JUMPER Enables Discontinuous Transcript Assembly in Coronaviruses

Palash Sashittal[1], Chuanyi Zhang[2], Jian Peng[1,3], and Mohammed El-Kebir[1,*]

[1]Dept. of Computer Science, University of Illinois at Urbana-Champaign, IL 61801

[2]Dept. of Electrical & Computer Engineering, University of Illinois at Urbana-Champaign, IL 61801

[3]College of Medicine, University of Illinois at Urbana-Champaign, IL 61801

[*]Correspondence: melkebir@illinois.edu

Abstract

Genes in SARS-CoV-2 and, more generally, in viruses in the order of *Nidovirales* are expressed by a process of discontinuous transcription mediated by the viral RNA-dependent RNA polymerase. This process is distinct from alternative splicing in eukaryotes, rendering current transcript assembly methods unsuitable to *Nidovirales* sequencing samples. Here, we introduce the DISCONTINUOUS TRANSCRIPT ASSEMBLY problem of finding transcripts $\mathcal{T}$ and their abundances $\mathbf{c}$ given an alignment $\mathcal{R}$ under a maximum likelihood model that accounts for varying transcript lengths. Underpinning our approach is the concept of a segment graph, a directed acyclic graph that, distinct from the splice graph used to characterize alternative splicing, has a unique Hamiltonian path. We provide a compact characterization of solutions as subsets of non-overlapping edges in this graph, enabling the formulation of an efficient mixed integer linear program. We show using simulations that our method, JUMPER, drastically outperforms existing methods for classical transcript assembly. On short-read data of SARS-CoV-1 and SARS-CoV-2 samples, we find that JUMPER not only identifies canonical transcripts that are part of the reference transcriptome, but also predicts expression of non-canonical transcripts that are well supported by direct evidence from long-read data, presence in multiple, independent samples or a conserved core sequence. JUMPER enables detailed analyses of *Nidovirales* transcriptomes.

Code availability: Software is available at `https://github.com/elkebir-group/Jumper`

1 Introduction

SARS-CoV-2 is part of the taxonomic order of *Nidovirales*, which comprise enveloped viruses containing a positive-sense, single-stranded RNA genome that encodes for non-structural proteins near the 5' end as well as structural and accessory proteins near the 3' end [1]. Since the ribosome starts at the 5' end, translation of the viral genome only generates the non-structural proteins. Expression of the other genes is achieved by *discontinuous transcription* performed by the viral RNA-dependent RNA polymerase (RdRp) [2], which is present in the non-structural part of the viral genome. Specifically, RdRp may 'jump' over contiguous genomic regions, or *segments*, in the viral RNA template. This process results in *discontinuous transcripts* that have matching 5' and 3' ends and each correspond to a subsequence of segments ordered as in the reference genome. In a recent paper, Kim *et al*. [3] analyzed sequencing samples showing that SARS-CoV-2 has both canonical discontinuous transcription events that produce an intact 3' open reading frame (ORF) as well as non-canonical discontinuous transcription events whose role is unclear. To understand the life cycle of *Nidovirales*, it is critical to assemble the complete set $\mathcal{T}$ of expressed transcripts and their abundances $\mathbf{c}$ (Fig. 1A).

There has been a lot of work in methods for transcript assembly from RNA-sequencing data. Briefly, there are two classes of methods, (i) *de novo* assembly methods and (ii) reference-based methods. The main distinction is that reference-based methods require the reference genome as input while *de novo* methods have no such requirement. As such, *de novo* assembly methods [4–8] are useful when the reference genome is unavailable or when the diversity of different species in the sample is too large. On the other hand, reference-based methods [9–13] are able to achieve higher accuracy as the reference genome provides a scaffold on which to align sequencing reads. Importantly, current methods from both classes are designed to assemble transcripts in eukaryotes, where transcripts expressed by a gene may differ in their composition due to alternative splicing. Alternative splicing is predominantly mediated by the spliceosome, which results in the removal of introns, defined by conserved splice sites, from the transcribed pre-mRNA. This is a distinct process compared to discontinuous transcription in *Nidovirales*, which is mediated by viral RdRp, which results in the removal of contiguous segments due to jumps of the RdRp at locations that are not fully characterized. As such, current methods for transcript assembly that have been mainly designed for use in eukaryotes are not optimized for transcript assembly in *Nidovirales*.

In this study, we introduce the DISCONTINUOUS TRANSCRIPT ASSEMBLY problem of finding discontinuous transcripts $\mathcal{T}$ and their abundances $\mathbf{c}$ (Fig. 1b) given an alignment $\mathcal{R}$. Underpinning our approach is the concept of a segment graph (Fig. 1c), a directed acyclic graph that, distinct from the splice graph used

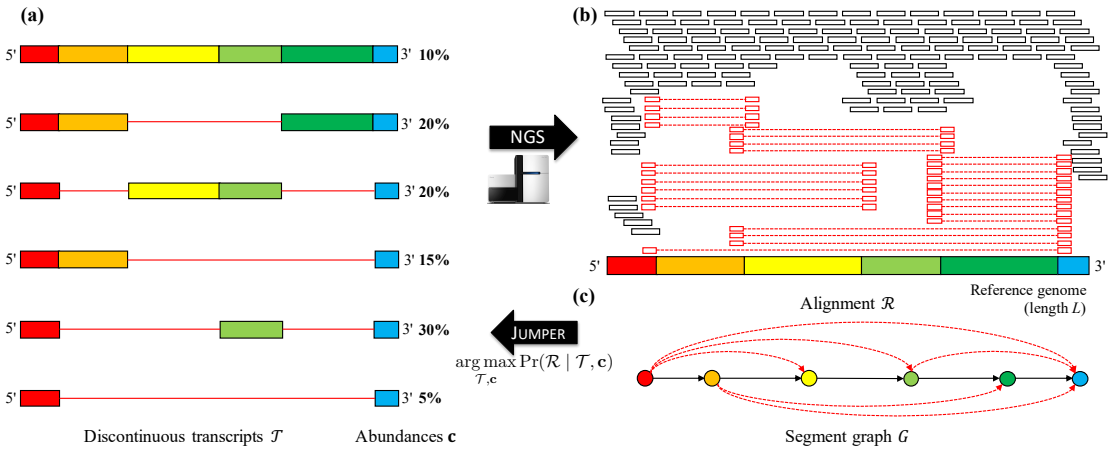


Figure 1: (a) Viruses in the order *Nidovirales* generate a set $\mathcal{T}$ of discontinuous transcripts with varying abundances $\mathbf{c}$ during infection. (b) Next generation sequencing will produce an alignment $\mathcal{R}$ with two types of aligned reads: unphased reads that map to a contiguous genomic region (black) and phased reads that map to distinct genomic regions (red). (c) From $\mathcal{R}$ we obtain the segment graph G, a directed acyclic graph with a unique Hamiltonian path. JUMPER solves the DISCONTINUOUS TRANSCRIPT ASSEMBLY to infer $\mathcal{T}$ and $\mathbf{c}$ with maximum likelihood.

to characterize alternative splicing, has a unique Hamiltonian path. We characterize solutions as subsets of non-overlapping edges in this graph, enabling the formulation of an efficient mixed integer linear program while accounting for paired-end read information. Using simulations, we show that our method, JUMPER, drastically outperforms SCALLOP [9] and STRINGTIE [10], existing methods for transcript assembly. In real data [3], we run JUMPER on paired-end short-read data of virus infected Vero cells and use long-read data of the same sample for validation. We find that JUMPER not only identifies canonical transcripts that are part of the reference transcriptome, but also predict expression of non-canonical transcripts that are well supported by long-read data. Similarly, JUMPER identifies canonical and non-canonical transcripts in SARS-CoV-1 samples [14]. In summary, JUMPER enables detailed *de novo* analyses of *Nidovirales* transcriptomes.

2 Preliminaries and Problem Statement

We define discontinuous transcripts as follows.

Definition 1. Given a reference genome, a *discontinuous transcript* T is a sequence $\mathbf{v}_1, \ldots, \mathbf{v}_{|T|}$ of segments where (i) each segment corresponds to a contiguous region in the reference genome, (ii) segment $\mathbf{v}_i$ precedes segment $\mathbf{v}_{i+1}$ in the reference genome for all $i \in \{1, \ldots, |T| - 1\}$, (iii) segment $\mathbf{v}_1$ contains the 5' end of the reference genome and (iv) segment $\mathbf{v}_{|T|}$ contains the 3' end of the reference genome.

In the literature, discontinuous transcripts that differ from the genomic transcript T_0 are called *subgenomic transcripts*, which correspond to subgenomic RNAs (sgRNAs) [3]. Transcripts $\mathcal{T} = \{T_i\}$ occur in *abundances* $\mathbf{c} = [c_i]$ where $c_i \geq 0$ is the relative abundance of transcript T_i such that $\sum_{i=1}^{|\mathcal{T}|} c_i = 1$. While next-generation sequencing technologies provide high coverage of the viral genome of length L of about 10 to $30\,\mathrm{Kbp}$, they are limited to short reads with fixed length ℓ ranging from 100 to $400\,\mathrm{bp}$. For ease of exposition, we describe the formulation in context of single-end reads, but in practice we use the paired-end information if it is available. As $\ell \ll L$, the identity of the transcript of origin for a given read is ambiguous. Therefore we need to use computational methods to reconstruct the transcripts and their abundances from the sequencing reads. Specifically, given a *Nidovirales* reference genome of length L and reads of a fixed length ℓ, we use a splice-aware aligner such as `STAR` [15] to obtain an alignment $\mathcal{R}$. This alignment provides information about the abundance $\mathbf{c}$ and composition of the underlying transcripts $\mathcal{T}$ in the following two ways. First, the *depth*, or the number of reads along the genome is informative for quantifying the abundance $\mathbf{c}$ of the transcripts. Second, the composition $\mathcal{T}$ of the transcripts is embedded in *phasing reads*, which are reads that align to multiple distinct regions in the reference genome (Fig. 1B).

To make the relationship between $\mathcal{T}$, $\mathbf{c}$ and $\mathcal{R}$ clear, we introduce the *segment graph* G, which is obtained from the phasing reads in a alignment $\mathcal{R}$. As mentioned, each phasing read $r \in \mathcal{R}$ maps to $q \geq 2$ distinct regions in the reference genome. Each pair of regions that are adjacent in the phasing read are separated by two positions v, w (where $w - v \geq 2$) in the reference genome called *junctions*. Thus, each phasing read contributes $2q - 2$ junctions. The collective set of junctions contributed by all phasing reads in $\mathcal{R}$ in combination with positions $\{1, L\}$ induces a partition of the reference genome into closed intervals $[v^-, v^+]$ of junctions that are consecutive in the reference genome (*i.e.* there exists no other junction that occurs in between v^- and v^+). The resulting set of segments equals the node set V of segment graph G (Fig 2a). The edge set E of segment graph G is composed of continuous edges $E^{\rightarrow}$ and discontinuous edges $E^{\curvearrowright}$. Continuous edges $E^{\rightarrow}$ are composed of ordered pairs $(\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+])$ of nodes that correspond to segments that are adjacent in the reference genome, *i.e.* where $v^+ = w^-$. On the other hand, discontinuous edges $E^{\curvearrowright}$ are composed of ordered pairs $(\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+])$ of nodes that corresponds to segments that are adjacent in at least one phasing read in $\mathcal{R}$ but not adjacent in the reference genome (*i.e.* $w^- - v^+ \geq 2$). Fig. 1c shows an example of a segment graph.

Definition 2. Given an alignment $\mathcal{R}$, the corresponding *segment graph* $G = (V, E^{\rightarrow} \cup E^{\curvearrowright})$ is a directed graph whose node set V equals the set of segments induced by the junctions of phasing reads in $\mathcal{R}$ and whose edge set $E = E^{\rightarrow} \cup E^{\curvearrowright}$ is composed of edges $(\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+])$ that are either continuous, *i.e.* $v^+ = w^-$, or discontinuous, *i.e.* $w^- - v^+ \geq 2$ and there exists a phasing read where junctions v^+ and

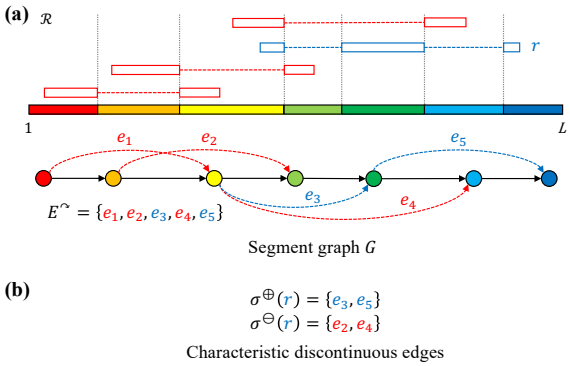


Figure 2: (a) Phasing reads in an alignment $\mathcal{R}$ define a set of junctions, which in turn define the segment graph G. (b) Each phasing read has characteristic discontinuous edges indicating the set $\sigma^{\oplus}$ of discontinuous edges present in the read as well as conflicting/overlapping discontinuous edges $\sigma^{\ominus}$. Here, phasing read r (blue), has $\sigma^{\oplus}(r) = \{e_3, e_5\}$ and $\sigma^{\ominus}(r) = \{e_2, e_4\}$. Note that e_1 is not included in $\sigma^{\ominus}(r)$ as it does not overlap with $\pi(r) = \{e_3, e_5\}$.

w^- are adjacent.

We note that the segment graph G is closely related to the splice graph used in regular transcript assembly where transcripts correspond to varying sequences of exons due to alternative splicing. The key difference, however, is that an alignment $\mathcal{R}$ generated from reads obtained from discontinuous transcripts induces a segment graph G that is a directed acylic graph (DAG) with a unique Hamiltonian path. This is because, as stated in Definition 1, discontinuous transcripts $\mathcal{T}$ have matching 5' and 3' ends, and, although their comprising segments may vary, their order follows the reference genome.

