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

February 15, 2020

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. Current quantification methods rely on probabilistic models, and the scenario where it admits multiple optimal solutions is called non-identifiability. This means multiple estimates of transcript abundances generate the observed RNA-seq reads equally likely, and the underlying true expression cannot be determined. The non-identifiability problem is further exacerbated when incompleteness of reference transcriptome and existence of unannotated transcripts are taken into consideration. State-of-the-art quantification methods usually output a single inferred set of abundances, and the accuracy of the single expression set is unknown compared to other equally optimal solutions. Obtaining the set of equally optimal solutions is necessary for evaluating and extending downstream analyses to take non-identifiability into account. We propose methods to compute the range of equally optimal estimates for the expression of each transcript, accounting for non-identifiability of the quantification model using several novel graph theoretical approaches. It works under two scenarios, one assuming the reference transcriptome is complete, another assuming incomplete reference and allowing for expression of unannotated transcripts. Our methods calculate a "confidence" range for each transcript, representing its possible abundance across equally optimal estimates. This range can be used for evaluating the reliability of detected differentially expressed (DE) transcripts, as a large overlap of confidence range between DE conditions indicates the prediction may be unreliable due to uncertainty. We observe that 5 out of 257 DE predictions are unreliable on an MCF10 cell line and 19 out of 3152 are unreliable on a CD8 T cell dataset. 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 RNA-seq expression quantification [1–4], as well as characterization and evaluation of the errors of

*Equal contribution

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

quantified expression [5–7]. However, there is an under-characterized type of estimation error which is due to the non-uniqueness of solutions to the probabilistic model. When the sequencing data can be explained by multiple sets of transcripts expression 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 intrinsic to the quantification model and almost inevitable. The non-uniqueness of solutions to an optimization model is known as the model non-identifiability under the assumption of infinite observations. Here we use a relaxed definition of non-identifiability to refer to the non-uniqueness of a model's solution under a finite set of observations. Whether the model is identifiable is usually determined by the form of the objective function and the dimension of the parameter space in the optimization problem. Figure 1 shows a toy example where the optimal abundances of the set of four transcripts are not unique. The interdependence between parameters is one of the causes of the non-identifiability phenomenon. Particularly for transcript expression quantification problems, the interdependence comes from exon sharing between transcripts.

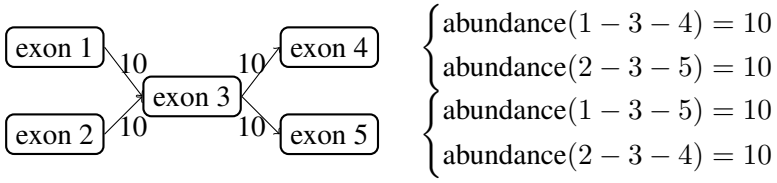

Figure 1: Example of a non-identifiable quantification model with four transcripts. Transcripts are the paths in the splice graph, denoted by the concatenation of exon indices. For example, $1-3-5$ denotes the transcript consisting of exon 1, 3, and 5. The number on each edge indicates the number of observed reads mapped to this splice junction. The set of transcript abundances are optimal if it perfectly explains the observed reads. That is, for each junction, the total abundances of transcripts containing that junction sum up to the number displayed on the edge. The right side of the figure shows two co-optimal expression abundances. In the first optimal solution, transcript $1-3-4$ and $2-3-5$ have an abundance of 10, and transcript $1-3-5$ and $2-3-4$ are not expressed. In the second optimal solution, transcript $1-3-5$ and $2-3-4$ have an abundance of 10, and transcript $1-3-4$ and $2-3-5$ are not expressed. It can be verified that both solution explain the observed reads perfectly as they all predict 10 reads on each junction.

Assessing the potential error due to non-identifiability helps increase the accuracy of analyses that are based on transcript expression data. Expression data is used in differential expression detection to study gene function, or treated as features to predict disease and treatment outcome [8, 9]. 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 [10, 11] 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.

Non-identifiability in transcript quantification has been discussed in previous works, but in-depth characterization is still lacking: no previous work lists out all values of expression estimates that are equally optimal. Lacroix et al. [12] and Hiller et al. [13] developed methods to list the transcripts that have non-unique expression abundances but their methods do not provide information about what values their optimal abundances can be. Roberts and Pachter [14] 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 a single expression estimate. Bernard et al. [15] proposed a network-flow-based model that has a similar

premise and approach to our work. Similar to our approach, their first step is to reparameterize using a network flow on a fragment graph. Bernard et al. [15] then quantified the transcripts with regularization followed by a flow decomposition, without taking identifiability of transcript expression into consideration. 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. Similarly, LeGault and Dewey [16] 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.

The reference transcriptome may not be complete and cells may express transcripts outside the reference. It is important to study the set of equally optimal estimates under the incomplete reference assumption, as studies of transcriptome assembly, third-generation sequencing with long reads and k-mer decomposition shows that the assumption likely hold for many datasets [17–22]. To accurately quantify the expression under the incomplete reference assumption, the model should take expression of novel transcripts into consideration. This greatly expands the dimension of the parameter space making the model more susceptible to non-identifiability, could lead to different quantification results, and is relatively unstudied. An independent line of research on reference-free transcript quantification (LeGault and Dewey [16] and Bernard et al. [15]) uses specialized splice graph-centric methods to quantify transcript abundances under this assumption. However, these methods deviate from the mainstream quantification models and leave the effect of non-identifiability unaddressed. We resolve both concerns with our purposed methods.

In this work, we propose methods to derive the confidence ranges for transcript abundances. Specifically, we derive a "range of optima" for each transcript, indicating its range of abundance in equally optimal estimates. The set of equally optimal estimates varies by assumptions of reference completeness, and we analyze three cases. (1) Assuming a complete reference, we propose a linear programming (LP) formulation to derive the ranges of optima for each transcript based on a phasing-aware reparameterization of the model (that is, the model properly takes phasing reads into account). (2) Assuming that the reference transcriptome is incomplete but the splice graph is correct, we propose a multi-stage approach: After the reparameterization, we reorganize the quantification problem using a variant of splice graph inspired by the multiple pattern matching problem, optimize using a specialized EM algorithm, and derive the ranges via efficient max-flow-based algorithms. (3) Under the assumption that the reference is incomplete but a known and given percentage of expression is contributed by the reference transcriptome, we derive the ranges based on the ranges under the two previous assumptions.

Applying our method on the probabilistic models used in the quantification software Salmon [4] on 16 Human Body Map samples, we are able to provide an interval of optimal abundances for each transcript. The ranges of optima can be used to evaluate the "confidence" of the expression estimates of Salmon. We observe that Salmon estimates tend to be near the lower bounds or upper bounds of the range of optima rather than in the middle. This may be related to the initialization or specific optimization method of convex optimization and deserves to be further studied from a theoretical perspective. The upper bounds of the range of optima are rarely above 10 times of Salmon's estimates, indicating that the under-estimation of expression abundance is not severe against non-identifiability.

Applying our method on sequencing datasets of a MCF10 cell line and of CD8 T cells, we use the ranges of optima to evaluate the differentially expressed transcripts detected using tximport [23] and DESeq2 [24]. Between the group of samples treated with and without epidermal growth factor (EGF) in the MCF10 dataset, 257 transcripts are detected as differentially expressed based on single estimates. The majority of the calls are reliable after considering the ranges of optima, assuming the reference transcriptome is somewhat complete and contributes to more than 40% of the expression. There are still a few unreliable

detections for which the two groups' ranges of optima overlap. The tolerance of reference incompleteness and the number of unreliable DE transcript predictions are the CD8 T cell dataset where we compare the differential transcript expression of naive CD8 T cells and effector memory CD8 T cells.

2 Method Overview

We provide a high-level walkthrough of our purposed methods in this section, and provide more details fore each component in Section 3.

Our goal is to investigate and quantify the effect of non-identifiability on transcript quantification. Doing this beyond qualitative statements (saying the expression of a transcript is non-identifiable under some model) requires a deeper understanding of the quantification model, especially the parameterization. Solving the problem also requires development of original frameworks and tools from graph theory and convex optimization, tailored to the quantification model. The end result of our process will be the **range of optima** $[l_i, h_i]$ for transcript T_i, indicating its inferred abundance across a set of equally optimal transcript abundances.

Our method depends on the following concepts and definitions. A **splice graph** is a directed acyclic graph representing alternative splicing events in a gene. The graph has two special vertices: S represents start of transcripts and T represents termination of transcripts. Every other vertex represent an exon or a partial exon. Edges in the splice graph represent **splice junctions**, potential adjacency between the exons in transcripts. A **path** is a list of vertices such that every adjacent pair is connected by an edge, and an $S-T$ path is a path that starts with S and ends with T. Each transcript corresponds to a unique $S-T$ path in the splice graph, and the set of known transcripts is called the **reference transcriptome**. $S-T$ paths that are not present in the reference transcriptome correspond to **unannotated transcripts**. We assume that a correct and complete splice graph is accessible, and use the phrase **quantified transcript set** to denote a set of transcripts with corresponding abundances.

2.1 Reparameterization

In general, non-identifiability indicates redundancy in the parameter space. Intuitively, if several sets of parameters lead to the same observations, there should be a parameter space with lower dimension that fits the data (in our case, the reads) equally well. This is indeed the case with our toy example present in Figure 1: assuming the four junction read observations can always be perfectly explained, one of them can be derived from the other three by the flow balance condition on exon 3 (total observed reads on incoming edges equal total observed reads on outgoing edges).

However, there are two important differences from our toy example and actual transcript quantification. First, modern quantification methods are based around generative models that are much more complex than the toy setup, and perfect fit is usually impossible. Second, phasing reads, that is, read pairs whose inferred fragments span multiple junctions exist in the input library. Splice graph paths that given set of fragments can map on are the **phasing paths**. We overcome these differences by first identifying a core model of transcript quantification, then showing that the core model can be reparameterized. The reparameterization uses the abundance over each phasing path as the parameters, called **path abundance** and denoted by $\{c_p\}$ where p is a phasing path. For genes with abundant alternative splicing events, the process indeed results in a lower dimensional representation.

With the reparameterization, we can concisely state the condition where the model is non-identifiable. If two quantified transcript sets yield the same set of path abundance c_p, they generate the same distribution of fragments under the generative model and fit the reads equally well. Given a reference quantification result, we can turn this observation into a linear program (LP) that yields the minimal and maximal abundance across all equivalent parameterizations, which is the range of optima under our first assumption that the reference is complete. See Section 3.1 for the mathematical details of the process.

