graph G_B is the union of all B_i and E_i.*

$\textsc{MaxFlow}(G_B)$ is the tight upper bound for AND-Quant of $\{E_k\}$.

The auxiliary graph G_B^+ is built by using the construction in Split-Flow, replacing E' by G_B. $\textsc{MaxFlow}(G_B^+)$ is the tight lower bound for AND-Quant of $\{E_k\}$.

The proof of Block-Flow theorem is essentially a complementarity argument, as shown in the following lemmas.

Lemma S5 (Well-Order Property)**.** *If $u \in U_i, v \in V_j, v \leq u$, then $i > j$.*

Proof. We can derive this from the well-order property of $\{E_k\}$. $v \leq u$ means there is a path that visits an edge in E_j then an edge in E_i, as there exists a path from S to an edge in E_j using v, and a path from an edge in E_i using u to T. □

Lemma S6. *For each $v_i \in V_i$, $v_i \to U_{i+1}$.*

Proof. Recall that we removed all edges in E_i that does not belong to any of paths that would satisfy the quantification criteria. $v_i \in V_i$ implies there is an edge $(u_i, v_i) \in E_i$ that is contained in such a path, so there is some edge (u_{i+1}, v_{i+1}) after it, which implies $v_i \to U_{i+1}$. □

As a corollary, this means if $u_i \in U_i$ then $u_i \to U_j$ where $j > i$, because there exists v_i such that $(u_i, v_i) \in E_i$ and $v_i \to U_j$. Same goes for V_i.

Lemma S7. *For a fixed i, each vertex x in G_B either satisfies $x \to U_i$ or $V_i \to x$, but not both.*

Proof. x is in G_B because there is a path that use x and satisfy the quantification criteria: This holds if x is in any E_i, and if x is in any of B_i, by definition of B_i there is a path from some v_{j-1} to u_i that includes

x, and we can extend both ends for a path that satisfies the criteria. The path uses some edge in E_i, and depending on whether x appears before E_i or after E_i it satisfies either $x \to U_i$ or $V_i \to x$.

No vertices can satisfy both, otherwise we have $u \in U_i, v \in V_i$ and $v \leq x \leq u$, which would imply $i < i$, contradiction. $\square$

Lemma S8 (Main Lemma for *AND-Quant*)**.** *An $S - T$ path in G satisfies the quantification criteria if and only if it is a subset of G_B.*

Proof. We first prove that if a path satisfies the quantification criteria, it is within G_B. An edge (u, v) on the path is either in some E_i, or between some edge in E_{i-1} and some edge in E_i; In latter case, we know $V_{i-1} \to u$ and $v \to U_i$, which directly imply $(u, v) \in B_i$.

To prove the other direction, we show that removing each of E_i results in disconnection of S to T. By the previous lemma, the vertices of G_B can be partitioned as: $\bar{S} = \{x : x \to U_i, x \in G_B\}, \bar{T} = \{x : V_i \to x, x \in G_B\}$. Consider an edge $(u, v) \in G_B$ that starts in $\bar{S}$ and ends in $\bar{T}$:

- If $(u, v) \in B_j$, we know $V_{j-1} \to u \to U_i$, so $j - 1 < i$, and at the same time $V_i \to v \to U_j$ so $i < j$, but there is not an integer i between $j - 1$ and j, contradiction.
- If $(u, v) \in E_j$, we know $U_j \to u \to U_i$ so $j \leq i$ (because $V_{j-1} \to u$), and $V_i \to v \to V_j$ so $i \leq j$ (because $v \to U_{j+1}$), which implies $i = j$.

So we have $(u, v) \in E_i$. In other words, removing all edge in E_i results in disconnection between $\bar{S}$ and $\bar{T}$, so each $S - T$ path in G_B uses an edge in E_i, for each i. This means the path satisfies the quantification criteria. $\square$

The last lemma means that all path that does NOT satisfy the AND-QUANT criteria intersects with $G - G_B$, meaning that we can complement the result of OR-QUANT on $E' = G - G_B$ to get the desired result. This completes the proof for the main theorem.