Observation 1. Segment graph G is a directed acyclic graph with a unique Hamiltonian path.

The unique Hamiltonian path of G corresponds to the sequence of continuous edges $E^{\rightarrow}$. This path corresponds to the whole viral genome which is generated by the RdRp during the replication step [2]. Moreover, by the above observation, G has a unique source node $\mathbf{s}$ and sink node $\mathbf{t}$. Importantly, each transcript $T \in \mathcal{T}$ that is compatible with an alignment $\mathcal{R}$ corresponds to an $\mathbf{s} - \mathbf{t}$ path $\pi(T)$ in G. Here, a *path* π is a subset of edges E that can be ordered $(\mathbf{v}_1, \mathbf{w}_1), \ldots, (\mathbf{v}_{|\pi|}, \mathbf{w}_{|\pi|})$ such that $\mathbf{w}_i = \mathbf{v}_{i+1}$ for all $i \in [|\pi| - 1] = \{1, \ldots, |\pi| - 1\}$. While splice graphs are DAGs and typically have a unique source and sink node as well, they do not necessarily contain a Hamiltonian path [9, 16–18].

Our goal is to find a set $\mathcal{T}$ of transcripts and their abundances $\mathbf{c}$ that maximize the posterior probability

$$\Pr(\mathcal{T}, \mathbf{c} \mid \mathcal{R}) \propto \Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c}) \Pr(\mathcal{T}, \mathbf{c}).$$

Under an uninformative, flat prior, this is equivalent to maximizing the probability $\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$. We use the segment graph G to compute the probability $\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$ of observing an alignment $\mathcal{R}$ given transcripts $\mathcal{T}$ and abundances $\mathbf{c}$. We follow the generative model which has been extensively used for transcription

quantification [19–21]. The notations used in this paper best resemble the formulation described in [18]. Let $\mathcal{R}$ be composed of reads be $\{r_1, \ldots, r_n\}$ and the set $\mathcal{T}$ of transcripts be $\mathcal{T} = \{T_1, \ldots, T_k\}$ with lengths $L_1, \ldots, L_k$ and abundances $\mathbf{c} = [c_1, \ldots, c_k]$. In line with current literature, reads $\mathcal{R}$ are generated independently from transcripts $\mathcal{T}$ with abundances $\mathbf{c}$. Further, we must marginalize over the set of transcripts $\mathcal{T}$ as the transcript of origin of any given read is typically unknown, since $\ell \ll L$. Moreover, we assume that the fixed read length ℓ is much smaller than the length L_i of any transcript T_i. As such, we that $\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$ equals

$$\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c}) = \prod_{j=1}^{n} \Pr(r_j \mid \mathcal{T}, \mathbf{c}) = \prod_{j=1}^{n} \frac{1}{\sum_{b=1}^{k} c_b L_b} \sum_{i:\pi(T_i) \supseteq \pi(r_j)} c_i, \tag{1}$$

where $\pi(T) \subseteq E$ is the $\mathbf{s} - \mathbf{t}$ path corresponding to transcript T and $\pi(r) \subseteq E$ is the path induced by the ordered sequence of segments (or nodes of G) spanned by read r. By construction, $\pi(T) \supseteq \pi(r)$ is a necessary condition for transcript T to be a candidate transcript of origin of read r. Appendix A gives the derivation of the above equation (Eq. (1)). Our goal is to find $\arg\max_{\mathcal{T},\mathbf{c}} \Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$, leading to the following problem.

Problem 1 (DISCONTINUOUS TRANSCRIPT ASSEMBLY (DTA))**.** Given alignment $\mathcal{R}$ and integer k, find discontinuous transcripts $\mathcal{T} = \{T_1, \ldots, T_k\}$ and abundances $\mathbf{c} = [c_1, \ldots, c_k]$ such that (i) each transcript $T_i \in \mathcal{T}$ is an $\mathbf{s} - \mathbf{t}$ path in segment graph G, and (ii) $\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$ is maximum.

3 Combinatorial Characterization of Solutions

Eq. (1) defines the probability $\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$ in terms of the observed reads r and their induced paths $\pi(r) \subseteq E(G)$ in the segment graph G. The authors in [18] use this characterization of reads as paths in a general *splice* graph to account for ambiguity in the transcript of origin for the reads. For a general splice graph, such a characterization is required to capture all the possible observed reads. However, in our setting, where the segment graph G is a DAG with a unique Hamiltonian path, it is possible to describe each read and each transcript *uniquely* in a more concise form. Each path in the segment graph is characterized by a set of *non-overlapping* discontinuous edges. To describe this, we introduce the following definition.

Definition 3. Two edges $(\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+])$ and $(\mathbf{x} = [x^-, x^+], \mathbf{y} = [y^-, y^+])$ of G *overlap* if the open intervals (v^+, w^-) and (x^+, y^-) intersect, *i.e.* $(v^+, w^-) \cap (x^+, y^-) \neq \emptyset$.

For any transcript T corresponding to an $\mathbf{s} - \mathbf{t}$ path in G, for which we are only given its discontinuous edges $\sigma(T)$, the continuous edges of T are uniquely determined by G and $\sigma(T)$. That is, the continuous

edges of T equal precisely the subset of continuous edges $E^{\rightarrow}$ that do *not* overlap with any of the discontinuous edges in $\sigma(T)$. Conversely, given an $\mathbf{s}-\mathbf{t}$ path $\pi(T)$ of G the corresponding set of discontinuous edges is given by $\sigma(T) = \pi(T) \cap E^{\curvearrowright}$. Thus, we have the following proposition with the proof in Appendix B.1.

Proposition 1. There is a bijection between subsets of discontinuous edges that are pairwise non-overlapping and $\mathbf{s} - \mathbf{t}$ paths in G.

In a similar vein, rather than characterizing a read r by its induced path $\pi(r) \subseteq E$ in the segment graph, we characterize a read r by a pair $(\sigma^{\oplus}(r), \sigma^{\ominus}(r))$ of *characteristic discontinuous edges*. Here, $\sigma^{\oplus}(r)$ is the set of discontinuous edges that must be present in any transcript that could generate read r, *i.e.* $\sigma^{\oplus}(r) = \pi(r) \cap E^{\curvearrowright}$. Conversely, $\sigma^{\ominus}(r)$ is the set of discontinuous edges that must be absent in any transcript that could generate read r due to the unidirectional nature of RdRp transcription. Thus, the set $\sigma^{\ominus}(r)$ consists of discontinuous edges $E^{\curvearrowright} \setminus \sigma^{\oplus}$ that overlap with any edge in $\pi(r)$. Clearly, while $\sigma^{\oplus}(r) \cap \sigma^{\ominus}(r) = \emptyset$, it need not hold that $\sigma^{\oplus}(r) \cup \sigma^{\ominus}(r)$ equals $E^{\curvearrowright}$ (see Fig. 2b). Formally, we define $(\sigma^{\oplus}(r), \sigma^{\ominus}(r))$ as follows.

Definition 4. The *characteristic discontinuous edges* of a read r are a pair $(\sigma^{\oplus}(r), \sigma^{\ominus}(r))$ where $\sigma^{\oplus}(r)$ is the set of discontinuous edges present in read r, *i.e.* $\sigma^{\oplus}(r) = \pi(r) \cap E^{\curvearrowright}$, and $\sigma_i^{\ominus}$ is the set of discontinuous edges $(\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+]) \in E^{\curvearrowright} \setminus \sigma^{\oplus}(r)$ that *overlaps* with an edge $(\mathbf{x} = [x^-, x^+], \mathbf{y} = [y^-, y^+])$ in $\pi(r)$.

We have the following result with the proof given in Appendix B.1.

Proposition 2. Let G be a segment graph, T be a transcript and r be a read. Then, $\pi(T) \supseteq \pi(r)$ if and only if $\sigma(T) \supseteq \sigma^{\oplus}(r)$ and $\sigma(T) \cap \sigma^{\ominus}(r) = \emptyset$.

Hence, we may rewrite the likelihood $\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$ as

$$\prod_{j=1}^{n} \frac{1}{\sum_{b=1}^{k} c_b L_b} \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i. \tag{2}$$

where $X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})$ be the subset of indices i corresponding to transcripts $T_i \in \mathcal{T}$ where $\sigma(T_i) \supseteq \sigma_j^{\oplus}$ and $\sigma(T_i) \cap \sigma_j^{\ominus} = \emptyset$. Note that the only difference between Eq. (2) and the formulation in Eq. (1) is the way that the candidate transcripts of origin for a given read are described. In Eq. (1), they are described as paths in the splice graph wheres in Eq. (2), they are described by sets of pairwise non-overlapping discontinuous edges in the segment graph. This leads to the following theorem.

Theorem 1. For any alignment $\mathcal{R}$, transcripts $\mathcal{T}$ and abundances $\mathbf{c}$, Equations (1) and (2) are identical.

Although we have described the formulation for single-end reads, this characterization is applicable to paired-end and even synthetic long reads. Moreover, our implementation provides support for both single-end and paired-end read samples with a fixed read length. The above characterization using discontinuous edges allows us to reduce the number of terms in the likelihood function since multiple reads can be characterized by the same characteristic discontinuous edges. We describe this in detail in Section 4.

4 Methods

To solve the DTA problem, we use the results of Section 3 to write a more concise form of the likelihood. Specifically, let $\mathcal{S} = \{(\sigma_1^{\oplus}, \sigma_1^{\ominus}), \ldots, (\sigma_m^{\oplus}, \sigma_m^{\ominus})\}$ be the set of characteristic discontinuous edges generated by the reads in alignment $\mathcal{R}$. Let $\mathbf{d} = \{d_1, \cdots, d_m\}$ be the number of reads that map to each pair in $\mathcal{S}$. Using that reads r with identical characteristic discontinuous edges $(\sigma^{\oplus}(r), \sigma^{\ominus}(r))$ have identical probabilities $\Pr(r \mid \mathcal{T}, \mathbf{c})$, we obtain the following mathematical program for the log-likelihood $\log \Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$ (see Appendix A for derivation).

$$\max_{\mathcal{T},\mathbf{c}} \sum_{j=1}^{m} d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i - n \log \sum_{b=1}^{k} c_b L_b \tag{3}$$

$$\text{s.t. } \pi(T_i) \text{ is an } \mathbf{s}-\mathbf{t} \text{ path in the segment graph } G \qquad \forall i \in [k], \tag{4}$$

$$\sum_{i=1}^{k} c_i = 1, \tag{5}$$

$$c_i \geq 0 \qquad \forall i \in [k]. \tag{6}$$

Observe that the first sum (over reads) is concave and the second sum (over transcripts) is convex. Since we are maximizing, our objective function would ideally be concave. In Appendix B.1, we prove the following lemma, which enables us to remove the second term using a scaling factor for the relative abundances $\mathbf{c}$ that does not alter the solution space.

Lemma 1. Let $D > 0$ be a constant, $\overline{c}_i(\mathbf{c}) = c_i D / \sum_{j=1}^{k} c_j L_j$ and $c_i(\overline{\mathbf{c}}) = \overline{c}_i / \sum_{j=1}^{k} \overline{c}_j$ for all $i \in [k]$. Then, $(\mathcal{T}, \mathbf{c} = [c_1(\overline{\mathbf{c}}), \ldots, c_k(\overline{\mathbf{c}})])$ is an optimal solution for (3)-(6) if and only if $(\mathcal{T}, \overline{\mathbf{c}} = [\overline{c}_1(\mathbf{c}), \ldots, \overline{c}_k(\mathbf{c})])$

is an optimal solution for

$$\max_{\mathcal{T},\bar{\mathbf{c}}} \sum_{j=1}^{m} d_j \log \sum_{i \in X(\mathcal{T},\sigma_j^{\oplus},\sigma_j^{\ominus})} \bar{c}_i \tag{7}$$

$$\text{s.t. } \pi(T_i) \text{ is an } \mathbf{s}-\mathbf{t} \text{ path in the segment graph } G \qquad \forall i \in [k], \tag{8}$$

$$\sum_{i=1}^{k} \bar{c}_i L_i = D, \tag{9}$$

$$\bar{c}_i \geq 0 \qquad \forall i \in [k]. \tag{10}$$

We formulate the mathematical program given in Lemma 1 as a mixed integer linear program. More specifically, we encode (i) the composition of each transcript T_i as a set $\sigma(T_i)$ of non-overlapping discontinuous edges, (ii) the abundance c_i and length L_i of each transcript T_i, (iii) the total abundance $\sum_{i \in X(\mathcal{T},\sigma_j^{\oplus},\sigma_j^{\ominus})} c_i$ of transcripts supported by characteristic discontinuous edges $(\sigma_j^{\oplus}, \sigma_j^{\ominus})$, and (iv) a piecewise linear approximation of the $\log$ function using a user-specified number h of breakpoints. We will describe (i) and (ii) in the following and refer to Appendix B.2 for (iii) and (iv).