2.2 Quantification with Incomplete Transcriptome

We next study the core quantification model where all $S - T$ path on the splice graph can be transcripts. Equivalently, the reference transcriptome is now expanded to include all previously unannotated transcripts.

The reparameterized model is uniquely fit for this study, as the main issue with just using the core quantification model is the prohibitive size of the expanded reference transcriptome. The reference transcriptome plays a central role in the original optimization problem as it determines the set of variables and appear in the objective function, but after the reparameterization, it only appears as constraints over path abundances.

A natural step now is to restructure the constraints to eliminate this single dependence on the reference transcriptome. The constraints were in place to ensure the inferred path abundances can be generated by a quantified transcript set. Re-implementing the same set of constraints using the expanded reference transcriptome still results in a large optimization instance, which does not help with our issues.

To motivate the next step, assume there are no phasing reads (every inferred fragment either resides within an exon or contains one junction). In this case, the phasing paths are nodes or edges in the splice graph. If the quantified transcript set is mapped onto the splice graph, we obtain a network flow. The path abundance for a phasing path equals either the flow through a vertex or an edge. By the flow decomposition theorem, given a network flow on the splice graph, we can decompose it into $S - T$ paths with weights, which then naturally maps back to a quantified transcript set. As the two-way mapping (between quantified transcript sets and splice graph flows) preserves path abundances, we conclude it is equivalent to optimize over a quantified transcript set and optimize over a splice graph flow. Specifically, it is easy to restructure the constraints to represent a splice graph flow, and the resulting model is guaranteed equivalent to the core quantification model with the expanded reference transcriptome.

We need to adapt the splice graph with the presence of phasing reads, as the mapping no longer preserves path abundances. We draw inspiration from pattern matching problems, and construct the Aho-Corasick automaton (a classical approach for multiple pattern matching) with the phasing path set. The automaton can be viewed as a directed acyclic graph with states as vertices and transitions as edges, which we call the **prefix graph**. We can show that the prefix graph, like the splice graph, has a two-way connection with a quantified transcript set: The set can be mapped onto a prefix graph flow, and decomposition of the flow translates to a quantified transcript set. Moreover, path abundances are preserved during the process, where each path abundance is now sum of flows through a set of edges in the prefix graph flow. By the same argument, we can now restructure the constraints with a prefix graph flow, and optimization over it leads to desired quantification results with the same guarantee.

Even if we have a tractable optimization instance now, efficient inference is still a problem, especially with the presence of multi-mapped reads and multiple genes. We handle both concerns with a modified EM algorithm, where the E-step and the M-step operate on different scales. We describe the process in more detail in Section 3.3.

2.3 Subgraph Quantification

To quantify transcripts under the incomplete reference assumption, after constructing and solving the optimization instance as described last section, we need to recover the transcript information from inferred prefix graph flows. This can be done by decomposing the flow, and the result corresponds to a quantified transcript set. However, as shown in our toy example, different flow decompositions may generate the same path abundance set, which is a direct manifestation of the non-identifiability problem. We now determine the minimal and maximal abundance of a particular transcript across all possible decompositions of the same flow, which is the range of optima under our second assumption that the reference is incomplete but the splice graph is correct.

We propose an abstraction of the problem in graph theory language. In our framework, a transcript becomes a list of edge sets from the prefix graph, each corresponding to a junction. An $S - T$ path maps to a given transcript and called a "good path", if it intersects each edge set in the list corresponding to the transcript (indicating the path includes each junction on the transcript). The problem now becomes finding the minimal and maximal total flow through good paths for any flow decomposition. By different choice of the edge set list, we can quantify different substructures other than full transcripts in a similarly non-identifiability-aware manner.

We call the resulting problem AND-QUANT, and solve it in two steps. The first step transforms the problem into an instance of OR-QUANT, where a path is good if it intersects with a given edge subset. We then solve OR-QUANT by two invocations of MAXFLOW. Both steps are as efficient as a MAXFLOW on the prefix graph. Details can be found in Section 3.4.

2.4 Hybrid Assumptions

To complete our analysis on effect of non-identifiability, we propose a hybrid model of read generation in Section 3.5, where our previous models can be seen as two extremes to the hybrid model (thus, the third model). We combine the results from the two models to derive the range of optima for each transcript as a hybrid model under our third assumption that there is a given percentage of contribution from the reference as opposed to the rest of splice graph paths. We demonstrate one use of our framework as reliability check for differentially expressed transcripts.

3 Methods

We now describe each component of proposed methods in greater detail. Due to limited space, most detailed descriptions and extended discussions are left to Supplementary Materials.

3.1 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 an RNA-seq experiment are error-free and uniquely aligned to a reference genome with possible gaps as fragments (Assumptions will be relaxed later). We denote the set of fragments (mapped from paired-end 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 $\{c_i\}$ normalized then multiplied by 10^6. Under the core model, the probability of observing F is:

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

$P(T_i)$ denotes the probability of sampling a fragment from transcript T_i, and $P(f \mid T_i)$ denotes the probability of sampling the fragment f given it comes from T_i. $A(f)$ is the set of transcript indices 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 S1.2 and refer to previous papers [1–4, 25] for more detailed explanation of the model. This leads to:

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

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. The splice graph is constructed so each transcript can be uniquely mapped to an $S-T$ path $p(T_i)$ on the graph, and we assume the read library satisfies that each fragment f can be uniquely mapped to a (non $S-T$) path $p(f)$ on the graph (this assumption will be relaxed later). 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 **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 S1.3). The likelihood can now be rewritten as follows:

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

This reparameterizes 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 (as each path can be shared by multiple transcripts), a direct manifestation of the non-identifiability problem. In practice, we reduce the size of $\mathcal{P}$ by discarding long paths with small $\hat{l}_p$ and no mapped fragments, as they contribute little to the likelihood (See Section S1.4). Our model is also capable of certain bias correction (See Section S1.5).

With the reparameterized model, we now derive the range of optima 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 [4] 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. This also holds with presence of read pairs mapped to multiple paths, as the probability of generating these read pairs are weighted sums of $\{c_p\}$. Note that identical path abundances is not a sufficient condition for non-identifiability, and co-optimal path abundances are outside the scope of discussion for this paper. 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. For each transcript T_i, we can calculate the range of c_i in this indistinguishable set by setting up two LP instances (called the **Reallocation LP**), one maximizing c_i and one minimizing c_i, within aforementioned system.

3.2 Representing Path Abundance with Prefix Graphs

Our next goal is to infer the optimal transcript abundance under the incomplete reference transcriptome assumption, without iterating over every possible $S-T$ path. This is necessary as the number of $S-T$ paths can be exponential to the splice graph size, but assuming most phasing paths contain few exons, the size of $\mathcal{P}$ grows polynomially instead.

As discussed in the overview, we solve this problem by producing a set of constraints over $\{c_p\}$ that is only dependent on the splice graph, and show optimization over the new model is equivalent to the core model assuming all $S-T$ paths can be transcripts. Equivalently, we want to find the set of constraints, such that: a set of path abundance satisfies the constraints if and only if there exists a corresponding quantified transcript set, with expression of novel transcripts allowed.

The most natural idea is to use the splice graph flow to represent $\{c_p\}$, as shown in the overview section. However, it no longer works when some phasing path p contains three or more exons. Informally, this can be solved by constructing higher-order splice graphs (as defined in [16] 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, [15] 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 discussed before, this assumption no longer holds when paired-end reads are taken into consideration. For the connection between our model and pattern matching problems, we explain our model starting from analysis of a seeming 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 (the set of exons) constitute the alphabet, $\mathcal{P}$ is the set of patterns (short strings to match against), and $\mathcal{T}$ is the set of texts (long strings to identify pattern in). In this setup, the Aho-Corasick algorithm [26] 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 given the current state and the upcoming character. We do not need the links for all characters (exons), as we know $T_i \in \mathcal{T}$ is an $S-T$ 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 now construct a directed graph from its states and transitions as described above. 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 some path 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 path 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'.

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 resembles 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 state (the first state after that is always $[S]$ as the first vertex in an $S-T$ path is 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.

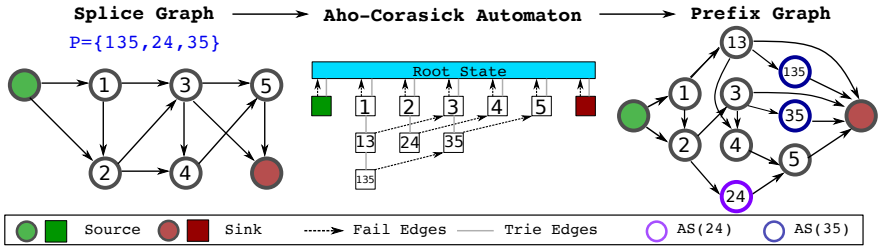

Figure 2: An example construction of the Prefix Graph. The source and sink of the prefix graph are $[S]$ and $[T]$, respectively. We draw the trie and the fail edges for the A-C automaton as it reduces cluttering (dictionary suffix link can be derived from both edge sets). The colored nodes in prefix graph are the vertices (states) in $AS(35)$ and $AS(24)$.

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

Theorem 1. *Every quantified transcript set 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 quantified transcript set onto the prefix graph as a prefix graph flow, and reconstruct a quantified transcript set 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 definition given a quantified transcript set, and $\{c'_p\}$ be the path abundance calculated from the prefix graph flow. We will show $\{c_p\} =$

$\{c'_p\}$ for any finite decomposition of the prefix graph flow.

For any transcript T_i and any path $p \in \mathcal{P}$, since no exon 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 abundance to c'_p and c_p. If T_i does not contain p, by similar reasoning, 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 quantified transcript set to prefix graph flow. Since the prefix graph flow is preserved in flow decomposition, the path abundance is preserved as a function of prefix graph flow. □

As discussed in overview, this connection allows us to directly optimize over $\{c_p\}$ by using the prefix graph flow as variables (c_p is now represented as seen in Theorem 1), then flow balance and non-negativity as constraints. The corresponding quantified transcript set is guaranteed to exist by a flow decomposition followed by the mapping process.

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 S1.6. 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.

3.3 Inference with Prefix Graph Flows

While a restructuring process described last section reduces the size of the optimization problem, we still need to solve it efficiently. With no multi-mapped reads (or fragments) and a single gene, we optimize $\sum_{f \in F} \log(c_{p(f)} / \sum_{p \in \mathcal{P}} (c_p \hat{l}_p))$, where the variables and the constraints come from the prefix graph flow, and c_p is represented as in Theorem 1. This is a convex problem solvable with existing solvers.