S1.9 Subgraph Quantification with Flow Reallocation

When we do subgraph quantification, we are finding path decompositions over prefix graph flows. As we proved before, if $\mathcal{P}$ satisfies a regularity constraint, the compact prefix graph (defined in Section S1.5) flows keep track of exactly the set of path abundance that is required for the likelihood. However, if this condition fails, the compact prefix graph flow is keeping more path abundance than necessary: these are the paths that are prefix of some other paths in $\mathcal{P}$, but not in the set itself. The fact that we can calculate c_p for such paths from the compact prefix graph alone indicates a narrower range in subgraph quantification, because we implicitly add extra constraints to the recovered transcriptome.

This constraint can be removed. For each of the subgraph quantification algorithms, we calculate a

MAXFLOW of f, a flow over some variant of the compact prefix graph, which can be written in an LP form:

$$\begin{aligned} \max & \sum_{e \in Out(S)} f'_e \\ \text{s.t.} & \sum_{e \in Out(v)} f'_e = \sum_{e \in In(v)} f'_e, \forall v \in V - \{S, T\} \\ & 0 \le f'_e \le f_e, \forall e \in E \end{aligned}$$

where f_e is determined from the compact prefix graph flow and the corresponding algorithm. However, we do not care about the exact compact prefix graph flow, as the path abundances $\{c_p\}$ determine the likelihood of the model. In a similar spirit with the reallocation process under complete reference assumption (Section 2.2), we propose to reallocate the flow of prefix graph as an LP. This results in the following LP, where s denotes an edge in compact prefix graph, $\{c^*_s\}$ is the original compact prefix graph flow modified by the problem of interest, and $\{c_s\}$ is the re-estimated flow:

$$\begin{aligned} \max & \sum_{s \in Out([S])} c_s \\ \text{s.t.} & \sum_{s \in Out(v)} c_s = \sum_{s \in In(v)} c_s, \forall v \in V - \{[S], [T]\} \\ & \sum_{s \in AS(p)} c_s = \sum_{s \in AS(p)} c^*_s, \forall p \in \mathcal{P} \\ & 0 \le c_s \le c^*_s, \forall s \end{aligned}$$

In experiments we only report the bounds without this flow-re-estimation step because the effect is small. The idea of re-expressing maxflows as LPs is very powerful and possibly enables other extensions of subgraph quantification, but at the cost of higher computational complexity.

S1.10 Flow Inference with Partial Reference Completeness Assumptions

By expressing l^λ as an affine combination of l^0 and l^1, we implicitly assume every read is split into a read with count λ that would be from any transcript with known junctions, and a read with count $1 - \lambda$ that would be from the reference transcriptome. Such modeling might not be ideal for some applications. A natural alternative is to assume that λ portion of all reads would come from any transcript with known junctions, while $1 - \lambda$ portion would be from the reference. In this section, we show how to extend our methods for this alternative assumption by an alternative inference procedure. Note that in our original modeling, we only need to run inference for $\lambda = 1$, so this method is inherently more costly as we need to run an inference instance for every λ in consideration. In addition to the flow values, for each reference transcript T_i, we add a variable c_i denoting its abundance, and change the formula of c_p as follows:

$$c_p = \sum_{s : s \in AS(p)} f(s) + \sum_{i : p \subset p(T_i)} c_i$$

. We change the normalization term (which was $\sum_{p \in \mathcal{P}} c_p \hat{l}_p = 1$) as follows:

$$\sum_{p \in \mathcal{P}} c_p \hat{l}_p = \lambda, \sum_{T_i \in \mathcal{T}} c_i \hat{l}_i = 1 - \lambda$$

. The resulting optimization program would quantify the transcripts assuming at least $1 - \lambda$ portion of reads come from known transcripts. For genome-wide quantification, the same local-global EM algorithm can be used assuming λ for each gene stay unchanged. For calculation of range of optima, we use the reallocation LP on $\{c_i\}$ and subgraph quantification on $\{c_p\}$, then add the results together. This method is still limited in the sense that we have to make the assumption on a per-gene basis, and cannot model the scenario where we assume λ portion of the whole read library come from novel transcripts (which requires a huge optimization instance as discussed in Section 2.4). In practice, we stick to the simple affine combination of l^0 and l^1 as it is the most efficient, however, the algorithm presented in this section might be interesting for future extensions of the model.