Transcript composition. We begin modeling (8), which states that each transcript T_i must correspond to an $\mathbf{s}-\mathbf{t}$ path in the segment graph G. Using Proposition 1, we introduce binary variables $\mathbf{x} \in \{0,1\}^{|E^{\curvearrowright}| \times k}$ to encode the presence of discontinuous edges in each of the k $\mathbf{s}-\mathbf{t}$ paths corresponding to the k transcripts in $\mathcal{T}$. For any discontinuous edge $e = (\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+])$, let $I(e)$ denote the open interval (v^+, w^-) between the two segments $\mathbf{v}$ and $\mathbf{w}$. By Proposition 1, it must hold that $I(e) \cap I(e') = \emptyset$ for any two distinct discontinuous edges e and e' assigned to the same transcript. To encode this, we impose

$$x_{e,i} + x_{e',i} \leq 1, \quad \forall i \in [k], e, e' \in E^{\curvearrowright} \text{ s.t. } e \neq e', I(e) \cap I(e') \neq \emptyset.$$

Transcript abundance and length. We introduce non-negative continuous variables $\mathbf{c} = [c_1, \ldots, c_k]$ that encode the abundance of the k transcripts. The scale of these abundances depends on the choice of D. We choose $D = \ell^*$ where ℓ^* is the length of the shortest $\mathbf{s}-\mathbf{t}$ path in the segment graph G. Substituting $D = \ell^*$ into (9) yields $\sum_{i=1}^{k} c_i L_i = \ell^*$.

Since $c_i L_i \leq \sum_{j=1}^{k} c_j L_j = \ell^*$ and $L_i \geq \ell^*$, we have that $c_i \leq 1$. To model the product $c_i L_i$ of the length L_i of a transcript T_i and its abundance c_i, we focus on individual discontinuous edges e. For any discontinuous edge $e = (\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+])$, let $L(e) = w^- - v^+$ be the length of the interval. Observe that

$$c_i L_i = c_i L - c_i \sum_{e \in \sigma(T_i)} L(e) = c_i L - \sum_{e \in E^{\curvearrowright}} c_i x_{e,i} L(e).$$

We introduce continuous variables $z_e \in [0,1]^k$ and encode the product $z_{e,i} = c_i x_{e,i}$ for all $e \in E^{\curvearrowright}$ as

$$\begin{aligned} z_{e,i} &\leq c_i, \quad \forall i \in [k], \\ z_{e,i} &\leq x_{e,i}, \quad \forall e \in E^{\curvearrowright}, i \in [k], \\ z_{e,i} &\geq c_i + x_{e,i} - 1, \quad \forall e \in E^{\curvearrowright}, i \in [k]. \end{aligned}$$

Therefore, we may represent $\sum_{i=1}^{k} c_i L_i = \ell^*$ as

$$\sum_{i=1}^{k} c_i L - \sum_{i=1}^{k} \sum_{e \in E^{\curvearrowright}} z_{e,i} L(e) = \ell^*. \tag{11}$$

The resulting formulation has $O(|E^{\curvearrowright}|k + |E^{\curvearrowright}|m + mh)$ variables, where h is the user-specified number of breakpoints used in the piecewise linear approximation of the log function. This number includes $|E^{\curvearrowright}|k$ binary variables. The number of constraints is $O(k|E|^2 + |E|km)$.

Progressive heuristic. In practice, the number of discontinuous edges in the segment graph is inflated due to ambiguity in the exact location at which the RdRp jumps as well as sequencing and alignment errors. This leads to large number of binary variables in our MILP (we have $k \cdot |E^{\curvearrowright}|$ binary variables) which can make the MILP intractable. In order to approximately solve the problem with large values of k, we implement a progressive heuristic. Our heuristic takes as input the alignment $\mathcal{R}$ and an integer k, which is the maximum number of transcripts in the solution. At each iteration $p \leq k$, we are given a set $\mathcal{T}$ of $p-1$ previously computed transcripts and seek a new transcript T' by solving the MILP (Eq. (27)-(31)) using function SOLVEILP with additional constraints to fix the values of the variables that encode the presence/absence of discontinuous edges for the transcripts in $\mathcal{T}$. The resulting reduction in number of binary variables from $|E^{\curvearrowright}|k$ to $|E^{\curvearrowright}|$ improves the running time of the MILP. As an additional optimization, we re-estimate the abundances of a new set $\mathcal{T}'$ of transcripts. This set contains all transcripts in $\mathcal{T}$ as well additional transcripts corresponding to all possible subsets of discontinuous edges $\sigma(T')$ of the newly identified transcript T', identified by the function EXPAND. We solve a linear programming (Eq. (32)-(34)) with function SOLVELP to re-estimate the abundances $\mathbf{c}'$ of $\mathcal{T}'$, retaining only the top p transcripts T_i from $\mathcal{T}'$ with the largest abundances $c_i L_i$. We terminate upon convergence, *i.e.* if $\mathcal{T} = \mathcal{T}'$, or if the number p of iterations reaches the number k. Algorithm 1 provides the pseudo code of the progressive heuristic implemented in JUMPER. The details of the subproblems SOLVEILP and SOLVELP are given in Appendix B.3.

Implementation details. Matching core sequences that mediate the discontinuous transcription by RdRp lead to ambiguity in precise location of breakpoint during alignment of spliced reads. Therefore, in practice we observe multiple discontinuous edges with closely spaced 5' and 3' breakpoints. Moreover, false

Algorithm 1: JUMPER($\mathcal{R}$, k)

$(\mathcal{T}, \mathbf{c}) \leftarrow (\emptyset, [])$
for $p \leftarrow 1$ **to** k **do**
  $T' \leftarrow$ SOLVEILP($\mathcal{T}$)
  $\mathcal{T}' \leftarrow \mathcal{T} \cup$ EXPAND(T')
  $\mathbf{c}' \leftarrow$ SOLVELP($\mathcal{T}'$)
  Sort $(\mathcal{T}', \mathbf{c}')$ s.t. $L_i c'_i \geq L_{i+1} c'_{i+1}$ for all $i \in \{1, \ldots, |\mathcal{T}'| - 1\}$
  $(\mathcal{T}', \mathbf{c}') \leftarrow (\{T_1, \ldots, T_p\}, [c'_1, \ldots, c'_p])$
  if $\mathcal{T}' \neq \mathcal{T}$ **then**
    $(\mathcal{T}, \mathbf{c}) \leftarrow (\mathcal{T}', \mathbf{c}')$
  else
    return $(\mathcal{T}, \mathbf{c})$
return $(\mathcal{T}, \mathbf{c})$

positive discontinuous edges are introduced due to sequencing and alingment errors. We use a threshold on the number of spliced reads supporting a discontinuous edge to filter false positive edges with low support. This parameter can also be used to reduce computational burden and focus on the highly expressed transcripts in the sample. A discussion on the choice of the thresholding parameter is provided in Section B.4. JUMPER is implemented in Python 3 using `Gurobi` [22] (version 9.0.3) to solve the MILP and `pysam` [23] for reading and processing the input BAM file. JUMPER is available at `https://github.com/elkebir-group/Jumper`.

5 Results

Before we describe our results for coronavirus samples, we establish some terminology that will be used in this section. A discontinuous edge ($\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+]$) is *canonical* provided its 5' junction v^+ occurs in the transcription regulating leader sequence (TRS-L), *i.e.* between positions 55 and 85 and the first occurrence of 'AUG' downstream of the 3' junction w^- position coincides with the start codon of a known ORF, otherwise the discontinuous edge is called *non-canonical*. In a similar vein, a transcript is *canonical* if it contains at most one canonical and no non-canonical discontinuous edges, otherwise the transcript is *non-canonical*. We run our experiments on a server with two 2.6 GHz CPUs and 512 GB of RAM. We follow the approach of SCALLOP and use SALMON [20] to accurately re-estimate the abundance of the transcripts

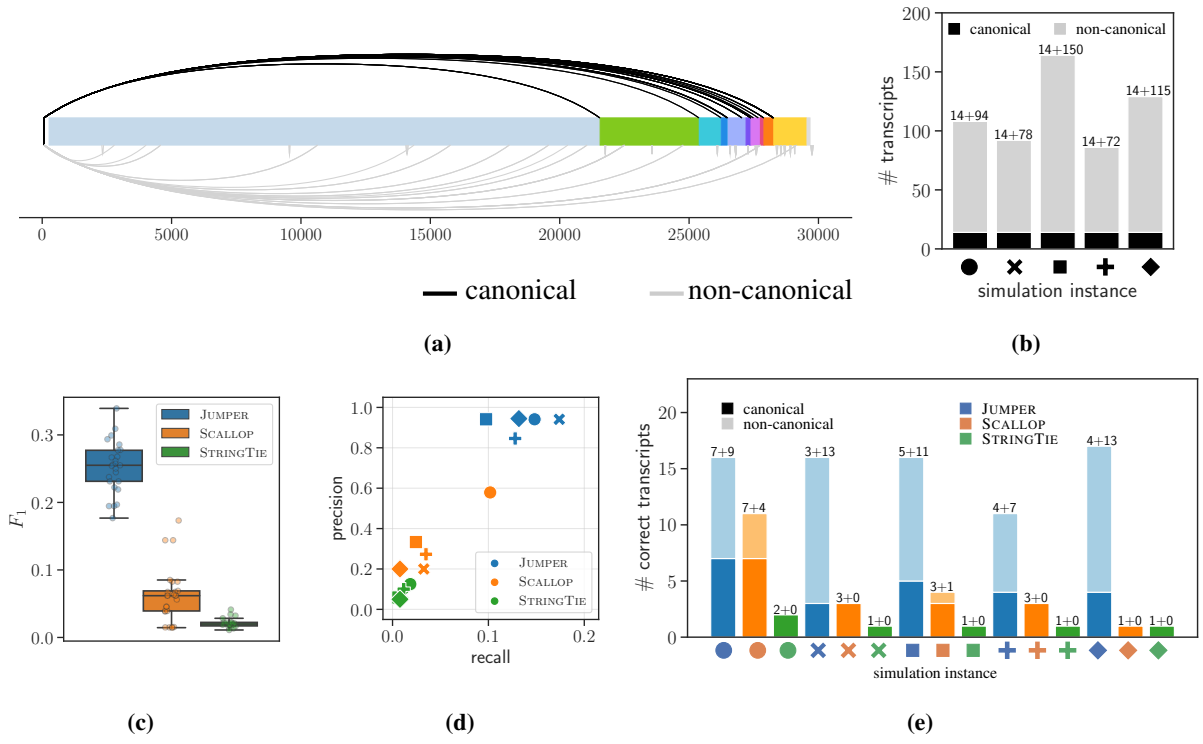


Figure 3: JUMPER consistently outperforms SCALLOP and STRINGTIE in reconstruction of viral transcripts from simulated SARS-CoV-2 sequencing data. (a) Sashimi plot showing the canonical (black) and non-canonical (gray) discontinuous mappings supported by reads in short-read sample SRR11409417. (b) Number of canonical and non-canonical transcripts for 5 simulation instances of $(\mathcal{T}, \mathbf{c})$ generated under the negative-sense discontinuous transcription model. (c) F_1 score of the three methods (JUMPER, SCALLOP and STRINGTIE) for all the 25 simulated instances under the negative-sense discontinuous transcription model. (d) Precision and recall values of the three methods with one of sequencing experiment for each simulated instance of $(\mathcal{T}, \mathbf{c})$ under the negative-sense discontinuous transcription model as input. (e) Total number of canonical and non-canonical transcripts recalled by the three methods for the simulated instances shown in panel (d).

assembled by JUMPER.

5.1 Simulations

We begin our simulations with a segment graph G obtained from a short-read sample (SRR11409417). Following Kim *et al.* [3], we used `fastp` to trim short reads (trimming parameter set to 10 nucleotides), which were input to `STAR` run in two-pass mode yielding an alignment $\mathcal{R}$. Fig. 3a shows the sashimi plot of the *canonical* and the *non-canonical* discontinuous edges (mappings) supported by the reads in the sample. From $\mathcal{R}$, we obtained G by only including discontinuous edges supported by at least 20 reads. The segment graph for this sample has $|V| = 39$ nodes and $|E| = 67$ edges, which include $|E^{\curvearrowright}| =$

29 discontinuous edges and $|E^{\rightarrow}| = 38$ continuous edges. The former are subdivided into 14 canonical discontinuous edges that produce a known ORF and 15 non-canonical discontinuous edges. The next step of our simulation pipeline is to generate transcripts $\mathcal{T}$ and their abundances $\mathbf{c}$ for the given segment graph. We used the negative-sense discontinuous transcription model (described in Appendix C.1). Upon generating the transcripts, we simulated the generation and sequencing of RNA-seq data, and aligned the simulated reads using `STAR` [15]. We generated 5 independent pairs $(\mathcal{T}, \mathbf{c})$ of transcripts and abundances under the negative-sense discontinuous transcription model. Fig. 3b shows the number of transcripts generated from each simulation using the negative-sense discontinuous transcription model. For each pair $(\mathcal{T}, \mathbf{c})$ we ran 5 paired-end short read sequencing simulations using `polyester` [24]. Therefore, we generated a total of $5 \times 5 = 25$ simulated instances. Details are provided in Appendix C.1. Note that our method JUMPER does *not* use the negative-sense discontinuous transcription model to infer the viral transcripts from the simulated data.

We compare the performance of our method JUMPER with two other reference-based transcript assembly methods, SCALLOP and STRINGTIE. All the methods were run with their default parameters. To avoid including false-positive discontinuous edges, JUMPER discards discontinuous edges with support less than 100 reads ($\Lambda = 100$) from the segment graph. The reader is referred to Appendix B.4 for a discussion on the choice of the Λ parameter and comparison with other methods. We compare the transcripts predicted by the three methods to the ground truth transcripts. Specifically, a predicted transcript is *correct* (or matches the ground truth) if there exists a transcript in the ground truth whose junction positions match the predicted junctions positions within a tolerance of 10 nucleotides.