With the presence of multimapped reads (to multiple fragments and/or multiple paths), we can employ a standard EM approach. Optimization over the genome is harder, as we need to infer the prefix graph flow for every gene and the instance becomes impractically huge. Denote the gene abundance $\sum_{p \in \mathcal{P}_g} c_p \hat{l}_p$ ($\mathcal{P}_g$ is the set of paths in gene g). We decouple the genes during M-step by fixing the gene abundance, which makes the optimization over each gene independent of each other, while provably maintaining correctness of the process. This results in the following localized EM:

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

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

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

$M(f)$ is the set of paths f maps to, $D(f \mid p)$ 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 S1.7.

3.4 Subgraph Quantification

As shown before, given inferred prefix graph flows, we can obtain a quantified transcript set by flow decomposition over the flow. Different decompositions from the same flow are a direct manifestation of the non-identifiability phenomenon. For this section, we define the range of optima for each transcript as the minimum and maximum weight of the corresponding $S-T$ path across all valid decomposition of input flow.

We will implement two small changes to the prefix graph model as described before. These are necessary as we present a simplified model in the main text (the more compact prefix graph model is described in Section S1.6). First, $AS(p)$ is now an edge subset. A vertex subset correspond to an edge subset by taking the union of incoming edges to any vertex in the set. Second, we assume each junction is in $\mathcal{P}$.

Now each junction, or phasing path in general, is represented by a set of edges $AS(p)$ in the prefix graph, and an $S-T$ path as a transcript includes the junction/path if it intersects with $AS(p)$. This motivates the following formal definition of the subgraph quantification problem:

Definition 2 (AND-Quant)**.** *Let $G=(V,E)$ be a directed acyclic graph with an edge flow, and $\{E_k\}$ be a list of edge sets satisfying the well-ordering property: If a path visits $e_i \in E_i$, then $e_j \in E_j$ at a later step, $i<j$. An $S-T$ path is good if it intersects each E_k. For a flow decomposition of G, the total good flow is the total flow from good $S-T$ paths.* AND-QUANT$(G,\{E_k\})$ *asks for the minimum and maximum total good flow for all possible decompositions of G.*

For quantifying a full transcript, we can construct one $E_k = AS([u,v])$ for each of its junction (u,v) in order, and the range of optima for this transcript is exactly AND-QUANT$(G,\{E_k\})$. This formulation also allows us to quantify more general substructures. For example, we can use only a subsequence of $\{E_k\}$ and the AND-QUANT will return the minimum and maximum total abundance for transcripts that include corresponding set of junctions. We discuss this topic in more detail in Section S1.10.

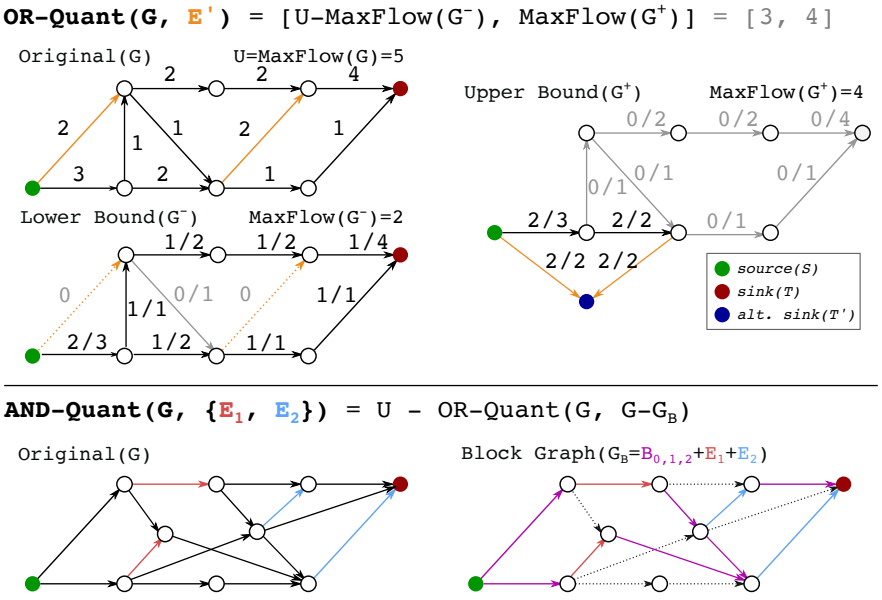

Figure 3: Algorithms for Subgraph Quantification (Section 2.5). Top: Algorithm for OR-QUANT, showing G and the auxiliary graphs G^+ and G^-. Bottom: Algorithm for AND-QUANT, showing G and the block graph G_B. As noted in the figure, G_B consists of all colored / non-dashed edges in the image.

To solve AND-QUANT, the first step is to define a similar problem called OR-QUANT as follows:

Definition 3 (OR-Quant)**.** *Let $G = (V, E)$ be a directed acyclic graph with an edge flow, and E' be an arbitrary subset of E. An $S - T$ path is good if it intersects E'. The total good flow and the objective are defined in the same way as in* AND-QUANT.

We now convert an instance of AND-QUANT to OR-QUANT by constructing a graph G_B, called the block graph, that contains every edge that is either in one of E_k, or is between an edge in E_{k-1} and an edge in E_k for some k (with some abuse of notation, assume the first edge set is $\{S\}$ and the last set is $\{T\}$). We claim the following for AND-QUANT:

Lemma 1. *An $S - T$ path is good if and only if it is a subset of G_B.*

We prove this in Section S1.9, which also provides a linear time algorithm for preprocessing and construction of G_B. If we know the minimum and maximum total "bad flow" (opposite of good flow), we can obtain the answer to AND-QUANT by complementing the result with U, the total flow of G. From the lemma, an $S - T$ path is bad if and only if it intersects with $G - G_B$, which turns the problem into an instance of OR-QUANT. To solve OR-QUANT(G, E'), we define two auxiliary graphs. G^- is a copy of G with all edges in E' removed. G^+ is a copy of G with an extra vertex T' which replaces T as the flow sink, and all edges in E' have their destination moved to T'. We claim running MAXFLOW on both graphs solves OR-QUANT:

Lemma 2. *Let G^+ and G^- be constructed as described above. Then* OR-QUANT$(G, E') = [U - \text{MAXFLOW}(G^-), \text{MAXFLOW}(G^+)]$.

We prove this in Section S1.8. Note that constructing G^+ and G^- can be done in linear time, so the time complexity of OR-QUANT depends on the MAXFLOW. Combining the arguments leads to the solution to AND-QUANT (recall U is the total flow on G):

Lemma 3. *Let G_B be the block graph, and $[l, r] =$* OR-QUANT$(G, G - G_B)$*. Then* AND-QUANT$(G, \{E_i\}) = [U - r, U - l]$.

As described before the preprocessing steps can be done in linear time, so the time complexity of AND-QUANT depends on the OR-QUANT, which in turn depends on the MAXFLOW.

See Figure 3 for examples of both algorithms. Throughout the section, we define the range of optima by assuming all equally optimal quantified transcript sets map to the same prefix graph flow. This can be overly restrictive, as other sets may have the same set of path abundance with different prefix graph flow (meaning the prefix graph flow itself is non-identifiable). We show that we can integrate the Reallocation LP into the framework to cover this scenario, as discussed in Section S1.10.

3.5 Structured Analysis of Differential Expression

We have discussed non-identifiability-aware transcript quantification under two assumptions. In this section, we model the quantification problem under a hybrid assumption: Some fragments 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 fragments 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 S1.11. 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.

4 Results

We focus on the quantification method Salmon [4] to calculate the range of optima. Salmon learns and outputs a refined bias model that takes into consideration the GC bias, sequence bias, and positional bias, which we use to calculate the path effective length in inferring graph flows. We retrieve the splice graphs from Gencode transcript annotation (version 26) [27] and assume that they encode all expressed transcripts. Even though splice graphs are constructed in many transcriptome assembly software in a sample-specific manner, we use the splice graph from the reference for all samples so that the ranges of optima across samples are directly comparable.

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

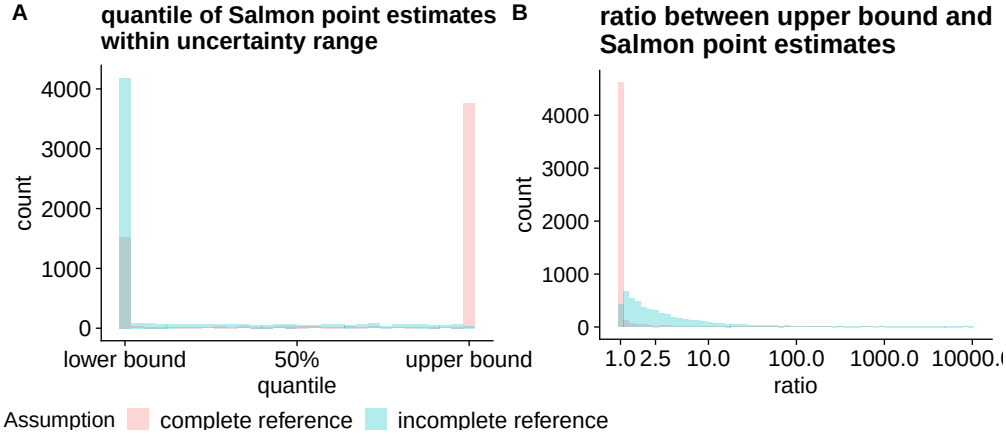

Figure 4: (A) Histogram of the position of Salmon estimates within the range of optima. X axis is the percentile between the lower bound and the upper bound of the range of optima. (B) Histogram of ratio between upper bounds and Salmon estimates.

Applying our method to the 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 non-identifiability. 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 4A). The cause of the behavior is currently unknown to us. But we suspect this phenomenon is related to the initialization and EM optimization of the convex objectives and it is of a theoretical interest to further study this phenomenon. The upper bounds of the optimal ranges are on the same scale as and usually no greater than 10 times of Salmon's estimate (Figure 4B). 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.

Note that Figure 4 only shows the subset of transcripts for which the Salmon estimate is in between the lower and upper bound of the range of optima under incomplete reference case. Under the incomplete reference assumption, Salmon's point estimation may be outside the range of optima (Supplementary Figure S1). 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).