S2 Supplementary figures

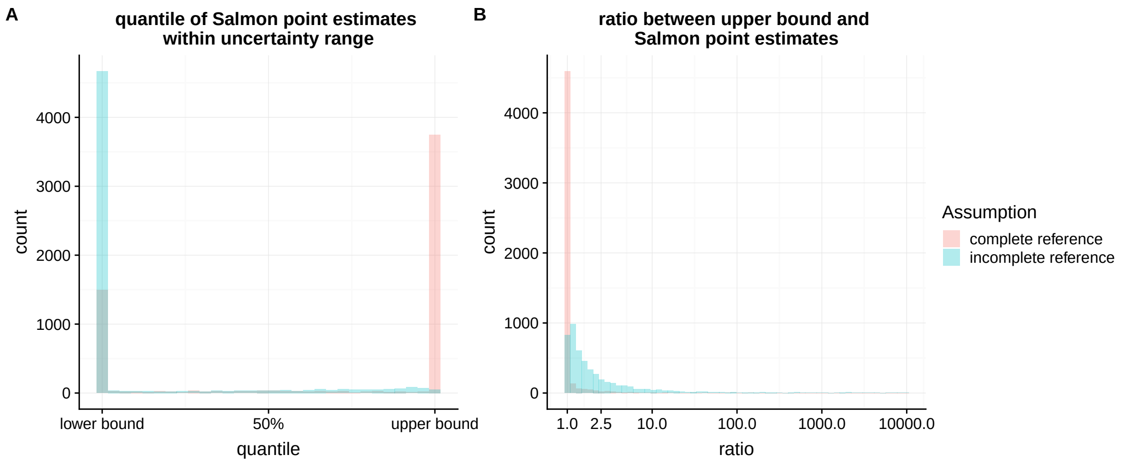

Figure S1: (A) Histogram of the position of Salmon estimates within the range of optima. X axis is the percentile between the lower bound and the upper bound of the range of optima. (B) Histogram of ratio between upper bounds and Salmon estimates.

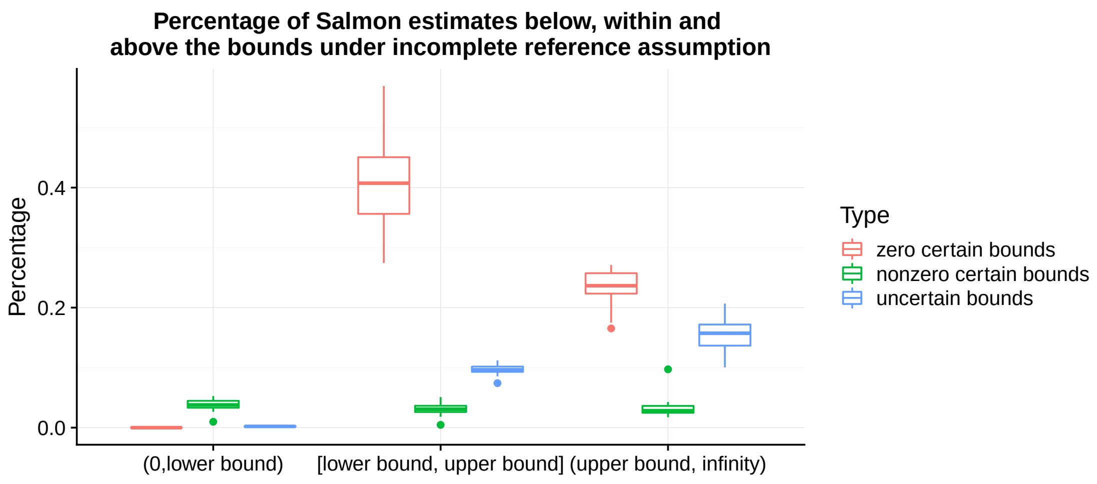

Figure S2: Percentage of Salmon estimates that are below the lower bounds, between the lower and upper bounds, and above the upper bounds of the uncertainty range under the assumption of freely expressed splice graph paths. That Salmon estimates are outside the uncertainty range can be explained by the difference between optimal objectives under different parameter spaces: the parameter space of Salmon is annotated transcripts and the one under the incomplete reference assumption is all paths in the splice graph.