Fig. 3c shows the F_1 score (harmonic mean of recall and precision) of the three methods for all the simulated sequencing data instances under the negative-sense discontinuous transcription model. JUMPER achieves a much higher F_1 score compared to SCALLOP and STRINGTIE. For the 25 simulation instances, JUMPER has a median F_1 score of 0.255 (range $[0.176, 0.339]$) while SCALLOP has a median F_1 score of 0.062 (with range $[0.0145, 0.173]$) and STRINGTIE has a median F_1 score of 0.01904 (with range $[0.0114, 0.0412]$). Fig. C2 shows that this trend holds for recall and precision values and our method has a run-time comparable to the existing methods. To investigate the effect of threshold parameter Λ on the performance of JUMPER, we ran our method on the simulated instances with $\Lambda \in \{10, 50, 100, 200\}$. Fig. C3 shows that JUMPER outperforms SCALLOP and STRINGTIE for all values of Λ, although it incurs significantly more runtime for $\Lambda = 10$. Thus, we find that JUMPER consistently performs better than SCALLOP and STRINGTIE.

To better understand the tradeoff between precision and recall, we zoom in one five distinct pairs $(\mathcal{T}, \mathbf{c})$

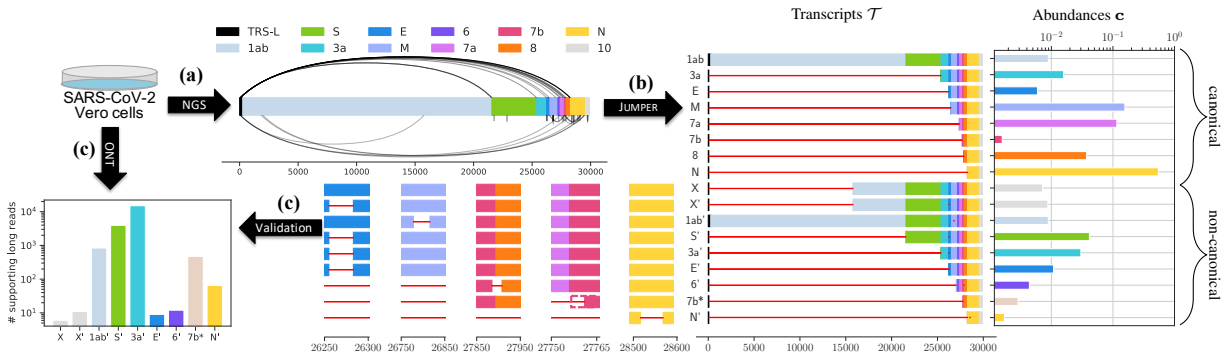


Figure 4: Using short-read data of SARS-CoV-2 infected Vero cells [3], JUMPER identifies canonical and non-canonical transcripts that are well supported by long-read sequences of the same sample. (a) The segment graph for the short-read data contains both canonical (above) and non-canonical (below) edges. (b) JUMPER assembles 8 canonical transcripts and 9 non-canonical transcripts and estimates their abundances with zoomed-in view of the non-canonical transcripts X, X', 1ab', S', 3a', E', 6', 7b* and N'. (c) All non-canonical transcripts predicted by JUMPER are well supported by long-read data. (NGS: Next Generation Sequencing; ONT: Oxford Nanopore Technologies)

of simulation instance. Fig. 3d shows the precision and recall values achieved by each method for each of these five simulation instances, demonstrating that JUMPER consistently outperforms both SCALLOP and STRINGTIE. On average, JUMPER recalls 5 times more transcripts than SCALLOP and 11 times more transcripts than STRINGTIE while also having higher precision in all simulated cases. Fig. C4 shows that all three methods produce similar precision and recall values for different sequencing replicates of the same simulated instance of $(\mathcal{T}, \mathbf{c})$, demonstrating consistency in results. Finally, Fig. 3e shows the number of canonical and non-canonical transcripts generated by the three methods for each simulated instance that match the ground truth. In summary, we found that JUMPER correctly predicts higher number of both canonical and non-canonical transcripts compared to SCALLOP and STRINGTIE for all the simulated instances. We observe similar trends on simulated instances of a human gene (see Appendix C.4). The results from all the simulations are summarized in Table C1.

5.2 Transcript assembly in SARS-CoV-2

In a recent paper, Kim *et al.* [3] explored the transcriptomic architecture of SARS-CoV-2 by performing short-read as well as long-read sequencing of Vero cells infected by the virus. The authors used oligo(dT) amplification, which targets the poly(A) tail at the 3' end of messenger RNAs, thus limiting positional bias that would occur when using SARS-CoV-2 specific primers [25, 26]. Subsequently, the authors aligned the resulting reads using splice-aware aligners, *i.e.* `STAR` [15] for the short-read sample (median depth of 1763) and `minimap2` [27] for the long-read sample (median depth of 6707 and mean length of 2875 bp). For both

complementary sequencing techniques, the authors observed phasing reads that were indicative of canonical as well as non-canonical transcription events. While the authors quantified the fraction of phasing reads supporting each discontinuous transcription event, they did not attempt to assemble complete transcripts.

We use JUMPER to reconstruct the SARS-CoV-2 transcriptome of the short-read sequencing sample using the BAM file obtained by running Kim *et al.*'s pipeline [3], followed by running SALMON to identify precise transcript abundances. We note that running SCALLOP on the short-read data resulted in only a single, complete canonical transcript (corresponding to 'N') but required subsampling of the BAM file (to 20%) due to memory constraints, whereas STRINGTIE produced two incomplete transcripts ('ORF3a' and and a non-canonical transcript with low support). We build a segment graph with $|V| = 59$ nodes and $|E| = 93$ edges comprising of $|E^{\curvearrowright}| = 35$ most abundant discontinuous edges, 18 of which canonical and 17 non-canonical (Fig. 4a). JUMPER identifies 33 transcripts, 17 of which have an abundance of at least 0.001 as determined by SALMON. We focus our attention to these 17 transcripts (Fig. 4b). A subset of 8 transcripts are canonical, thus containing at most one discontinuous edge with the 5' junction in TRS-L and the first ATG downstream of the 3' junction coinciding with the start codon of a known ORF. These canonical transcripts correspond to ORF1ab, ORF3a, E, M, ORF7a, ORF7b, ORF8, N. In particular, ORF1ab (abundance of 0.008) corresponds to the complete viral genome, necessary for viral replication. Notably, ORF10 is the only missing ORF in the identified transcriptome, which is in line with previous studies [3,28] that did not find evidence for active transcription of ORF10.

As mentioned, JUMPER inferred 9 non-canonical transcripts, denoted as X, X', 1ab', S', 3a', 6', E', 7b* and N'. Among these, transcripts 1ab', S', 3a' and 6' encode for the 1ab polypeptide, spike protein S and accessory protein 3a and accessory protein 6, respectively. Transcripts X and X' both contain the discontinuous edge going from position 68 to 15774, with the latter containing an additional discontinuous edge from position 26256 to 26284. The 5' end of the common discontinuous edge occurs within TRS-L, whereas the 3' end occurs in the middle of ORF1b but is out of frame with respect to the starting position of ORF1b (13468). Specifically, the start codon 'ATG' downstream of the 3' end is located at position 15812 and occurs within nsp12 (RdRp) and the first stop codon is located at position 15896, encoding for a peptide sequence of 28 amino acids. Interestingly, when we examine the reference genome, we observe matching sequences "GAACTTTAA" near the 5' and 3' junctions of the discontinuous edge common to X and X', possibly explaining why the viral RdRp generated this jump (Fig. C5a,b). Strikingly, both matching sequences are conserved within the *Sarbecovirus* subgenus but not in other subgenera of the *Betacoronavirus* genus (Fig. C5a,c). To further corroborate this transcript, we examined short and long-read SARS-CoV-2 sequencing samples from the NCBI Sequence Read Archive (SRA). Specifically, we looked for the presence

of reads potentially originating from transcript X focusing on high-quality samples with 100 or more leader-spanning reads (reads whose 5' end intersects with the TRS-L region). A read r *supports* a transcript T if the discontinuous edges of r exactly match those of T, *i.e.* $\pi(r) \subseteq \pi(T)$ and $|\sigma^{\oplus}(r)| = |\sigma(T)|$ (Fig. C6). We find ample support for transcript X in both short and long-read samples on SRA, with 100 out of 351 short-read samples and 81 out of 653 long-read samples having more than 0.001 leader-spanning reads supporting transcript X (Fig. C7). We note that although this discontinuous transcription event was also observed in [28], the authors found no evidence of this transcript leading to protein product in the ribo-seq data. Further research into a potentially regulatory function of this transcript is required.

As stated, the difference between transcripts X and X' is that the latter includes an additional discontinuous edge, corresponding to a short jump of ~ 27 nucleotides between positions 26256 and 26284. This is an in frame deletion inside ORF E, resulting in the loss of 9 amino acids that span the N-terminal domain (4 amino acids) and the transmembrane domain (5 amino acids) of the E protein [29]. A similar in-frame deletion of 24 nucleotides (from position 26259 to 26284) was observed by Finkel *et al.* [28] that resulted in the loss of a subset of 8 out of the 9 amino acids in the deletion that we observed. Furthermore, it is possible that this common deletion is being selected for during passage in Vero E6 cells, which were used by both Kim *et al.* [3] and Finkel *et al.* [28]. Non-canonical transcripts S', 3a' and E' also contain the same discontinuous edge from position 26256 to 26284. While transcript E' produces a version of protein E with 9 missing amino acids, transcripts S' and 3a' produce complete viral proteins S and 3a, respectively. Non-canonical transcript 6' differs from the canonical transcript 6, containing a jump from position 27886 to 27909. This jump is downstream of ORF6 and therefore does not disrupt the translation of accessory protein 6. Similarly, transcript 1ab' has a single jump from position 26779 to 26817, which is downstream of the ORF1ab gene and therefore will yield the complete polypeptide 1ab. Transcript 7b*, on the other hand, has a single discontinuous edge from position 71 to 27762. The start codon 'ATG' downstream of the 3' end occurs at position 27825, maintaining the frame of 7b, and thus leading to an N-terminal truncation [3] of 23 amino acids. Interestingly, transcript 7b and transcript 7b* appear with similar abundances in our solution. Finally, transcript N' has one canonical discontinuous edge from TRS-L (position 65) to the transcription regulating body sequence (TRS-B) region corresponding to ORF N (position 28255) and an additional jump from position 28525 to 28577, which leads to an in-frame deletion of 17 amino acids in the N-terminal RNA-binding domain [30, 31] of ORF N. Thus, with the exception of transcripts X and X', the non-canonical transcripts identified by JUMPER either produce complete viral proteins (1ab', S', 3a', 6'), contain in-frame deletions in the middle of known proteins (E', N') or produce N-terminally truncated proteins (7b*).

One of the major findings of the Kim *et al.* paper [3] is that the SARS-CoV-2 transcriptome is highly complex owing to numerous non-canonical discontinuous transcription events. Strikingly, our results show that these non-canonical transcription events do not significantly change the resulting proteins. Indeed, we find that 4 out of the 9 non-canonical transcripts produce a complete known viral protein and the total abundance of the predicted transcripts that produce a complete known viral protein is 0.968. Moreover, these predicted transcripts account for more than 90% of the reads in the sample according to the estimates provided by SALMON.

Typically, reads from short-read sequencing samples are not long enough to contain more than one discontinuous edge. As a result, short-read data can only provide direct evidence for transcripts with closely spaced discontinuous edges. For instance, we observe ample support (63485 short reads) for the predicted non-canonical transcript E', which has two discontinuous edges $(69, 26237)$ and $(26256, 26284)$, in short-read data due to the close proximity of the two discontinuous edges (*i.e.* the discontinuous edges are only $26256 - 26237 = 19$ nucleotides apart). The other non-canonical transcripts with multiple discontinuous edges, *i.e.* X', S', 3a', 6' and N', have edges that are too far apart to be spanned by a single short read. Using the long-read sequencing data of this sample, we detect supporting long reads that span the exact set of discontinuous edges of all 9 non-canonical transcripts (Fig. 4c). Moreover, we find support for the canonical transcripts as well (Fig. C8). Thus, all transcripts identified by JUMPER from the short-read data are supported by direct evidence in the long-read data.

In summary, using JUMPER we reconstructed a detailed picture of the transcriptome of a short-read sequencing sample of Vero cells infected by SARS-CoV-2. While existing methods failed to recall even the reference transcriptome, JUMPER identified transcripts encoding for all known viral protein products. In addition, our method predicted non-canonical transcripts and their abundances, whose presence we subsequently validated on a long-read sequencing sample of the same cells.