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

Applying our method to the MCF10 cell line samples with and without EGF treatment (accession SRX1057418) [28], we analyze the reliability of the detection of differentially expression (DE) on the transcript level. The differential expression pipeline uses Salmon [4] for quantification, and then uses tximport [23] and DESeq2 [24] for detecting DE transcripts. This pipeline uses only Salmon's estimates and predicts 257 DE transcripts of which the adjusted p-value is no greater than 0.01. We use the overlap between the mean range of optima for the two conditions as an indicator for unreliable DE detection (as extended from Section 3.5). Since the range of optima depends on the percentage of expression from the reference according to the hybrid assumption, we compute the overlap for each percentage between 0 and 1 and refer to this percentage as the reference completeness parameter. 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 an unreliable DE detection is 25%.

We identify some examples of reliable and unreliable DE predictions. There are five differentially expressed transcripts for which their differential expression statuses are very sensitive to the expression of unannotated transcripts. Figure 5A–C shows the lower bounds and upper bounds of the transcript expression of three examples among the five transcripts. Even when reference completeness parameter values are larger than 90% (the unannotated transcripts have less than 10% expression), their expression estimates suffer from great uncertainty such that the ranges of optima between the two DE groups overlap. The five genes corresponding to the five transcripts are involved in the following cellular processes or pathways: mRNA degradation, cell apoptosis, glucose transportation, DNA repair, inhibition and transportation of certain kinetics [29]. These genes contain 6–22 isoforms. The sensitivity of these transcripts to unannotated transcript expression suggest that their differential expression status may be unreliable. Instead, combining all annotated and unannotated splice graph paths for these unreliable transcripts and testing the differential expression on the gene level may provide a more convincing conclusion.

Table 1: A list of five most unreliable DE transcripts in the MCF10 dataset and their corresponding gene-level DE result.

transcript id	adj-p (transcript)	gene symbol	adj-p (gene)
ENST00000509980.5	1.1×10^{-4}	*EXOSC9*	0.9998
ENST00000308394.8	2.2×10^{-7}	*CASP3*	0.9998
ENST00000443868.6	2.5×10^{-3}	*MRS2*	0.9998
ENST00000566749.5	2.3×10^{-3}	*PPP4C*	0.9998
ENST00000617373.4	1.9×10^{-5}	*SLC16A3*	0.9479

Table 1 shows the five transcripts and the adjusted p-values of differential expression on both transcript level and gene level. We use tximport to summarize the transcript-level quantification of Salmon to gene-level, and use the same procedure of DEseq2 for gene-level DE detection. Using a threshold of 0.01 on the adjusted p-value, we obtain 34 differentially expressed genes. For some differentially expressed transcripts, their gene-level DE tests are not significant, which indicates the occurrence of differential alternative splicing events. The gene-level DE test for the five most unreliable transcripts are not significant. For the five non-significant genes with significant but unreliable DE transcripts, it is very likely that neither differential gene/transcript expression nor differential alternative splicing occurs.

On the other hand, when both the gene and its corresponding transcripts are differentially expressed, the transcript is usually reliable against the non-identifiability and has high tolerance to the expression of unannotated transcripts. There are 25 DE transcripts of which the corresponding genes also pass the DE test on gene level. All of them are able to tolerate more than 50% of unannotated transcript expression, at which value the ranges of optima between the two DE groups still have zero or only a small overlap. Transcript ENST00000527470.5 is an example of them (Figure 5D). This transcript belongs to gene *NASP*, which is involved in histone transportation during cell division [29].

In general, the ranges of optima tend to be wider and more likely to overlap with each other when more expression is from unannotated transcripts. However, the majority of DE predictions are reliable when the reference transcripts are relatively complete and contribute more than 40% to the expression (Figure 5E). Meanwhile, there exists a few DE predictions that are unreliable and sensitive to small proportions of expression from unannotated transcripts.

On another dataset, we observe the same pattern of the reliability of transcript-level differential expression (Supplementary Figure S2). The dataset includes the RNA-seq of several immune cell types of peripheral blood mononuclear cells (accession GSE107011) [30, 31], and we compare the four replicates of naive CD8 T cells and four replicates of effector memory CD8 T cells for differential expression detection. There are 3152 differentially expressed transcripts with an adjusted p-value threshold of 0.01, out of which 19 are unreliable under less than 10% of unannotated transcript expression. The distribution of the reference completeness parameter values under which the DE transcripts are unreliable is similar to the one in the MCF10 cell line dataset. The Gencode reference transcriptome annotation is used for both datasets. Thus, the conclusion that most DE transcript prediction are reliable when the reference transcript expression is more than 40%, is potentially generalizable to more datasets under the same reference annotation.

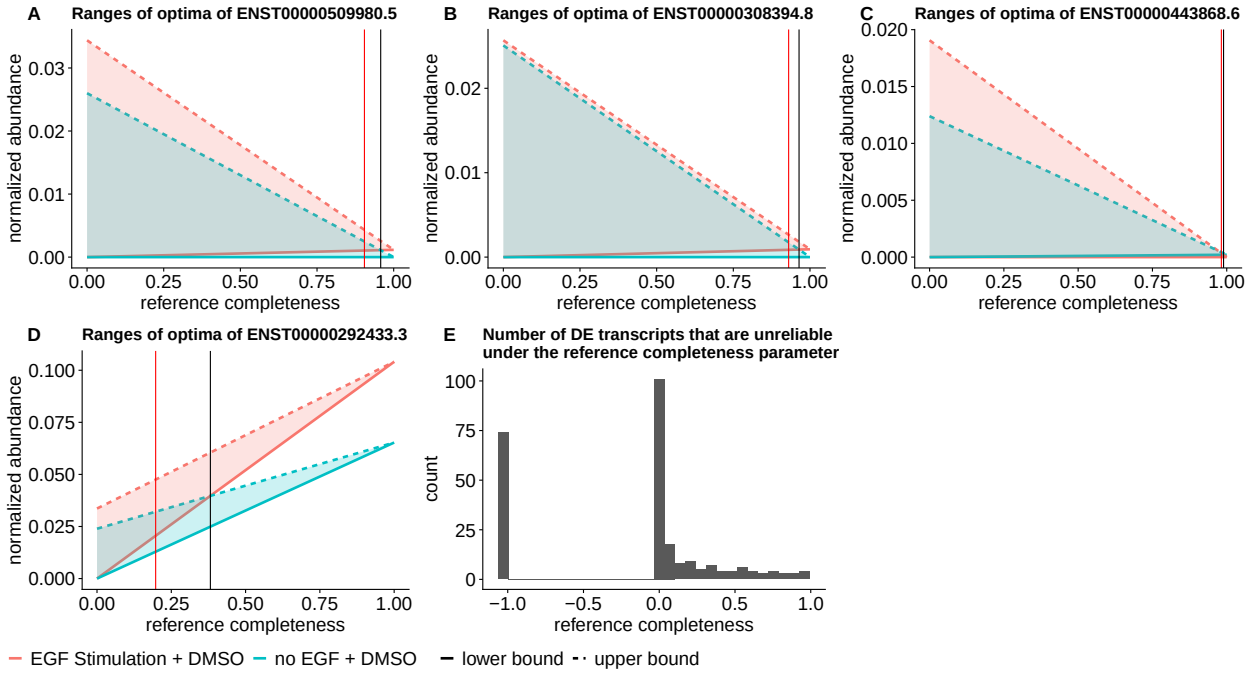

Figure 5: (A–D) Mean ranges of optima of DE groups. X axis is reference completeness parameter, which is the proportion of expression from the reference transcriptome. Y axis is the normalized abundances, where Salmon estimates are normalized into TPM for linear programming under complete reference assumption, 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 ENST00000509980.5. (B) The potential unreliable DE transcript ENST00000308394.8. (C) The reliable DE transcript ENST00000443868.6. (D) The reliable DE transcript ENST00000527470.5. (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.

5 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 five unreliable predictions for which the ranges of optima between conditions largely overlap even when the reference transcriptome expression contributes more than 90% of the whole expression.

The reference completeness parameter is unknown in our model, and we address this by investigating the ranges of optima under varying reference completeness values. However, determining the best reference completeness value 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 reference completeness 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, full-length transcript sequencing technique 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, Dr. Guillaume Marçais, Xiangrui Zeng 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] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] Jérôme Audoux, Nicolas Philippe, Rayan Chikhi, Mikaël Salson, Mélina Gallopin, Marc Gabriel, Jérémy Le Coz, Emilie Drouineau, Thérèse Commes, and Daniel Gautheret. DE-kupl: exhaustive capture of biological variation in RNA-seq data through k-mer decomposition. *Genome Biology*, 18 (1):243, 2017.

[22] Antonin Morillon and Daniel Gautheret. Bridging the gap between reference and real transcriptomes. *Genome Biology*, 20(1):112, 2019.

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

[24] 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.

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

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

[27] Adam Frankish, Mark Diekhans, Anne-Maud Ferreira, Rory Johnson, Irwin Jungreis, Jane Loveland, Jonathan M Mudge, Cristina Sisu, James Wright, Joel Armstrong, et al. GENCODE reference annotation for the human and mouse genomes. *Nucleic Acids Research*, 47(D1):D766–D773, 2018.

[28] Vladimir Yu Kiselev, Veronique Juvin, Mouhannad Malek, Nicholas Luscombe, Phillip Hawkins, Nicolas Le Novère, and Len Stephens. Perturbations of PIP3 signalling trigger a global remodelling of mRNA landscape and reveal a transcriptional feedback loop. *Nucleic Acids Research*, 43(20):9663–9679, 2015.

[29] 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.

[30] Weili Xu, Gianni Monaco, Eleanor Huijin Wong, Wilson Lek Wen Tan, Hassen Kared, Yannick Simoni, Shu Wen Tan, Wilson Zhi Yong How, Crystal Tze Ying Tan, Bernett Teck Kwong Lee, et al. Mapping of γ/δ T cells reveals Vδ2+ T cells resistance to senescence. *EBioMedicine*, 39:44–58, 2019.

[31] Gianni Monaco, Bernett Lee, Weili Xu, Seri Mustafah, You Yi Hwang, Christophe Carre, Nicolas Burdin, Lucian Visan, Michele Ceccarelli, Michael Poidinger, et al. RNA-Seq signatures normalized by mRNA abundance allow absolute deconvolution of human immune cell types. *Cell Reports*, 26(6): 1627–1640, 2019.

S1 Omitted Proofs and Discussion

We will cover more discussions and omitted proofs in the supplementary materials, some due to page limits and some due to being too technical, as indicated in various places in the main text. The individual sections, unless otherwise specified, are self-contained and can be read out of order. The first section below provides a brief overview over each section and how they support and extend the arguments in the main text.