5.3 Transcript assembly in SARS-CoV-1

To show the generalizability of our method, we consider another coronavirus, SARS-CoV-1. We analyzed two published samples of human Calu-3 cells infected with SARS-CoV-1 [14], SRR1942956 and SRR1942957, with a median depth of $21{,}358$ and $20{,}991$, respectively. These two samples originate from the same SRA project ('PRJNA279442') whose metadata states that both samples were sequenced 24 hours after infection. We used `fastp` to trim the short reads (trimming parameter set to 10 nucleotides) and we aligned the resulting reads using `STAR` in two-pass mode. We ran JUMPER with the 35 most abundant discontinuous edges in the segment graph. Similarly to the previous analysis, we restrict our attention to

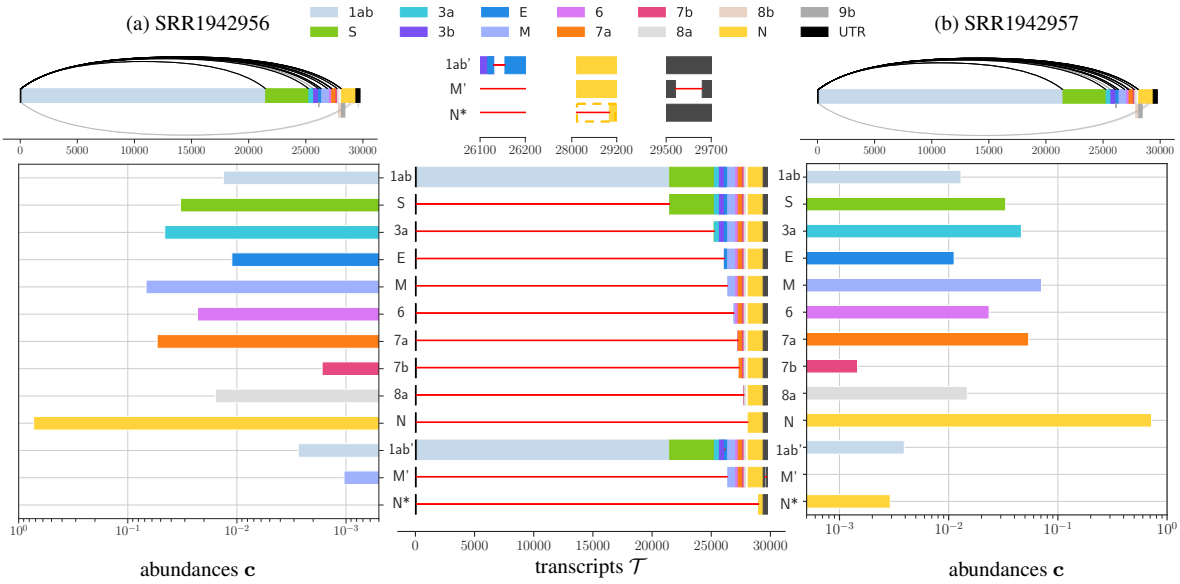


Figure 5: Using short-read data of SARS-CoV-1 infected Calu-3 cells [14], JUMPER identifies canonical and non-canonical transcripts consistently across two samples, (a) SRR1942956 and (b) SRR1942957. For both the samples, we show the segment graph, with canonical (above) and non-canonical (below) discontinuous edges. We also show the predicted transcripts and their abundances in the two samples with a zoomed-in view of the non-canonical transcripts 1ab', M' and N*. UTR: untranslated region.

transcripts identified by JUMPER that have more than 0.001 abundance as estimated by SALMON. There are 13 out of 25 such transcripts for sample SRR1942956 and 13 out of 26 such transcripts for sample SRR1942957 (Fig. 5).

SARS-CoV-1 has a genome of length 29751 bp, and consists of 13 ORFs (1ab, S, 3a, 3b, E, M, 6, 7a, 7b, 8a, 8b, N and 9b), two more than SARS-CoV-2. For both samples, JUMPER identifies canonical transcripts corresponding to all the ORFs of SARS-CoV-1 except ORF3b, ORF8b and ORF9b (Fig. 5). Notably, ORF8b and ORF9b share transcription regulating body sequences (TRS-B) with ORF8a and ORF N respectively [32]. More specifically, ORF9b (from position 28130 to 28426) is nested within ORF N (from position 28120 to 29388) with start codons only 10 nucleotides apart and consequently shares the same TRS-B as ORF N. ORF8b (from position 27864 to 28118) intersects with ORF8a (from 27779 to 27898) and previous studies have failed to validate a TRS-B region for ORF8b [32]. One possible way that these ORFs are translated is due to ribosome leaky scanning, which was also hypothesized to lead to ORF7b translation in SARS-CoV-2 [28]. This explains why JUMPER was unable to identify transcripts that directly encode for 8b and 9b. As for ORF3b, JUMPER did identify a canonical transcript corresponding to 3b in both samples, but the SALMON estimated abundances for these transcripts were below the cut-off value (0.00044 for SRR1942956 and 0.0005 for SRR1942957). Finally, we note that the relative abundances of the canonical transcripts are consistent for the two samples (Fig. 5), ranked in the same order (Fig. C9), with ORF7b being the least abundant and ORF N having the largest abundance, in line with the observations in SARS-CoV-2 (see Section 5.2).

Fig. 5 shows the three non-canonical transcripts predicted by JUMPER in the two SARS-CoV-1 samples, designated as 1ab', M' and N*. Since these non-canonical transcripts are in very low abundance, we see some discrepancy in the prediction between the two samples. The first non-canonical transcript 1ab' with a single short discontinuous edge from position 26131 to 26156 is detected in both samples and has a very low abundance compared to the canonical transcript 1ab (0.0133 vs 0.002 in SRR1942956, and 0.013 vs 0.0039 in SRR1942956). Since the discontinuous edge occurs downstream of the stop codon of 1ab (position 21492), the 1ab' transcript encodes for the complete polypeptide 1ab. The second non-canonical transcript M' has two discontinuous edges: a canonical discontinuous edge from TRS-L (position 65) to TRS-B of ORF M (position 26351) and a non-canonical discontinuous edge from 29542 to 29661 in the 3' untranslated region (UTR). As such, this transcript encodes for the complete M protein. This transcript is detected in SRR1942956 with a very low abundance of 0.001 and is detected at an even lower abundance of 0.0008 in SRR1942957, which is below the cut-off threshold of 0.001. The third non-canonical transcript, denoted by N*, has a single discontinuous edge from position 65 to 29003. While JUMPER and SALMON

detect this transcript only in sample SRR1942957 with a low abundance of 0.003, we do observe 119 reads in SRR1942956 (compared to 151 reads in SRR1942957) that support this edge, suggesting that N* might be present in the latter sample at too small of an abundance to be detected. Transcript N* is interesting because the first 'ATG' downstream of the 3' end of its discontinuous edge occurs at position 29071 maintaining the frame of N (which starts at position 28120). Thus transcript B* encodes for an N-terminally truncated version of protein N with 105 out the 422 amino acids that only contain part of the C-terminal dimerization domain [30]. This is similar to transcript 7b* in the SARS-CoV-2 sample (Section 5.2) which yields a N-terminal truncated version of protein 7b. Detection of non-canonical transcripts such as E' and 7b* in SARS-CoV-2 and N' in SARS-CoV-1 suggests that generation of N-terminally truncated proteins might be a common feature in coronaviruses. In summary, JUMPER can be applied to all coronaviruses to reconstruct the transcriptome from short-read sequencing data and be used to discover novel transcripts and the viral protein products that they encode.

6 Discussion

In this paper, we formulated the DISCONTINUOUS TRANSCRIPT ASSEMBLY (DTA) problem of reconstructing viral transcripts from short-read RNA-seq data for SARS-CoV-2 and, more generally, viruses in the order of *Nidovirales*. The discontinuous transcription process exhibited by the viral RNA-dependent RNA polymerase (RdRp) is distinct from alternative splicing observed in eukaryotes. Our proposed method, JUMPER, is specifically designed to reconstruct the viral transcripts generated by discontinuous transcription and is therefore able to outperform existing transcript assembly methods such as SCALLOP and STRINGTIE, as we have shown in both simulated and real data. For real-data analysis, we used publicly available short-read and long-read sequencing data of the same sample of virus infected Vero cells [3]. We performed transcript assembly using the short-read sequencing data and used the long-read data for validation. JUMPER was able to identify transcripts encoding for all known viral proteins except ORF10, which has been shown to have little support of active transcription in previous studies [3, 28]. Moreover, we predicted 9 non-canonical transcripts that are well supported by long-read sequencing data. For two samples of Calu-3 cells infected by SARS-CoV-1 [14], JUMPER reconstructed all the canonical transcripts with distinct TRS-B regions and additionally predicted the presence of non-canonical transcripts encoding for either complete or truncated versions of known viral proteins.

There are several avenues for future work. First, the computational complexity of the DTA problem remains open. Second, we plan to extend our current model to account for positional and sequencing biases

in the data. Doing so will enable us to assemble transcriptomes from sequencing samples that used SARS-CoV-2-specific primers, which form the majority of currently available data. Third, we currently make the assumption that the read fragments are much smaller compared to the viral transcripts. We will relax this assumption in order to support long-read sequencing data that have variable read length. Fourth, we plan to study the effect of mutations (including single-nucleotide variants as well as indels) on the transcriptome. Along the same lines, there is evidence of within-host diversity in COVID-19 patients [33–38]. It will be interesting to study whether this diversity translates to distinct sets of transcripts and abundances within the same host. Fifth, there are possibly multiple optimal solutions to the DTA problem that present equally likely viral transcripts with different relative abundances in the sample. A useful direction of future work is to explore the space of optimal solutions similar to the work done in [18]. Finally, the approach presented in this paper can extended to the general transcript assembly problem, where a topological ordering of the nodes in the splice graph will serve the same function as the unique Hamiltonian path of the segment graph did in the DTA problem. We envision this will facilitate efficient use of combinatorial optimization tools such as integer linear programming to transcript assembly problems.

Data availability. Simulated data is available at `https://doi.org/10.13012/B2IDB-6667667_V1`. The code is available at `https://github.com/elkebir-group/Jumper`.

Acknowledgments. The authors were supported by the National Science Foundation under award number CCF-2027669.

References

[1] Antoine AF De Vries, Marian C Horzinek, Peter JM Rottier, and Raoul J De Groot. The genome organization of the Nidovirales: similarities and differences between Arteri-, Toro-, and Coronaviruses. In *Seminars in VIROLOGY*, volume 8, pages 33–47. Elsevier, 1997.

[2] Helena Jane Maier, Erica Bickerton, Paul Britton, et al. *Coronaviruses: methods and protocols.* Springer Berlin, 2015.

[3] Dongwan Kim, Joo-Yeon Lee, Jeong-Sun Yang, Jun Won Kim, V Narry Kim, and Hyeshik Chang. The architecture of SARS-CoV-2 transcriptome. *Cell*, 2020.

[4] Gordon Robertson, Jacqueline Schein, Readman Chiu, Richard Corbett, Matthew Field, Shaun D Jackman, Karen Mungall, Sam Lee, Hisanaga Mark Okada, Jenny Q Qian, et al. De novo assembly and analysis of RNA-seq data. *Nature Methods*, 7(11):909–912, 2010.

[5] Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, et al. Trinity: reconstructing a full-length transcriptome without a genome from RNA-Seq data. *Nature Biotechnology*, 29(7):644, 2011.

[6] Yinlong Xie, Gengxiong Wu, Jingbo Tang, Ruibang Luo, Jordan Patterson, Shanlin Liu, Weihua Huang, Guangzhu He, Shengchang Gu, Shengkang Li, et al. SOAPdenovo-Trans: de novo transcriptome assembly with short RNA-Seq reads. *Bioinformatics*, 30(12):1660–1666, 2014.

[7] Zheng Chang, Guojun Li, Juntao Liu, Yu Zhang, Cody Ashby, Deli Liu, Carole L Cramer, and Xiuzhen Huang. Bridger: a new framework for de novo transcriptome assembly using RNA-seq data. *Genome Biology*, 16(1):30, 2015.

[8] Marcel H Schulz, Daniel R Zerbino, Martin Vingron, and Ewan Birney. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092, 2012.

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

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

[11] Juntao Liu, Ting Yu, Tao Jiang, and Guojun Li. TransComb: genome-guided transcriptome assembly via combing junctions in splicing graphs. *Genome Biology*, 17(1):213, 2016.

[12] Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J Van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 2010.

[13] Li Song and Liliana Florea. CLASS: constrained transcript assembly of RNA-seq reads. In *BMC Bioinformatics*, volume 14, page S14. Springer, 2013.

[14] Xi Zhang, Hin Chu, Lei Wen, Huiping Shuai, Dong Yang, Yixin Wang, Yuxin Hou, Zheng Zhu, Shuofeng Yuan, Feifei Yin, et al. Competing endogenous RNA network profiling reveals novel host dependency factors required for MERS-CoV propagation. *Emerging microbes & infections*, 9(1):733–746, 2020.

[15] Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

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

[17] Elsa Bernard, Laurent Jacob, Julien Mairal, Eric Viara, and Jean-Philippe Vert. A convex formulation for joint RNA isoform detection and quantification from multiple RNA-seq samples. *BMC bioinformatics*, 16(1):1–10, 2015.

[18] Cong Ma, Hongyu Zheng, and Carl Kingsford. Finding ranges of optimal transcript expression quantification in cases of non-identifiability. *BioRxiv*, pages 2019–12, 2020.

[19] 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.

[20] 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.

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

[22] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[23] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[24] Alyssa C Frazee, Andrew E Jaffe, Ben Langmead, and Jeffrey T Leek. Polyester: simulating RNA-seq datasets with differential transcript expression. *Bioinformatics*, 31(17):2778–2784, 2015.

[25] Daryl M Gohl, John Garbe, Patrick Grady, Jerry Daniel, Ray HB Watson, Benjamin Auch, Andrew Nelson, Sophia Yohe, and Kenneth B Beckman. A rapid, cost-effective tailed amplicon method for sequencing SARS-CoV-2. *BMC genomics*, 21(1):1–10, 2020.

[26] Josh Quick. nCoV-2019 sequencing protocol v3 (LoCost). *protocols.io*, 08 2020. https://protocols.io/view/ncov-2019-sequencing-protocol-v3-locost-bh42j8ye.

[27] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

[28] Yaara Finkel, Orel Mizrahi, Aharon Nachshon, Shira Weingarten-Gabbay, David Morgenstern, Yfat Yahalom-Ronen, Hadas Tamir, Hagit Achdout, Dana Stein, Ofir Israeli, et al. The coding capacity of sars-cov-2. *Nature*, 589(7840):125–130, 2021.

[29] Venkata S. Mandala, Matthew J. McKay, Alexander A. Shcherbakov, Aurelio J. Dregni, Antonios Kolocouris, and Mei Hong. Structure and drug binding of the SARS-CoV-2 envelope protein transmembrane domain in lipid bilayers. *Nature Structural & Molecular Biology*, 27(12):1202–1208, December 2020.

[30] Sisi Kang, Mei Yang, Zhongsi Hong, Liping Zhang, Zhaoxia Huang, Xiaoxue Chen, Suhua He, Ziliang Zhou, Zhechong Zhou, Qiuyue Chen, et al. Crystal structure of sars-cov-2 nucleocapsid protein rna binding domain reveals potential unique drug targeting sites. *Acta Pharmaceutica Sinica B*, 10(7):1228–1238, 2020.

[31] Qiaozhen Ye, Alan MV West, Steve Silletti, and Kevin D Corbett. Architecture and self-assembly of the sars-cov-2 nucleocapsid protein. *Protein Science*, 29(9):1890–1901, 2020.

[32] Yiyan Yang, Wei Yan, A Brantley Hall, and Xiaofang Jiang. Characterizing Transcriptional Regulatory Sequences in Coronaviruses and Their Role in Recombination. *Molecular Biology and Evolution*, 11 2020. msaa281.

[33] Palash Sashittal, Yunan Luo, Jian Peng, and Mohammed El-Kebir. Characterization of SARS-CoV-2 viral diversity within and across hosts. *bioRxiv*, 2020.

[34] Rebecca Rose, David J Nolan, Samual Moot, Amy Feehan, Sissy Cross, Julia Garcia-Diaz, and Susanna L Lamers. Intra-host site-specific polymorphisms of SARS-CoV-2 is consistent across multiple samples and methodologies. *medRxiv*, 2020.

[35] Daniele Ramazzotti, Fabrizio Angaroni, Davide Maspero, Carlo Gambacorti-Passerini, Marco Antoniotti, Alex Graudenzi, and Rocco Piazza. Characterization of intra-host SARS-CoV-2 variants improves phylogenomic reconstruction and may reveal functionally convergent mutations. *bioRxiv*, 2020.

[36] Zijie Shen, Yan Xiao, Lu Kang, Wentai Ma, Leisheng Shi, Li Zhang, Zhuo Zhou, Jing Yang, Jiaxin Zhong, Donghong Yang, et al. Genomic diversity of SARS-CoV-2 in coronavirus disease 2019 patients. *Clinical Infectious Diseases*, 2020.

[37] Timokratis Karamitros, Gethsimani Papadopoulou, Maria Bousali, Anastasios Mexias, Sotiris Tsiodras, and Andreas Mentis. SARS-CoV-2 exhibits intra-host genomic plasticity and low-frequency polymorphic quasispecies. *bioRxiv*, 2020.

[38] Xiaolu Tang, Changcheng Wu, Xiang Li, Yuhe Song, Xinmin Yao, Xinkai Wu, Yuange Duan, Hong Zhang, Yirong Wang, Zhaohui Qian, et al. On the origin and continuing evolution of SARS-CoV-2. *National Science Review*, 2020.

[39] Cong Ma, Hongyu Zheng, and Carl Kingsford. Exact transcript quantification over splice graphs. In *20th International Workshop on Algorithms in Bioinformatics (WABI 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[40] Jon Lee and Dan Wilson. Polyhedral methods for piecewise-linear functions I: the lambda method. *Discrete Applied Mathematics*, 108(3):269–285, 2001.

[41] Alyson Imamoto and Benjamim Tang. A recursive descent algorithm for finding the optimal minimax piecewise linear approximation of convex functions. In *Advances in Electrical and Electronics*

Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, pages 287–293. IEEE, 2008.

[42] Alexander O Pasternak, Willy JM Spaan, and Eric J Snijder. Nidovirus transcription: how to make sense...? *Journal of General Virology*, 87(6):1403–1421, 2006.

[43] Stanley G Sawicki and Dorothea L Sawicki. Coronaviruses use discontinuous extension for synthesis of subgenome-length negative strands. In *Corona-and Related Viruses*, pages 499–506. Springer, 1995.

[44] Guido Van Marle, Jessika C Dobbe, Alexander P Gultyaev, Willem Luytjes, Willy JM Spaan, and Eric J Snijder. Arterivirus discontinuous mRNA transcription is guided by base pairing between sense and antisense transcription-regulating sequences. *Proceedings of the National Academy of Sciences*, 96(21):12056–12061, 1999.

[45] Sonia Zuniga, Isabel Sola, Sara Alonso, and Luis Enjuanes. Sequence motifs involved in the regulation of discontinuous coronavirus subgenomic RNA synthesis. *Journal of Virology*, 78(2):980–994, 2004.

[46] Alexander O Pasternak, Erwin van den Born, Willy JM Spaan, and Eric J Snijder. Sequence requirements for RNA strand transfer during nidovirus discontinuous subgenomic RNA synthesis. *The EMBO Journal*, 20(24):7220–7228, 2001.

[47] Dorothea L Sawicki, Tao Wang, and Stanley G Sawicki. The RNA structures engaged in replication and transcription of the A59 strain of mouse hepatitis virus. *Journal of General Virology*, 82(2):385–396, 2001.

[48] Antoine AF de Vries, Amy L Glaser, Martin JB Raamsman, and Peter JM Rottier. Recombinant equine arteritis virus as an expression vector. *Virology*, 284(2):259–276, 2001.

A Likelihood Model for DISCONTINUOUS TRANSCRIPT ASSEMBLY

We use the segment graph G to compute the probability $\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c})$ of observing the alignment $\mathcal{R}$ given transcripts $\mathcal{T}$ and abundances $\mathbf{c}$. We follow the generative model described in [39], which has been extensively used for transcription quantification [19–21]. Let the set $\mathcal{R}$ of reads be $\{1, \ldots, r_n\}$ and the set $\mathcal{T}$ of transcripts be $\mathcal{T} = \{T_1, \ldots, T_k\}$ with lengths $L_1, \ldots, L_k$ and abundances $\mathbf{c} = [c_1, \ldots, c_k]$. In line with current literature, reads $\mathcal{R}$ are generated independently from transcripts $\mathcal{T}$ with abundances $\mathbf{c}$. Further, we must marginalize over the set of transcripts $\mathcal{T}$ as the transcript of origin of any given read is typically unknown, due to $\ell \ll L$. Thus,

$$\begin{aligned}\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c}) &= \prod_{j=1}^{n} \Pr(r_j \mid \mathcal{T}, \mathbf{c}) \\ &= \prod_{j=1}^{n} \sum_{i=1}^{k} \Pr(r_j, Z_{i,j} \mid \mathcal{T}, \mathbf{c}) \\ &= \prod_{j=1}^{n} \sum_{i=1}^{k} \Pr(r_j \mid Z_{i,j}) \Pr(Z_{i,j} \mid \mathcal{T}, \mathbf{c}),\end{aligned}$$

where $Z_{i,j}$ is the indicator random variable for the event that T_i is the transcript of origin for read r_j. We denote by $\Pr(r_j \mid Z_{i,j})$ the probability of observing read r_j given that it is generated from transcript T_i and $\Pr(Z_{i,j} \mid \mathcal{T}, \mathbf{c})$ denotes the probability of generating a read from transcript T_i given transcripts $\mathcal{T}$ and abundances $\mathbf{c}$.

Assuming no amplification and sequencing bias, the probability $\Pr(Z_{i,j} \mid \mathcal{T}, \mathbf{c})$ of generating a read from a transcript T_i of length L_i is given by

$$\Pr(Z_{i,j} \mid \mathcal{T}, \mathbf{c}) = \frac{c_i L_i}{\sum_{j=1}^{k} c_j L_j}.$$

We now derive the probability $\Pr(r_j \mid Z_{i,j})$ of transcript T_i generating read r_j of fixed length ℓ. We do so using the segment graph $G = (V, E)$. Recall that a transcript T must correspond to an $\mathbf{s}$ to $\mathbf{t}$ path in G. Let $\pi(T) \subseteq E$ denote the path corresponding to transcript T. Similarly, each read r induces a path $\pi(r) \subseteq E$ in G. Read r can only be generated by transcript T if $\pi(r) \subseteq \pi(T)$. Hence, the probability of transcript T_i generating a given read r_j is given by

$$\Pr(r_j \mid Z_{i,j}) = \begin{cases} 1/L_i', & \text{if } \pi(r_j) \subseteq \pi(T_i), \\ 0, & \text{otherwise,} \end{cases}$$

where $L_i' = L_i - \ell$ is the *effective length* of the transcript. We assume that the transcripts are much longer

than the reads and as such $L'_i/L_i \approx 1$. Putting it all together we get

$$\begin{aligned}
\Pr(\mathcal{R} \mid \mathcal{T}, c) &= \prod_{j=1}^{n} \sum_{i=1}^{k} \Pr(r_j \mid Z_{i,j}) \Pr(Z_{i,j} \mid \mathcal{T}, \mathbf{c}) \\
&= \prod_{j=1}^{n} \sum_{i=1}^{k} \frac{\mathbf{1}\{\pi(r_j) \subseteq \pi(T_i)\}}{L'_i} \cdot \frac{c_i L_i}{\sum_{b=0}^{k} c_b L_b} \\
&= \prod_{j=1}^{n} \sum_{i:\pi(T_i) \supseteq \pi(r_j)} \frac{1}{L'_i} \cdot \frac{c_i L_i}{\sum_{b=0}^{k} c_b L_b} \\
&= \prod_{j=1}^{n} \frac{1}{\sum_{b=1}^{k} c_b L_b} \sum_{i:\pi(T_i) \supseteq \pi(r_j)} c_i \frac{L_i}{L'_i} \\
&= \prod_{j=1}^{n} \frac{1}{\sum_{b=1}^{k} c_b L_b} \sum_{i:\pi(T_i) \supseteq \pi(r_j)} c_i.
\end{aligned}$$

B Supplementary Methods

B.1 Recharacterization of solutions using discontinuous edges

We prove the following two main text propositions.

(Main Text) Proposition 1. There is a bijection between subsets of discontinuous edges that are pairwise non-overlapping and $\mathbf{s} - \mathbf{t}$ paths in G.

Proof. Let Π be the set of $\mathbf{s} - \mathbf{t}$ paths in G. We indicate with Σ the family of subsets of discontinuous edges that are pairwise non-overlapping. Note that $\Sigma \subseteq 2^{E^{\frown}}$.

For an $\mathbf{s} - \mathbf{t}$ path $\pi \in \Pi$, let $f(\pi)$ be the set of discontinuous edges in π, *i.e.* $f(\pi) = \pi \cap E^{\frown}$. Since π is an $\mathbf{s} - \mathbf{t}$ path of G, we have that for each edge $(\mathbf{v} = [v^-, v^+], \mathbf{w} = [w^-, w^+]) \in \pi$ it holds that $v^+ \leq w^-$. Therefore, $f(\pi)$ is composed of pairwise non-overlapping disconnected edges.

Now, consider a subset $\sigma \in \Sigma$ of discontinuous edges that are pairwise non-overlapping. We obtain the corresponding $\mathbf{s} - \mathbf{t}$ path $f^{-1}(\sigma)$ by first ordering the edges of σ in ascending order. That is, let $\sigma = \{(\mathbf{v}_1 = [v_1^-, v_1^+], \mathbf{w}_1 = [w_1^-, w_1^+]), \ldots, (\mathbf{v}_{|\sigma|} = [v_{|\sigma|}^-, v_{|\sigma|}^+], \mathbf{w}_{|\sigma|} = [w_{|\sigma|}^-, w_{|\sigma|}^+])\}$ such that $w_i^+ \leq v_{i+1}^-$ for all $i \in \{1, \ldots, |\sigma|-1\}$. For every two consecutive discontinuous edges $(\mathbf{v}_i = [v_i^-, v_i^+], \mathbf{w}_i = [w_i^-, w_i^+])$ and $(\mathbf{v}_{i+1} = [v_{i+1}^-, v_{i+1}^+], \mathbf{w}_{i+1} = [w_{i+1}^-, w_{i+1}^+])$, we include the corresponding subpath of continuous edges from $\mathbf{w}_i$ to $\mathbf{v}_{i+1}$ into $f^{-1}(\sigma)$. In addition, we include the subpath of continuous edges from node $\mathbf{s}$ to node $\mathbf{v}_1$ as well as the subpath from node $\mathbf{w}_{|\sigma|}$ to $\mathbf{t}$ into $f^{-1}(\sigma)$. By construction, $f^{-1}(\sigma)$ is an $\mathbf{s} - \mathbf{t}$ path. □

(Main Text) Proposition 2. Let G be a segment graph, T be a transcript and r be a read. Then, $\pi(T) \supseteq \pi(r)$ if and only if $\sigma(T) \supseteq \sigma^{\oplus}(r)$ and $\sigma(T) \cap \sigma^{\ominus}(r) = \emptyset$.

Proof. **(⇒)** By the premise, $\pi(T) \supseteq \pi(r)$. By definition, $\sigma(T) = \pi(T) \cap E^{\curvearrowright}$. By Definition 4, $\sigma^{\oplus}(r) = \pi(r) \cap E^{\curvearrowright}$. As $\pi(T) \supseteq \pi(r)$, we have that $\sigma(T) = \pi(T) \cap E^{\curvearrowright} \supseteq \pi(r) \cap E^{\curvearrowright} = \sigma^{\oplus}(r)$. By definition, $\sigma^{\ominus}(r)$ is the subset of discontinuous edges in $E^{\curvearrowright} \setminus \sigma^{\oplus}(r)$ that overlaps with an edge in $\pi(r)$. Since $\pi(T) \supseteq \pi(r)$, every edge included in $\sigma^{\ominus}(r)$ because of an overlap with an edge in $\pi(r)$ must also overlap with the same edge in $\pi(T)$. Since $\pi(T)$ is an $\mathbf{s}-\mathbf{t}$ path, and thus does not contain pairwise overlapping edges, we infer that $\sigma^{\ominus}(r) \cap \sigma(T) = \emptyset$.