Note that as we deal with network flows more directly in the supplementary sections, we will use f_e to denote the flow through edge e instead of a fragment.

S1.1 Overview

Section S1.2 to Section S1.5 completes and extends our discussion on the reparameterization procedure. One core ingredient of the process is the derivation of path effective length $\hat{l}_p$, only possible with a suitably defined transcript effective length, which we define in a different way compared to most other literatures. In Section S1.2, we discuss in detail our definition of transcript effective length and how it relates to the "mainstream" definition (in conclusion, not much difference assuming transcripts are longer than reads). In Section S1.3, we complete the derivation for the "effective length swapping" equation: $\sum_{T_i \in \mathcal{T}} \hat{l}_i c_i = \sum_{p \in \mathcal{P}} \hat{l}_p c_p$, which removes dependence of $\hat{l}_i$ from the model and is critical for the reparameterization. The rest two sections are about implementation of the reparameterized model. Section S1.4 discusses in detail the effect and necessity of culling $\mathcal{P}$. Section S1.5 describes how bias correction, an important aspect of modern transcript quantification tools, can be integrated into the reparameterized model, and the limitation of the procedure.

Section S1.6 describes the **compact prefix graph**. This is both theoretically and practically important for the whole framework, but it is harder to describe than the model we presented in the main text (which has a direct connection to the Aho-Corasick automatons). As suggested by the name, the compact prefix graph yields a smaller prefix graph when given the same set of input (splice graph and phasing path set), while keeping the same guarantees of preserving both transcripts and path abundances. This is the preferred model for implementing our methods, and we indeed use it in our own code. The theoretical importance of the model comes from the fact that under some conditions on $\mathcal{P}$, the compact prefix graph is the "minimal" representation of the path abundance, in the sense that any change in the compact prefix graph flow changes the path abundance. This compactness means that we provably recover every optimal quantified transcript set by decomposing the optimal compact prefix graph flow, which might not be true for the original prefix graph model as we discussed before.

Section S1.7 describes the inference process in detail, which we call Localized EM. The idea of the process has been laid out in the main text.

Section S1.8 to S1.9 proves correctness of the algorithms for AND-QUANT and OR-QUANT. For OR-QUANT, we prove the correctness of upper and lower bound separately, and the general idea is the same: Given the maxflow, we generate a decomposition that has the correct amount of good flow, and then we prove the total good flow for any decomposition is limited by the maxflow in certain ways. For AND-QUANT, we prove the correctness by exploiting the well-ordering process and stratifying vertices of the prefix graph into layers.

Secton S1.10 extends the subgraph quantification algorithms with the Reallocation LP. The main purpose for such process is to be able to consider other prefix graph flows that yield the same path abundance

as the current flow, closing a hole in the statement of the optimal solution set in consideration. This combination is more useful than closing the hole. The main text only introduces edge subsets corresponding to single exons, which is powerful enough to quantify over full length transcripts and partially specified transcripts. However, by "injecting" nonexistent phasing path into the prefix graph, we support much more flexibility in substructure design for the subgraph quantification process, while the Reallocation LP ensures the quantification is accurate in presence of the injected paths.

Finally, Section S1.11 introduces an alternative modeling under hybrid assumptions. Recall in the main text we produce the range of optima for each transcript by linear interpolation of two ranges of optimas from the two extreme assumptions. In fact, this corresponds to a rather naive model of read generation, which we will describe in detail in the supplementary section. We will then propose an alternative and more realistic model of hybrid read generation, at the cost of higher computational complexity.

S1.2 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 S4. $\hat{l}_i = \sum_{j=1}^{l_i} \sum_{k=j}^{l_i} D(k-j+1) = (\sum_{t=1}^{l(i)} D_t)(l_i + 1 - \mu(T_i))$.

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 $\sum_{t=1}^{l_i} D(t)$ (Intuitively, this is the probability of sampling a fragment no longer than l_i). This factor is exactly 1 if the transcript is longer than any possible fragments, and very close to 1 as long as the transcript is longer than most fragments, 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 $P(T_i) \propto \sum_{f \in T_i} c_i D(f \mid T_i) = c_i \hat{l}_i$.

S1.3 Re-calculating Normalization Constant

In this section, we prove the following equation, which is necessary for the reparameterization process described in Section 2.2. The idea is to break down the expression of $\hat{l}_i$ into sum over fragments, and regroup the fragments by the path they are mapped to.

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

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 $D(k-j)$ assuming no bias correction.

S1.4 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 fragments 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 quantified transcript set there is a lot of fragments mapped to path p, even though **exactly zero** fragments are mapped to p 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 fragments mappable, and the fact that no single fragment 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.

S1.5 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 $A_p(j,k) = D(k-j+1)$. Certain motif-based correction and GC-content-based correction, that are calculated from the genomic sequence in between the paired-end alignment, can then be integrated into our analysis naturally. To adapt the likelihood model with bias correction, the definition of transcript and path effective length is changed as follows:

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

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

Note that $\hat{l}_p$ is still 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. This means we average the affinity value calculated from known transcripts on this locus, 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.

S1.6 Compact Prefix Graph

While we introduced the prefix graph in the main text as the state transition graph of the Aho-Corasick automaton, it is not the best way to construct the graph. Intuitively, this is because the edge of the transition graph stores more information than the vertex (states) of the transition graph, and we only used the vertices to represent phasing paths before. In this section, we show how to use the transitions to represent phasing paths, which we call the **compact prefix graph**. The compact prefix graph is a smaller graph than the prefix

graph, meaning it is easier to perform flow inference and subgraph quantification. Moreover, the compact prefix graph is also guaranteed to be a minimal representation of the path abundances, as we will show in a theorem later this section.

We define the compact prefix graph still by the transition graph of an Aho-Corasick FSA, but with a different set of patterns.

Definition S4 (Aho-Corasick CFSA for phasing path matching)**.** *The Compact Aho-Corasick Finite-State Automaton (CFSA) is built as follows:*

- *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 tagged with $t(e) = sa$, 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 tag s', if p is a suffix of s'. $AS(p)$ is defined as the set of suffix links that recognizes p.*

Intuitively, the matching to phasing path process now happens over FSA transitions, instead of FSA states, which means the states does not need to contain full phasing paths. We construct the compact prefix graph in the same way as in the main text. The states of the CFSA are the vertices and the dictionary suffix links are the edges, with the initial state excluded as usual. The flow over the compact prefix graph, which we denote as $\{f_e\}$ for the rest of this section (NOT the fragments), has the same properties as the flow over the original prefix graph.

Lemma S5 (Properties of Compact Prefix Graph)**.** *Given a compact prefix graph and a flow $\{f_e\}$ mapped from a quantified transcript set $\{c_i\}$:*

- *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 quantified transcript sets: $c_p = \sum_{e \in AS(p)} f_e$.*

Proof.

- We label each node in the prefix graph by the last vertex of their corresponding CFSA state, which we denote $l(v)$ (this is different from the edge tags). If there is an edge (u, v) in the compact prefix graph, there is an edge $(l(u), l(v))$ in the splice graph. If the compact prefix graph has a cycle, the cycle can be mapped back to a cycle on the splice graph via the labels, leading to a contradiction as we assume the splice graph is a DAG.
- For a $[S] - [T]$ path of the compact prefix graph, the labels of every vertex on the prefix graph constitutes a path 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.

- We need to show that exactly one edge in the set $AS(p)$ is visited if a transcript T contains p. This is the transition right before the last exon in p is visited (the destination of this edge has the label that equals the last exon in p), so it happens exactly once as no exons can be visited twice. The rest of the proof is identical to the original prefix graph proof.

□

The compact prefix graph formulation is the most compact representation of path abundance in a certain sense.

Theorem S2 (Compact Prefix Graph Flow can be Minimal Sufficient)**.** *With fixed splice graph and phasing 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 path abundance $\{c_p \mid p \in \mathcal{P}\}$ and feasible set of compact prefix graph flows $\{f_e\}$.*

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 $\{f_e\}$, compact prefix graph flow with identical $\{c_p\}$. Since each c_p is sum of several flow values, the difference between the sets $\{\Delta f_e\}$ satisfies the following:

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

We now prove that $\Delta f_e = 0$ for all e, using induction on the size of $|t(e)|$ (the length of the tag on e), which we denote k. The induction hypothesis for k is that $\Delta f_e = 0$ for all edge e with tag no shorter than k. For the base case where $k > \max_e |t(e)|$, there are no edges with tag length k or longer, so the hypothesis holds trivially. Assume this holds for $k+1$. For each edge e' with tag length k, let $p' = t(e')$. By the construction process of the CFSA, p' is a prefix of some phasing paths in $\mathcal{P}$, and by our assumption p' is also in $\mathcal{P}$. This means there is an equation of form $\sum_{e \in AS(p')} \Delta f_e = 0$. The set $AS(p')$ contains the edge e' and several other edges with tag longer than p' (as p' is a suffix of these tags). By induction hypothesis, Δf_e for all edges in $AS(p')$, other than e', equals 0, which means $\Delta f_{e'}$ is also 0. □

S1.7 Localized Expectation-Maximization

Technically this section depends on the last section, as we only use the compact prefix graph (which we simply call the prefix graph) for actual derivations of the model. However, the only notable difference in this section is that $AS(p)$ is now a set of edges, instead of vertices. This applies to all later sections.

In this section, we provide full derivation and specification of the EM process designed for the reparameterized model. For clarity, we start with the base case, that is, when there are no multimapped read pairs and only one single gene. The full optimization problem is defined as follows (again, f_e is the flow through an edge of the compact prefix graph), where we let E be the edge set of the prefix graph and V be the vertex set of the prefix graph. In(v) and Out(v) denotes the incoming and outgoing edges of v.

$$
\begin{aligned}
\max & \sum_{f \in F} \log c_{p(f)} \\
\text{s.t.} & \sum_{p \in \mathcal{P}} c_p \hat{l}_p = 1 \\
& \sum_{e \in AS(p)} f_e = c_p && \forall p \in \mathcal{P} \\
& f_e \geq 0 && \forall e \in E \\
& \sum_{e \in \text{In}(v)} f_e = \sum_{e \in \text{Out}(v)} f_e && \forall v \in V - \{[S], [T]\}
\end{aligned}
$$

Note that we rewrite the optimization target in a slightly different form by restricting $\sum c_p \hat{l}_p = 1$ instead of normalizing c_p by this term in the objective function. The two formulations are equivalent to each other, except c_p might be scaled by a constant between the optimal solutions.