(⇐) By the premise, $\sigma(T) \supseteq \sigma^{\oplus}(r)$ and $\sigma(T) \cap \sigma^{\ominus}(r) = \emptyset$. As $\sigma(T) \supseteq \sigma^{\oplus}(r)$, we have that $\pi(T) \cap E^{\curvearrowright} = \sigma(T) \supseteq \sigma^{\oplus}(r) = \pi(r) \cap E^{\curvearrowright}$. Since $\sigma(T) \cap \sigma^{\ominus}(r) = \emptyset$, we have by Definition 4, that no discontinuous edge in $\sigma(T)$ overlaps with any edge in $\pi(r)$. Since $\pi(T)$ is an $\mathbf{s}-\mathbf{t}$ path containing the subset $\sigma^{\oplus}(r)$ of discontinuous edges in $\pi(r)$, it holds that $\pi(T) \cap E^{\rightarrow} \supseteq \pi(r) \cap E^{\rightarrow}$. Finally, as $E^{\curvearrowright} \cup E^{\rightarrow} = E$, $\pi(r) \subseteq E$ and $\pi(T) \subseteq E$, we get $\pi(T) \supseteq \pi(r)$. □

Using this proposition, we derive a simpler form of the likelihood given in Equation (2). Let $\mathcal{S} = \{(\sigma_1^{\oplus}, \sigma_1^{\ominus}), \ldots, (\sigma_m^{\oplus}, \sigma_m^{\ominus})\}$ be the set of characteristic discontinuous edges generated by the reads in alignment $\mathcal{R}$. Let $\mathbf{d} = \{d_1, \cdots, d_m\}$ be the number of reads that map to each pair in $\mathcal{S}$. Using that distinct reads r_j and $r_{j'}$ with the same characteristic discontinuous edges $(\sigma^{\oplus}(r_j), \sigma^{\ominus}(r_j)) = (\sigma^{\oplus}(r_{j'}), \sigma^{\ominus}(r_{j'}))$ have the same likelihood in terms of (2), we have

$$\Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c}) = \prod_{j=1}^{n} \frac{1}{\sum_{b=1}^{k} c_b L_b} \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i = \prod_{j=1}^{m} \left(\frac{1}{\sum_{b=1}^{k} c_b L_b} \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i \right)^{d_j}. \tag{12}$$

Now, taking the logarithm yields

$$\begin{aligned}
\log \Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c}) &= \sum_{j=1}^{m} d_j \left(\log \left(\frac{1}{\sum_{b=1}^{k} c_b L_b} \right) + \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i \right) \\
&= -\sum_{j=1}^{m} d_j \left(\log \sum_{b=1}^{k} c_b L_b \right) + \sum_{j=1}^{m} \left(d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i \right) \\
&= \sum_{j=1}^{m} \left(d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i \right) - n \log \sum_{b=1}^{k} c_b L_b.
\end{aligned} \tag{13}$$

The goal is the remove the second sum in the above equation, as it is convex and we are maximizing. In order to do so, we first prove the following lemma.

Lemma 2. For any given scaling factor $\alpha > 0$, we have that $\log \Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c}) = \log \Pr(\mathcal{R} \mid \mathcal{T}, \alpha\mathbf{c})$.

Proof.

$$\begin{aligned}
\log \Pr(\mathcal{R} \mid \mathcal{T}, \alpha\mathbf{c}) &= \sum_{j=1}^{m} \left(d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} \alpha c_i \right) - n \log \sum_{b=1}^{k} \alpha c_b L_b \\
&= \sum_{j=1}^{m} \left(d_j \log \left(\alpha \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i \right) \right) - n \log \alpha \sum_{b=1}^{k} c_b L_b \\
&= \sum_{j=1}^{m} d_j \log \alpha + \sum_{j=1}^{m} \left(d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i \right) - n \log \alpha - n \log \sum_{b=1}^{k} c_b L_b \\
&= n \log \alpha + \sum_{j=1}^{m} \left(d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i \right) - n \log \alpha - n \log \sum_{b=1}^{k} c_b L_b \\
&= \sum_{j=1}^{m} \left(d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i \right) - n \log \sum_{b=1}^{k} c_b L_b \\
&= \log \Pr(\mathcal{R} \mid \mathcal{T}, \mathbf{c}).
\end{aligned}$$

□

This enables us to prove the following lemma.

(Main Text) Lemma 1. Let $D > 0$ be a constant, $\overline{c}_i(\mathbf{c}) = c_i D / \sum_{j=1}^{k} c_j L_j$ and $c_i(\overline{\mathbf{c}}) = \overline{c}_i / \sum_{j=1}^{k} \overline{c}_j$ for all $i \in [k]$. Then, $(\mathcal{T}, \mathbf{c} = [c_1(\overline{\mathbf{c}}), \ldots, c_k(\overline{\mathbf{c}})])$ is an optimal solution for (3)-(6) if and only if $(\mathcal{T}, \overline{\mathbf{c}} = [\overline{c}_1(\mathbf{c}), \ldots, \overline{c}_k(\mathbf{c})])$ is an optimal solution for

$$\max_{\mathcal{T}, \overline{\mathbf{c}}} \sum_{j=1}^{m} d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} \overline{c}_i \tag{14}$$

$$\text{s.t. } \pi(T_i) \text{ is an } \mathbf{s} - \mathbf{t} \text{ path in the segment graph } G \qquad \forall i \in [k], \tag{15}$$

$$\sum_{i=1}^{k} \overline{c}_i L_i = D, \tag{16}$$

$$\overline{c}_i \geq 0 \qquad \forall i \in [k]. \tag{17}$$

Proof. We will refer to the optimization problem in (3)-(6) as P and the optimization problem (14)-(17) as Q. Further, we will refer to the objective function in (3) as $J(\mathcal{T}, \mathbf{c})$ and the objective function in (14) as

$K(\mathcal{T}, \overline{\mathbf{c}})$. Observe that

$$\begin{aligned} K(\mathcal{T}, \overline{\mathbf{c}}) =& \log \Pr(\mathcal{R} \mid \mathcal{T}, \overline{\mathbf{c}}) + n \log \sum_{b=1}^{k} \overline{c}_b L_b \\ =& J(\mathcal{T}, \overline{\mathbf{c}}) + n \log \sum_{b=1}^{k} \overline{c}_b L_b, \end{aligned} \tag{18}$$

where the last equality uses (13).

(⇒) Let $(\mathcal{T}, \mathbf{c})$ be an optimal solution to problem P. We begin by showing that $(\mathcal{T}, \overline{\mathbf{c}})$ is a feasible solution to Q where $\overline{\mathbf{c}} = [\overline{c}_1(\mathbf{c}), \ldots, \overline{c}_k(\mathbf{c})]$. By definition of $\overline{c}_i(\mathbf{c})$, constraints (16) are satisfied. Hence, $(\mathcal{T}, \overline{\mathbf{c}})$ is a feasible solution to problem Q.

We now show that if $(\mathcal{T}, \mathbf{c})$ is an optimal solution to problem P, then $(\mathcal{T}, \overline{\mathbf{c}})$ is an optimal solution to problem Q. Let $(\mathcal{T}', \overline{\mathbf{c}}')$ be an optimal solution to problem Q. Then, by optimality of $(\mathcal{T}', \overline{\mathbf{c}}')$, we have

$$K(\mathcal{T}', \overline{\mathbf{c}}') \geq K(\mathcal{T}, \overline{\mathbf{c}}). \tag{19}$$

Let $\mathbf{c}' = [c_1(\overline{\mathbf{c}}'), \ldots, c_k(\overline{\mathbf{c}}')]$. Note that $\mathbf{c}'$ satisfies constraints (5). Thus $(\mathcal{T}', \mathbf{c}')$ is a feasible solution to problem P. Since $(\mathcal{T}, \mathbf{c})$ is an optimal solution of P, we have

$$J(\mathcal{T}, \mathbf{c}) \geq J(\mathcal{T}', \mathbf{c}'). \tag{20}$$

Since $\mathbf{c}'$ and $\overline{\mathbf{c}}'$ only differ by a positive scaling factor $\alpha = 1/\sum_{i=1}^{k} \overline{c}'_i$, we use Lemma 2 to get $J(\mathcal{T}', \mathbf{c}') = J(\mathcal{T}', \overline{\mathbf{c}}')$. Similar result holds for $\mathbf{c}$ and $\overline{\mathbf{c}}$, *i.e.* $J(\mathcal{T}, \mathbf{c}) = J(\mathcal{T}, \overline{\mathbf{c}})$. Applying this to (20), we get

$$J(\mathcal{T}, \overline{\mathbf{c}}) \geq J(\mathcal{T}', \overline{c}').$$

Using (16) and (18), we get

$$\begin{aligned} & J(\mathcal{T}, \overline{\mathbf{c}}) && \geq J(\mathcal{T}', \overline{\mathbf{c}}') \\ \implies & K(\mathcal{T}, \overline{\mathbf{c}}) - n \log \sum_{b=1}^{k} \overline{c}_b L_b && \geq K(\mathcal{T}', \overline{\mathbf{c}}') - n \log \sum_{b=1}^{k} \overline{c}'_b L_b \\ \implies & K(\mathcal{T}, \overline{\mathbf{c}}) - n \log D && \geq K(\mathcal{T}', \overline{\mathbf{c}}') - n \log D \\ \implies & K(\mathcal{T}, \overline{\mathbf{c}}) && \geq K(\mathcal{T}', \overline{\mathbf{c}}'). \end{aligned} \tag{21}$$

Finally, using (19) and (21), we get $K(\mathcal{T}, \overline{\mathbf{c}}) = K(\mathcal{T}', \overline{\mathbf{c}}')$. Hence, $(\mathcal{T}, \overline{\mathbf{c}})$ is an optimal solution of Q.

(⇐) Let $(\mathcal{T}, \overline{\mathbf{c}})$ be an optimal solution to problem Q. We begin by showing that $(\mathcal{T}, \mathbf{c})$ is a feasible solution to P where $\mathbf{c} = [c_1(\overline{\mathbf{c}}), \ldots, c_k(\overline{\mathbf{c}})]$. By definition of $c_i(\overline{\mathbf{c}})$, constraints (5) are satisfied. Hence, $(\mathcal{T}, \mathbf{c})$ is a feasible solution to problem P.

Next, we need to show that $(\mathcal{T}, \mathbf{c})$ is an optimal solution to problem P. Let $(\mathcal{T}', \mathbf{c}')$ be an optimal solution to problem P.

Then, from the optimality condition, we get

$$J(\mathcal{T}', \mathbf{c}') \geq J(\mathcal{T}, \mathbf{c}). \tag{22}$$

Let $\overline{\mathbf{c}}' = [\overline{c}_1(\mathbf{c}'), \ldots, \overline{c}_k(\mathbf{c}')]$. Note that $\overline{\mathbf{c}}'$ satisties constraint (16) and thus $(\mathcal{T}', \overline{\mathbf{c}}')$ is a feasible solution to problem Q. Using (18) and the fact that $(\mathcal{T}, \overline{\mathbf{c}})$ is an optimal solution of problem $\overline{P}$ we get

$$\begin{aligned}
& K(\mathcal{T}, \overline{\mathbf{c}}) && \geq K(\mathcal{T}', \overline{\mathbf{c}}') \\
\implies & J(\mathcal{T}, \overline{\mathbf{c}}) + n \log \sum_{b=1}^{k} \overline{c}_b L_b && \geq J(\mathcal{T}', \overline{\mathbf{c}}') + n \log \sum_{b=1}^{k} \overline{c}'_b L_b \\
\implies & J(\mathcal{T}, \overline{\mathbf{c}}) + n \log D && \geq J(\mathcal{T}', \overline{\mathbf{c}}') + n \log D \\
\implies & J(\mathcal{T}, \overline{\mathbf{c}}) && \geq J(\mathcal{T}', \overline{\mathbf{c}}').
\end{aligned} \tag{23}$$

Observe that $\mathbf{c}'$ and $\overline{\mathbf{c}}'$ only differ by a positive scaling factor $\alpha = D / \sum_{j=1}^{k} c'_j L_j$. Therefore, using Lemma 2, we have $J(\mathcal{T}', \mathbf{c}') = J(\mathcal{T}', \overline{\mathbf{c}}')$. Similarly, for $\mathbf{c}$ and $\overline{\mathbf{c}}$, we have $J(\mathcal{T}, \mathbf{c}) = J(\mathcal{T}, \overline{\mathbf{c}})$. Using this together with (23), we obtain

$$J(\mathcal{T}, \mathbf{c}) \geq J(\mathcal{T}', \mathbf{c}'). \tag{24}$$

Moreover, (22) and (24) simultaneously imply $J(\mathcal{T}, \mathbf{c}) = J(\mathcal{T}', \mathbf{c}')$. Hence, $(\mathcal{T}, \mathbf{c})$ is an optimal solution to problem P.

□

B.2 Mixed integer linear program

In the following, we introduce variables and constraints to encode the following.

(i) The composition of each transcript T_i as a set $\sigma(T_i)$ of non-overlapping discontinuous edges.

(ii) The abundance c_i and length L_i of each transcript T_i.

(iii) The total abundance $\sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i$ of transcripts supported by characteristic discontinuous edges $(\sigma_j^{\oplus}, \sigma_j^{\ominus})$.

(iv) A piecewise linear approximation of the $\log$ function.

We describe (iii) and (iv) in the following and refer to Section 4 for (i) and (ii).