We first look at multimapped read pairs. 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:

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

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:

$$
\begin{aligned}
\text{E-step:} & z_{f,p} = c'_p D(f \mid p) / (\sum_{p' \in M(f)} c'_{p'} D(f \mid p')) \\
\text{M-step:} & c = \arg\max_c \sum_{p \in \mathcal{P}} (\sum_{f \in F} z'_{f,p}) \log c_p \\
& \text{s.t.} \sum_{p \in \mathcal{P}} c_p \hat{l}_p = 1
\end{aligned}
$$

c' and z' denote values from last iteration. We omitted the prefix graph flow constraints for c in the M-step for brevity.

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 gene abundance $c_g = \sum_{p \in \mathcal{P}_g} c_p \hat{l}_p$ can be known in advance, the optimization can be done on each gene (as the prefix graph flow constraints are all local to genes). The following lemma shows this is indeed achievable, without even knowing c_g:

Lemma S6. *For an optimization instance on gene g with gene abundance 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 gene abundance c_g, then $\{c_p / c_g\}$ is optimal for gene abundance of 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:

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

where the Const term is constant to c_p. Note that the variable term on the right hand side can be seen as the target function of the problem with $c_g = 1$ and c_p/c_g as variables. This means maximizing the two sides 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:

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

The gene abundances are normalized by $|F|$, because sum of all $z'_{f,p}$ over the whole library is exactly $|F|$ (it sums up to 1 for every fragment). Again, in the M-step, the variable satisfies the prefix graph constraint.

S1.8 Algorithms for OR-QUANT

In this section, we state the algorithms for OR-QUANT from a pure graph theoretical perspective. Recall our setup consists of a DAG G with predetermined source S and sink T, a flow $F = \{f_e\}$ on the graph (we decouple flows from the graph, and also these are not the fragments as in the main text), and an edge subset. We also use $C(F)$ to denote the total flow, and in this way we write $U = C(F)$.

We start by several basic definitions.

Definition S5 (Flow)**.** *Given DAG G with predetermined source S and sink T, a flow F is a set of edge weights of G satisfying non-negativity on every edge weight and flow balance on every vertex except S and T.*

Definition S6 (Decompositions)**.** *Given graph G and a flow $F = \{f_e\}$, a decomposition R is written as $\{T_i, c_i\}$ where each T_i is an $S-T$ path on G and c_i is a nonnegative number, satisfying $\sum_{i|e\in T_i} c_i = f_e$ for every edge e. We use $|R|$ to denote the number of $S-T$ paths in R, and $C(R) = \sum c_i$ to denote the total flow / capacity of the decomposition.*

Definition S7 (Partial Decompositions)**.** *A partial decomposition of a set of non-negative edge weights $W = \{w_e\}$ on a graph G is written as $\{T_i, c_i\}$ where each T_i is again an $S-T$ path on G, and c_i is a nonnegative number satisfying $\sum_{i|e\in T_i} c_i \le w_e$ for every edge e. Alternatively, if W is a flow, a partial decomposition can simply be defined as a subset of a (full) decomposition of F.*

Note that we define partial decompositions on arbitrary non-negative edge weights, as required for a later proof. As mentioned in main text, we present the following lemma without proof.

Lemma S7 (Finite Decomposition)**.** *Every flow on a graph of finite size has a decomposition of finite size.*

We now restate the definition of OR-QUANT slightly more formally.

Definition S8. *Let $G = (V, E)$ be a DAG with a flow F, and $E' \subseteq E$. An $S - T$ path is good if it intersects E'. (We also say the path is bad if it does not intersect E'.) For a decomposition $R = \{T_i, c_i\}$, the total good flow is defined as $Q(R) = \sum_{i:|T_i \cap E'|>0} c_i \leq C(R)$, that is total flow from good paths in the decomposition.* OR-QUANT(G, E') *asks for the minimum and maximum $Q(R)$ for any possible decomposition R of F.*

We now state our algorithms and prove the correctness of the algorithms, starting from the lower bound.

Theorem S3 (Diff-Flow)**.** *The auxiliary graph G^- is built with edge set $E - E'$. $Z = U -$* MAXFLOW(G^-) *is the minimum total good flow $Q(R)$ from any possible decomposition R of F.*

Proof. We prove the theorem in two parts: First, we show there is a decomposition r such that $Q(R) = Z$, then we show any decomposition satisfies $Q(R) \geq Z$.

The key observation is that all $S - T$ paths that are bad are fully in G^-. For the first part, fix a maximum flow of G^- as F^-. We let R^- be an arbitrary decomposition of F^-, and R^+ be an arbitrary decomposition of $F - F^-$.

We have $Q(R^-) = 0$ because no path in R^- intersects with E'. We now claim $Q(R^+) = Z$. Because $C(R^+) = C(R) - C(R^-) = Z$, $Q(R^+) = Z$ if and only if every path in R^+ intersects with E'. Assuming otherwise, we can remove that path from R^+ and add it to R^-, which leads to larger $C(R^-)$. This is impossible: it implies F^- is not the MAXFLOW of G^-, because the flow of R^- is a strictly better solution. Having proved $Q(R^+) = Z$, we now know $R^- + R^+$, a (full) decomposition, has exactly Z total good flow.

Now for any decomposition R, we split it into two parts. R^+ contains all good paths in R, and R^- contains all bad paths in R. We have $C(R^-) \leq$ MAXFLOW(G^-), because the flow of R^- is a flow on G^-. Because $Q(R^-) = 0$, we have $Q(R) = Q(R^+) = C(R^+) \geq Z$. □

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. $Y =$* MAXFLOW(G^*) *is maximum total good flow for any possible decomposition of F.*

Proof. We prove the theorem in a similar style: First we show there is a decomposition R of F such that $Q(R) = Y$, then we show any decomposition satisfies $Q(R) \leq Y$.

For the first part, we first build an arbitrary decomposition R^* from a maximum flow of G^*. However, R^* is not a valid partial decomposition of F, because G^* and G have different sets of edges. Our goal is to obtain a partial decomposition R^+ of F that is a "reconstruction" of R^*. We will define several terms for convenience of our proof.

Path Mapping for G^*. A good $S - T$ path on G is mapped to an $S - T'$ path on G^* by finding the first edge of the path that is in E', change the destination of that edge to T', and discard the edges after that so the path ends at T'. (We will never map a bad $S - T$ path on G this way.)

Similarly, an $S - T'$ path on G^* is mapped to a (incomplete) 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 containing its

original destination is kept on each edge that ends at T'. This implies multiple edges can exist between a node and T', and the movement follows the label. 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, and R^* is a partial decomposition on F^* (of G^*), where F^* is constructed in the same way as G^* by moving endpoints of edges in E' to a new node T'. Note that F^* is not a flow, while partial decomposition is still well-defined in this scenario. The output of the instance is R^+, a partial decomposition of F, satisfying that (1) Each path in R^+ is good and (2) If we map each path in R^+ back to G^*, the resulting partial decomposition has the same flow as R^*.

We will apply the induction on $|R^*|$, number of paths in R^*. The base case is then $|R^*| = 0$, where we can simply return an empty R^+. Assume now the flow reconstruction can be solved for $|R^*| \leq k - 1$, we solve the instance with $|R^*| = k$. We first state the algorithm, then prove its correctness.

Induction Algorithm. We first pick the path (P, c) in R^* whose second-to-last vertex (right before T') is the last among all paths in topological ordering of G. Denote the last edge of this path, mapped back to G, as (u, v). P without last edge is then a path from S to u. We let P' denote P with last edge changed to (u, v).

We next generate an arbitrary decomposition R^0 of F, then do the following: First, discard all paths not containing (u, v), which results in a partial decomposition of F. At this step, note that R^0 is a full decomposition so its total flow through (u, v) is exactly w. This means the total flow of the partial decomposition is exactly w. Second, as every path in R^0 now contains (u, v), we change the routing before u to match P. Finally, let w be the flow on the edge (u, v) in F, and recall c is the flow of the path P we picked from R^*. We scale the flow of each path in R^0 by c/w, making the total flow c.

Given R^* is a partial decomposition of F^*, the total flow of R^* on the edge (u, T') mapped from (u, v) does not exceed w. As (P, c) is a path in R^*, this further implies $c \leq w$ and $c/w \leq 1$.

We still need to show the resulting R^0 is a valid partial decomposition of F. Every path in R^0 is an $S - T$ path, and the every flow is still nonnegative, so we only need to prove that the total flow of R^0 on each edge does not exceed F. For edges that are not in P', the condition holds because the total flow on this edge can only decreased during all three steps (discarding paths and scaling down flows). For edges in P', the total flow on this edge in R^0 is exactly c. As (P, c) is a path in R^*, each of the edges in P has at least c flow in F^*, which immediately means each edge in P' has at least c flow in F.

To recap, R^0 has total flow c, is a valid partial decomposition of F, and every path in R^0 has P' as prefix. We continue the process by recursively calling the procedure on (F', R'^*), where F' is F minus the flow of R, and R'^* is R^* without (P, c). We return the partial decomposition from the recursive call merged with R^0.

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 that the recursive call is valid, that is, R'^* is indeed a partial decomposition of F'^* (F'^* is constructed from F' by moving the endpoint of edges in E' to T').

If the call is invalid, it means some edge in F'^* has less flow than total flow from R'^*. Since R^* is a partial decomposition of F^*, the condition will only be invalidated for an edge with positive flow in R^0, as these are the only edges where the flow on F'^* is less than the flow on F^*. This edge can either be an edge in P, or an edge whose starting point is later than v in the topological order of G by construction of R^0. If

the edge is in P and is not the last edge, exactly c flow is subtracted on this edge from F to F' and from R^* to R'^*, so this would contradict with the condition that R^* is a partial decomposition of F^*. If this is the edge (u, T') that is mapped from (u, v), again exactly c flow is subtracted in both F to F' and R^* to R'^*.

If this is an edge not in P, we claim the edge has zero flow in R'^*. This is because the way we pick (P, c), where the path with latest second-to-last vertex is picked. If the edge has positive flow in R'^*, it implies there is a path in R'^* that includes this edge, and the ending point of the path would be later than v, meaning the path would have been chosen in place of P during the process.

Given the recursive call is valid, we can prove the correctness of the algorithm by showing that R^+ outputted by the algorithm indeed satisfies the two output conditions. For (1), each path in R^+ is good because it come from a R^0 at some iteration, and all paths in R^0 has P' as prefix whose last edge (u, v) is in E'. For (2), at each recursion call, R^0 mapped to F^* becomes one path P with flow c, which matches (P, c) in R^*, so the condition is maintained similarly by an induction argument. This finishes the correctness proof for the induction.

Completing the Proof. With the induction algorithm, we can reconstruct a partial decomposition R^+ of G from R^* (a decomposition of F^*), then similar to last proof, construct another partial decomposition R^- from F 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 remove that path from R^- and add it to R^+, then map the new R^+ to G^* to obtain a flow better than $\textsc{MaxFlow}(G^*)$, a contradiction. This means $Q(R^-) = 0$ and the decomposition $R = R^+ + R^-$ satisfies $Q(R) = \textsc{MaxFlow}(G^*)$.

For the second part of the statement, for a decomposition R of F, we can write $R = R^+ + R^-$ where R^+ contains all good paths in R and R^- contains all bad paths in R. We can map R^+ to G^* and denote the resulting partial decomposition R^* (R^* is indeed a partial decomposition by noting there is a one-to-one mapping between edges of F^* and edges of F). We have $Q(R) = Q(R^+) = C(R^+) = C(R^*) \leq \textsc{MaxFlow}(G^*)$. This completes the whole proof. □

S1.9 Algorithms for AND-QUANT

In this section, we prove the complementarity argument mentioned in the main text is correct. Our setup consists of a DAG G with predetermined source S and sink T, a flow $F = \{f_e\}$ on G, and a list of edge sets $\{E_k\}$. We define flows and decompositions in the same way as last section (Definition S5, Definition S6). Now recall the definition of AND-QUANT, written slightly differently with the new notations:

Definition S9 (Well-Ordering Property)**.** *A list of edge sets $\{E_k\}$ satisfies the well-ordering property if a path visiting $e_i \in E_i$ then $e_j \in E_j$ at a later step implies $i < j$.*

Definition S10 (AND-Quant)**.** *Let $G = (V, E)$ be a directed acyclic graph with an edge flow, and $\{E_k\}$ be a list of edge sets satisfying the well-ordering property. An $S - T$ path is good if it intersects each E_k. For a decomposition of F, the total good flow is the total flow from good $S - T$ paths.* AND-QUANT$(G, \{E_k\})$ *asks for the minimum and maximum total good flow for all possible decompositions of F.*

We start by discarding edges e in any of E_i such that no good $S - T$ paths would use e. By definition, removal of these edges will not change the answer to AND-QUANT as no good $S - T$ paths would be excluded by removing these edges. This also means each edge e in any E_i satisfies that there is a good $S - T$ path that includes e.

Now given G is a DAG and the edge sets satisfy the well-ordering property, we can define the following:

Definition S11 (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,* $V_0 = \{S\}, U_{m+1} = \{T\}$.

We 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.*

Definition S12 (The Block Graph)**.** *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*.*

Before we continue, we will determine the time complexity for the filtering step (that discards any edge e in some E_i that no good paths would include) and the construction of G_B. We claim both processes can be done as quick as topological sorting of G (up to a constant multiplier), assuming the edge sets satisfy the well-ordering property, as follows.

We first discuss the filtering algorithm. We can first run two breadth-first search from S and T to mark the set of vertices that are reachable from S and can reach T. All vertices that are unable to do both will never appear in an $S-T$ path, and will be removed first. We let $s(v)$ denote the largest i such that there is a path from S, via an edge in each of $E_1, E_2, \cdots, E_i$ in order, to v. We now generate the topological order of G, and let $s(S) = 0$. For every vertex in the topological order other than S, the value of $s(v)$ is determined as follows. $s(v)$ is first set to be the largest of $s(u)$ from all its predecessors. Then for every $(u, v) \in E_j$ and $s(u) = j - 1$, we set $s(v) = \max(s(v), j)$.

We now show the values of $s(v)$ are correctly derived. If v is indeed reachable from S via an edge in each of $E_1, E_2, \cdots, E_i$, we have $s(v) \geq i$ as we will visit all nodes on the path in topological order and after an edge in E_i we always have $s(v) \geq i$. If $s(v) \geq i$, we show v is reachable from S via an edge in each of $E_1, E_2, \cdots, E_i$. We let $p(v)$ be the predecessor of v where $s(v)$ is calculated from, either by passing the value or passing an edge in E_j. By chaining $p(v)$, we obtain a path from S to v where each vertex is responsible for the value of $s(v)$ of its successor in the path. This means along the path the value will only increase by passing through an edge in E_j, and this can only bring $s(v)$ from $j - 1$ to j, so the path must contain an edge from each $E_1, E_2, \cdots, E_i$ in order.

Similarly, we can obtain $t(v)$ which is defined as the smallest i such that there is a path from v, via an edge in each of $E_i, E_{i+1}, \cdots, E_m$ to T by setting $t(T) = m+1$ and traverse the graph in reverse topological order. The filtering process then works as follows. For each $(u, v) \in E_i$, the edge is kept in E_i if and only if $s(u) = i - 1$ and $t(v) = i + 1$. We prove this procedure is correct. A good $S - T$ path containing (u, v) must visit an edge in each of $\{E_1, E_2, \cdots, E_{i-1}\}$ in order, visit $(u, v) \in E_i$, then an edge in each of $\{E_{i+1}, E_{i+2}, \cdots, E_m\}$. The first part is possible if and only if $s(u) \geq i - 1$, and the last part is possible if and only if $t(v) \leq i + 1$. It remains to prove that $s(u) \leq i - 1$ (the other part is symmetric). This is because otherwise there will be an edge in E_i that precedes u. As (u, v) is also in E_i, this means there will exist a path containing two edges in E_i, which violates the well-ordering property.

We next discuss the block generation algorithm. In fact, we can simply reuse the value of $s(v)$ and $t(v)$. For an edge (u, v) not in any of E_i, if $s(u) + 1 = t(v)$, the edge is allocated to $B_{t(v)}$. We prove this procedure is correct. If an edge satisfies $s(u) + 1 = t(v) = k$, there is an edge in E_{k-1} that leads to u, meaning $V_{k-1} \to u$, and similarly $u \to U_k$ which is the definition of B_k. Similarly, if $(u, v) \in B_k$, we show $s(u) + 1 = t(v) = k$. Given $V_{k-1} \to u$ and the edge set is filtered, there exists $v_{k-1} \in V_{k-1}, v_{k-1} \leq v$, and $s(v) \geq s(v_{k-1}) = k - 1$. If $s(v) \geq k$, this means there is an edge in E_k that leads to u and $V_k \to u$. Given $v \to U_k$ at the same time, we have $V_k \to u \leq v \to U_k$, and there is a path that visits an edge in E_k twice, violating the well-ordering property. We can similarly prove it is necessary and sufficient that $t(v) = k$.

Combining the results, we have the following lemma for preprocessing (note that topological sorting is a linear time algorithm):

Lemma S8. *There is an $O(n+m)$ algorithm that filters $\{E_i\}$ such that every remaining edge in every E_i is included in at least one good $S-T$ path, and constructs $\{B_i\}$ as described in Definition S12.*

We now return to the problem of answering AND-QUANT.

Lemma S9 (Well-Ordering Property Extended)**.** *If $u \in U_i, v \in V_j, v \leq u$, then $i > j$.*

Proof. We can derive this from the well-ordering property of $\{E_k\}$, by constructing an $S-T$ path. First, as $v \in V_j$ there is an edge in E_j that contains v, and by our previous assumption there is a path from S to v that uses this edge. As $v \leq u$, there is a path from v to u. As $u \in U_i$, there is an edge in E_i that contains u, and similarly there is a path from u to T that uses this edge. Combining the three segments together, we have an $S-T$ path that first visits an edge in E_j, then an edge in E_i later, so by the well-ordering property of $\{E_k\}$, $i > j$. □

Lemma S10. *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 good paths. $v_i \in V_i$ implies there is an edge $(u_i, v_i) \in E_i$ that is contained in a good path, and later in this good path there is an edge in E_{i+1} that starts at some vertices in U_{i+1}. This implies $v_i \to U_{i+1}$. □

Lemma S11. *For each $u_i \in U_i$, $u_i \to U_j$ where $j \geq i$.*

Proof. $u_i \in U_i$ means there is an edge $(u_i, v_i) \in E_i$, and $u_i \to U_i$. As shown last lemma, $v_i \to U_j$ for $j > i$. This means the same also holds for u_i as $u_i < v_i$. □

We can prove the symmetric statements about V_i.

Lemma S12. *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. If x is in one of U_j, $x \to U_i$ if $i \leq j$ by Lemma S11. Similarly, if $x \in U_j$, $V_i \to x$ if $i > j$ by both Lemma S10.

We can similarly prove that the condition holds for all $x \in V_j$, where $x \to U_i$ if $j < i$, and $V_i \to x$ if $j \geq i$.

If x is in none of U_j or V_j, it is in an edge in B_j, which by definition of B_j means both $V_{j-1} \to x$ and $x \to U_j$. Combined with Lemma S11, this means $x \to U_i$ if $j \leq i$, and $V_i \to x$ if $j > i$.

We have proved for any vertex in x and any i, one of $x \to U_i$ or $V_i \to x$ must be satisfied. 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 $i < i$ by Lemma S9, contradiction. □

Lemma S13 (Main Lemma for AND-QUANT)**.** *An $S-T$ path in G is good if and only if it is a subset of G_B.*

Proof. We first prove that if a path is good, 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 implies $(u, v) \in B_i$.

To prove the other direction, we show that removing any of E_i results in disconnection of S to T. By the previous lemma, the vertices of G_B can be partitioned into two disjoint sets: $\bar{S} = \{x : x \to U_i, x \in G_B\}, \bar{T} = \{x : V_i \to x, x \in G_B\}$. For 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 $V_{j-1} \to u \to U_i$ (first part from definition of B_j and second part from $u \in \bar{S}$, similar for the rest of this proof), 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 $V_{j-1} \to u$ by using the fact there is a good path including u), and $V_i \to v \to V_j$ so $i \leq j$ (because $v \leq v_j \to U_{j+1}$ for some $v_j \in V_j$ by Lemma S10), 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 is good. □

The main lemma means that all bad path 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. Combined with our analysis for OR-QUANT, we can solve AND-QUANT as described in main text, with time complexity equaling that of a MAXFLOW.

S1.10 Subgraph Quantification with Flow Reallocation

In this section, we will describe how to integrate subgraph quantification with the Reallocation LP, and how such combination enables more flexibility in the choice of substructures in the subgraph quantification step. Note that the prefix graph we describe here are the compact ones (Section S1.6) and some of the discussions in this section is dependent on the definition of the compact prefix graph. For the most part, the only visible difference is that in a compact prefix graph $AS(p)$ is a set of edges, instead of vertices.

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 S1.6) 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. For such path p, the path abundance is also preserved during the mapping from and to quantified transcript sets. This means when we perform subgraph quantification, by only using the compact prefix graph flow we also fix the path abundance of c_p. This leads to the scenario, where other compact prefix graph flows might yield the same set of path abundance for $p \in \mathcal{P}$ but not for these outsider path. Existing subgraph quantification procedures will never consider these flows, which unnecessarily narrows the range of optima.

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

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

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

where f_e is determined from the compact prefix graph flow and the corresponding algorithm, and we use f'_e to denote the MAXFLOW solution. 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 which we call **reallocated subgraph quantification**, where e denotes an edge in compact prefix graph, $\{f^*_e\}$ is the re-allocated flow from $\{f_e\}$, and $\{f'_e\}$ is the MAXFLOW solution.

$$\begin{aligned} \max \quad & \sum_{e \in Out([S])} f'_e & \\ \text{s.t.} \quad & \sum_{e \in \text{Out}(v)} f'_e = \sum_{e \in \text{In}(v)} f'_e & \forall v \in V - \{[S], [T]\} \\ & \sum_{e \in AS(p)} f_e = \sum_{e \in AS(p)} f^*_e & \forall p \in \mathcal{P} \\ & 0 \leq f'_e \leq f^*_e & \forall e \in E \end{aligned}$$

Note that f_e only appeared once in the new LP, and we can get rid of explicit dependence of f_e by using c_p on the left-hand side, thanks to the Reallocation LP. 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. For the rest of this section, we discuss how to quantify various substructures of the splice graph with the Reallocation LP.

Reallocation LP allows us to expand the compact prefix graph arbitrarily. The first step is modify the splice graph: add an extra pair of vertices S^* and T^* and an extra pair of edges (S^*, S) and (T, T^*). S^* and T^* are now the source and sink of the splice graph. This step is necessary as we will allow $\mathcal{P}$ to contain S or T (that no longer strictly corresponds to a phasing path).

A simple example is quantifying an arbitrary set of transcripts together using an OR-QUANT. This was impossible without the Reallocation process, because the variable c_p where p corresponds to the whole $S-T$ path does not exist and the path abundance cannot be determined correctly from the prefix graph flow. However, we can now add all transcripts, as $S - T$ paths (note now the source and sink are S^* and T^*), to the set of phasing path $\mathcal{P}$ and construct the new compact prefix graph. We can now quantify the transcripts as OR-QUANT, by setting E' to be the union of $AS(T_i)$ for each transcript T_i and using the LP without dependence on f_e.

Adding a whole path to $\mathcal{P}$ is not free as it increases the size of the resulting compact prefix graph. The number of vertices will increase by at most the length of the transcript (measured in exons), and the number of edges is upper bounded by the sum of out degrees for every exon in transcript. This means it is feasible to perform the quantification on a small set of transcripts, but not a set whose size is comparable to the set of all $S - T$ paths.

Another use of the new LP is to sidestep the well-ordering constraint in AND-QUANT to an extent. Assume our goal is to quantify the set of transcripts that use either exon 1 or 2 as one of the conditions (so it appears as one of the edge sets E_i), but there is an path from 1 to 2 (if there aren't, we can simply merge $AS([1])$ and $AS([2])$ as E_i, which satisfies the well-ordering property). We will assume the path consists a simple edge, that is, there is an edge from 1 to 2. In this case, we add the following set to $\mathcal{P}$: $\{[1, v] \mid (1, v) \in E\} + \{[s, 2] \mid (s, 2) \in E\}$. That is, all length-two paths that starts with 1 or ends with 2. The union of $AS(p)$ for this new set now as one E_i satisfies the well-ordering property as none of the edges would precede another.

S1.11 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 fragments 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

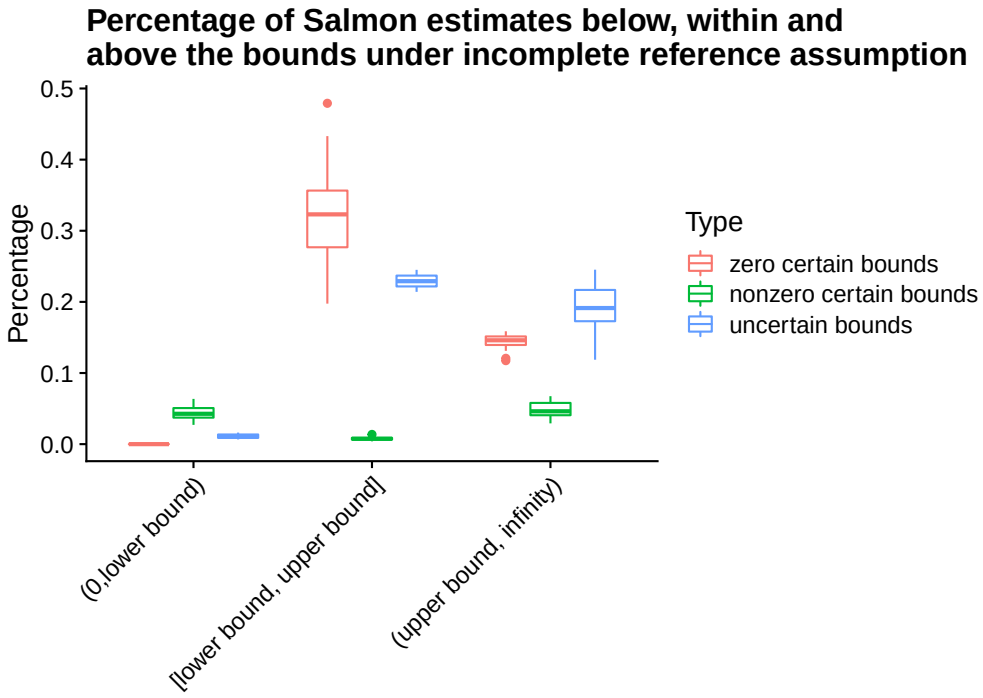

Figure S1: Percentage of Salmon estimates that are below the lower bounds, between the lower and upper bounds, and above the upper bounds of the uncertainty range under the assumption of freely expressed splice graph paths. That Salmon estimates are outside the uncertainty range can be explained by the 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.

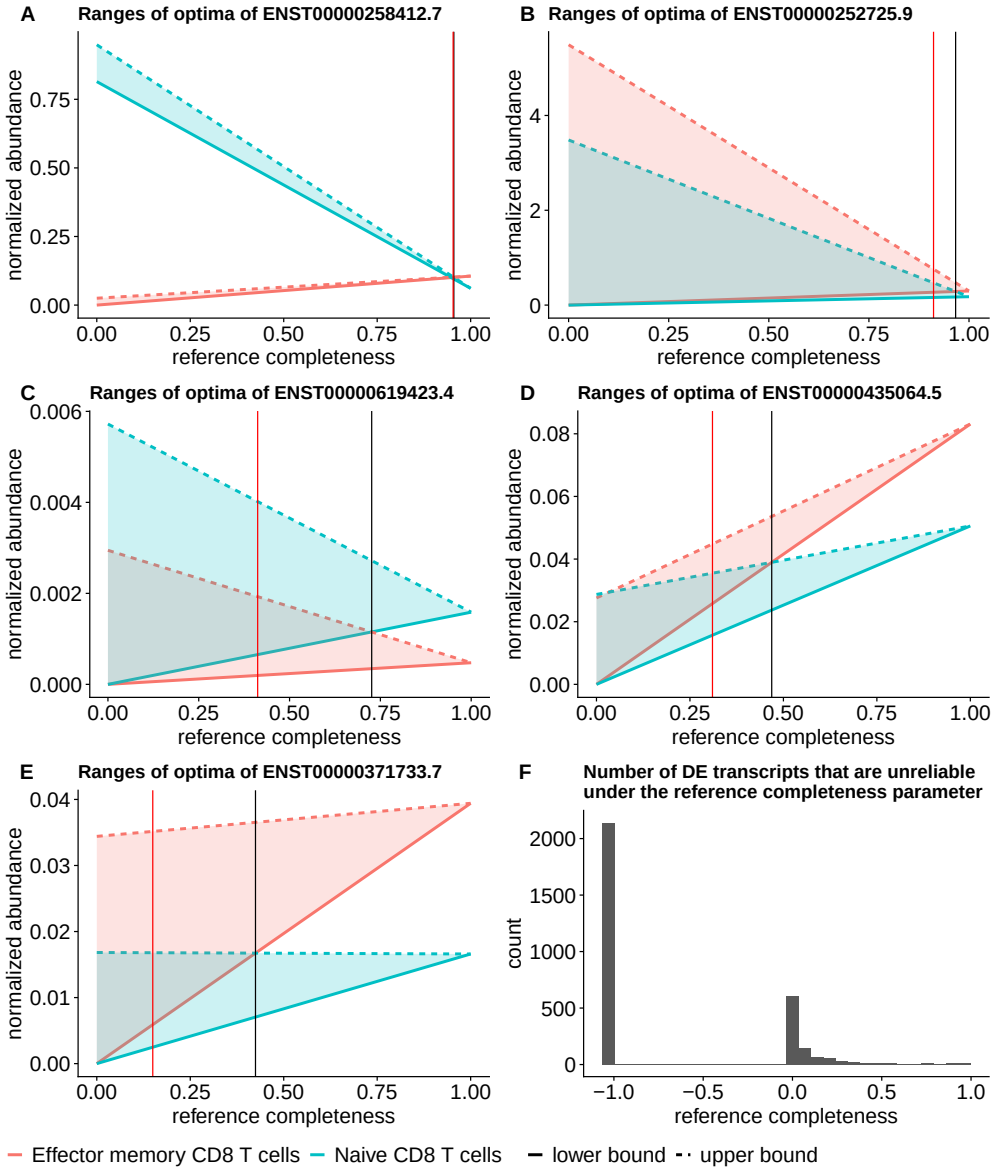

Figure S2: (A–E) Mean ranges of optimal abundances of DE groups. X axis is reference completeness parameter, which is the proportion of expression from the reference transcriptome. Y axis is the normalized abundances, where Salmon estimates are normalized into TPM for linear programming under complete 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. (F) 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.