Contribution of transcripts to each pair of characteristic discontinuous edges. The objective function has m terms, one corresponding to each pair $(\sigma_j^{\oplus}, \sigma_j^{\ominus}) \in \mathcal{S}$ of characteristic discontinuous edges (see Eq. (7)). Specifically, each term j equals $d_j \log \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i$ where d_j is a constant, for all $j \in [m]$. We introduce non-negative continuous variables $\mathbf{q} = \{q_1, \ldots, q_m\}$ such that

$$q_j = \sum_{i \in X(\mathcal{T}, \sigma_j^{\oplus}, \sigma_j^{\ominus})} c_i = \sum_{i=1}^{k} \left(c_i \prod_{e \in \sigma_j^{\oplus}} x_{e,i} \prod_{e' \in \sigma_j^{\ominus}} x_{e',i} \right), \tag{25}$$

where the last equality uses the characterization of candidate transcripts of origin for a given read described in Proposition 2. We introduce continuous variables $\mathbf{y}_j \in [0,1]^k$ that encode the product $y_{j,i} = c_i \prod_{e \in \sigma_j^{\oplus}} x_{e,i} \prod_{e' \in \sigma_j^{\ominus}} x_{e',i}$. Intuitively, each variable $y_{j,i}$ encodes the contribution of a transcript T_i for the given characteristic discontinuous edge sets $(\sigma_j^{\oplus}, \sigma_j^{\ominus})$. We linearize the product $c_i \prod_{e \in \sigma_j^{\oplus}} x_{e,i} \prod_{e' \in \sigma_j^{\ominus}} x_{e',i}$ as follows.

$$\begin{aligned}
y_{j,i} &\leq c_i, && \forall i \in [k], j \in [m], \\
y_{j,i} &\leq x_{e,i}, && \forall e \in \sigma_i^{\oplus}, i \in [k], j \in [m], \\
y_{j,i} &\leq 1 - x_{e,i}, && \forall e \in \sigma_i^{\ominus}, i \in [k], j \in [m], \\
y_{j,i} &\geq c_i + \sum_{e \in \sigma_j^{\oplus}} x_{e,i} + \sum_{e \in \sigma_j^{\ominus}} (1 - x_{e,i}) - |\sigma_j^{\oplus}| - |\sigma_j^{\ominus}|, && \forall i \in [k], j \in [m].
\end{aligned}$$

Hence, we have

$$q_j = \sum_{i=1}^{k} y_{j,i}. \tag{26}$$

Objective function. The objective function (Eq. (7)) can be written in terms of continuous variables $\mathbf{q}$ as

$$J(\mathbf{q}) = \sum_{j=1}^{m} d_j \log q_j,$$

where d_j is a constant and $\mathbf{q}$ is as in (26). We use the lambda method to approximate our objective method using a piecewise linear function [40]. Following the method described in [40], we partition the domain $(0,1]$ with h breakpoints $b_1 \leq b_2 \leq \ldots \leq b_h$. We introduce continuous variables $\boldsymbol{\lambda}_j \in [0,1]^h$ with the constraints

$$\begin{aligned}
\sum_{o=1}^{h} \lambda_{j,o} &= 1, && \forall j \in [m], \\
\sum_{o=1}^{h} b_o \lambda_{j,o} &= q_j, && \forall j \in [m].
\end{aligned}$$

Note that b_o for $o \in [h]$ are constants. Since each of the m terms in the objective function are individually concave and we are maximizing, the adjacency condition of breakpoints does not need to be enforced. For each $j \in [m]$, the $\log$ function is then approximated as

$$\log(q_j) \approx \sum_{o=1}^{h} \lambda_{j,o} \log(b_o),$$

where $\log(b_o)$ is a constant for each $o \in [h]$. Therefore the objective function we wish to maximize is

$$\sum_{j=1}^{m} d_j \sum_{o=1}^{h} \lambda_{j,o} \log(b_o).$$

Note that since we have a log-likelihood objective function, feasibility of the solution requires that $q_j > 0$ for $j \in [m]$. This means that for each characteristic discontinuous edge sets $(\sigma_j^{\oplus}, \sigma_j^{\ominus})$, there must be at least one candidate transcript of origin T_i with non-zero abundance $c_i > 0$. This leads to the solution containing a large number of transcripts and making the problem intractable while also preventing us from finding parsimonious sets of transcripts that support most but not all of the observed reads in the sample. Finding such parsimonious solutions is often desirable since they provide a reasonable explanation of the observed reads while keeping the problem computationally tractable. In order to allow us to generate solutions that can partially explain the observed reads, we slightly modify our objective function. We introduce a new breakpoint $b_0 = 0$ and associated continuous variables $\lambda_{j,0} \in [0, 1]$ for $j \in [m]$ so that

$$\sum_{o=0}^{h} \lambda_{j,o} = 1, \quad \forall j \in [m],$$

$$\sum_{o=0}^{h} b_o \lambda_{j,o} = q_j, \quad \forall j \in [m].$$

The objective function we maximize is

$$\sum_{j=1}^{m} d_j \left(\lambda_{j,0} \log(\delta) + \sum_{o=1}^{h} \lambda_{j,o} \log(b_o) \right),$$

where $\delta > 0$ is a small constant. Note that instead of evaluating the $\log$ function at b_0, we include $\log(\delta)$ which is well defined since $\delta > 0$. In this study, we choose $\delta = b_1/100 = 1/(2^{h-1} \times 100)$ while h is left as the user's choice with default value of 16.

Moreover, the choice of breakpoints to approximate the objective function (Eq. 7) can have a significant impact on the accuracy of the MILP solver. As a result, there has been research in efficient methods for choosing optimal breakpoint locations for convex functions, such as recursive descent algorithms [41]. In this work we take a simpler approach, by choosing breakpoints such that their spacing around a given breakpoint is proportional to the local gradient of the objective function. For the log function, this is equivalent to choosing breakpoints such that $b_i = 2^{i-1}/2^{h-1}$. Note that $b_0 = 1/2^{h-1}$ while $b_h = 1$.

Number of variables and constraints. The total number of binary variables $\mathbf{x}$ is $|E^{\curvearrowright}|k$. Note that $\mathbf{q}$ are auxiliary (intermediate) variables that are uniquely determined by $\mathbf{c}, \mathbf{y}, \mathbf{z}$ and $\boldsymbol{\lambda}$. Therefore, the total number of required continuous variables (*i.e.* $\mathbf{c}$, $\mathbf{y}$, $\mathbf{z}$ and $\boldsymbol{\lambda}$) is $k + mk + |E^{\curvearrowright}|k + mh$. The number of constraints is $O(k|E|^2 + |E|km)$. We provide the full MILP for reference.

$$\max \sum_{j=1}^{m} d_j \sum_{o=1}^{h} \lambda_{j,o} \log(b_o)$$

$$\begin{aligned}
\text{s.t. } & x_{e,i} + x_{e',i} \le 1, && \forall i \in [k] \text{ and } e, e' \in E^{\curvearrowright}, \\
& && \text{s.t. } I(e) \cap I(e') \neq \emptyset, \\
& y_{j,i} \le c_i, && \forall i \in [k], j \in [m], \\
& y_{j,i} \le x_{e,i}, && \forall e \in \sigma_j^{\oplus}, i \in [k], j \in [m], \\
& y_{j,i} \le 1 - x_{e,i}, && \forall e \in \sigma_j^{\ominus}, i \in [k], j \in [m], \\
& y_{j,i} \ge c_i + \sum_{e \in \sigma_j^{\oplus}} x_{e,i} + \sum_{e \in \sigma_j^{\ominus}} (1 - x_{e,i}) - |\sigma_j^{\oplus}| - |\sigma_j^{\ominus}|, && \forall i \in [k], j \in [m], \\
& z_{e,i} \le c_i, && \forall i \in [k], \\
& z_{e,i} \le x_{e,i}, && \forall e \in E^{\curvearrowright}, i \in [k], \\
& z_{e,i} \ge c_i + x_{e,i} - 1, && \forall e \in E^{\curvearrowright}, i \in [k], \\
& \sum_{i=1}^{k} c_i L - \sum_{i=1}^{k} \sum_{e \in E^{\curvearrowright}} z_{e,i} L(e) = \ell^*, && \\
& \sum_{o=1}^{h} \lambda_{j,o} = 1, && \forall j \in [m], \\
& \sum_{o=1}^{h} b_o \lambda_{j,o} = \sum_{i=1}^{k} y_{j,i}, && \forall j \in [m], \\
& x_{e,i} \in \{0, 1\}, && \forall i \in [k], e \in E^{\curvearrowright}, \\
& c_i \ge 0, && \forall i \in [k], \\
& y_{j,i} \ge 0, && \forall j \in [m], i \in [k], \\
& z_{e,i} \ge 0 && \forall e \in E^{\curvearrowright}, i \in [k], \\
& \lambda_{j,o} \ge 0, && \forall j \in [m], o \in [h].
\end{aligned}$$

B.3 JUMPER: progressive heuristic for the DTA problem

Here we describe the subproblems that are solved at each iteration of the greedy heuristic. For a given set of transcripts $\mathcal{T}$ and characteristic discontinuous edge sets $\mathcal{S}$, consider the optimization problem which we denote by P_1,

$$\max_{T',\mathbf{c},c'} \sum_{j=1}^{m} d_j \log \left(\sum_{i \in X(\mathcal{T},\sigma_j^{\oplus},\sigma_j^{\ominus})} c_i + \mathbf{1}(X(\{T'\},\sigma_j^{\oplus},\sigma_j^{\ominus})) \neq \emptyset) c' \right) \tag{27}$$

$$\text{s.t. } \pi(T') \text{ is an } \mathbf{s}-\mathbf{t} \text{ path in the segment graph } G \tag{28}$$

$$\sum_{i=1}^{|\mathcal{T}|} \bar{c}_i L_i + c' L' = D, \tag{29}$$

$$c_i \geq 0 \qquad \forall i \in [|\mathcal{T}|] \tag{30}$$

$$c' \geq 0 \qquad . \tag{31}$$

and the following optimization problem denoted by P_2,

$$\max_{\mathbf{c}} \sum_{j=1}^{m} d_j \log \sum_{i \in X(\mathcal{T},\sigma_j^{\oplus},\sigma_j^{\ominus})} \bar{c}_i \tag{32}$$

$$\sum_{i=1}^{|T|} c_i L_i = D, \tag{33}$$

$$c_i \geq 0 \qquad \forall i \in [|\mathcal{T}|]. \tag{34}$$

Solution to P_1 We obtain the solution of P_1 by solving the optimization problem (7)-(10) with additional constraints to fix the values of the variables that encode the presence/absence of discontinuous edges for the transcripts in $\mathcal{T}$. More specifically, for each transcript $T_i \in \mathcal{T}$, we enforce $x_{e,i} = 1$ for each edge $e \in \sigma(T_i)$ and $x_{e,i} = 0$ otherwise. Note that c_i for $T_i \in \mathcal{T}$ are still variables and are solved for in the optimization problem. By doing so, we only solve for the structure of the transcript T' while solving for the abundance of all transcripts.

Solution to P_2 Similar to the approach taken to solve P_1, we fix the values of the variables that encode the presence/absence of discontinuous edges in the transcripts. This results in all the binary variables in the MILP with fixed values rendering the resulting optimization problem a simpler linear program.

Heuristic Algorithm The Algorithm 1 is re-written here in form of an itemized list.

1. Initialize $\mathcal{T} = \{\}, i = 1$
2. Solve P_1 with $\mathcal{T}$ to get a new transcript T' with abundance c'
3. Generate a new set of transcripts $\mathcal{T} \leftarrow \mathcal{T} \cup \text{EXPAND}(T')$ where $\text{EXPAND}(T') = \{T : \sigma(T) \in 2^{\sigma(T')}\}$.
4. Solve P_2 with $\mathcal{T}$ as input
5. Select i transcripts from $\mathcal{T}$. If $i < k$ go to step (2) else return $(\mathcal{T}, \mathbf{c})$

B.4 Filtering false positive discontinuous edges

In practice, we see spurious discontinuous edges in the resulting segment graph due to sequencing and alignment errors. We filter these edges by requiring a minimum number Λ of spliced reads to support each discontinuous edge in the segment graph. The higher the value of Λ, fewer will be the number of edges and nodes in the resulting segment graph.

It is not trivial to know the right value of Λ to remove all false positive discontinuous edges. Several heuristics are used in existing methods to remove spurious splicing events. SCALLOP removes an edge e from its splice graph if the coverage of the exons of either end of the edge is more than $2w(e)^2 + 18$, where $w(e)$ is the number of spliced reads that support the edge e. STRINGTIE on the other hand, terminates its algorithm of assembling transcripts when the coverage of all the paths in the splice graph build from the un-assigned reads drops below a threshold, set by default to 2.5 reads per base-pair. By default, JUMPER requires a support of 100 reads for a discontinuous edge to be included in the segment graph.

Another parameter that can be used to filter false-positive splicing events is the number of discontinuous edges allowed in the segment graph. From tests on simulated instances emulating SARS-CoV-2 samples, we found that focusing on the 35 most abundant discontinuous edges is sufficient to get a summary of the transcriptome and highly expressed canonical and non-canonical transcripts in the sample. A higher value can be used to capture more complexity of the transcriptome. By default, we set this parameter to 35.

C Supplementary Results

C.1 Simulation pipeline

Our simulations are based on a widely believed model of discontinuous transcription. Briefly, there are two competing models of discontinuous transcription for coronaviruses [42]. Both models agree that the RdRP jump is mediated by matching core-sequences (motifs) present in the TRSs in the viral genome. The only

point of difference between the two models is whether discontinuous transcription occurs during the plus-strand synthesis or the minus-strand synthesis. The *negative-sense discontinuous transcription model* [43] proposes that the it is during the minus-strand synthesis that the RdRP performs discontinuous transcription. Transcription is initiated at the 3' end of the plus-strand RNA and the RdRP jumps to the TRS-L region when it reaches a TRS-B region adjacent to a gene, thereby generating a minus-strand subgenomic RNA. The minus-strand subgenomic RNA is then replicated by the RdRP to produce a plus-strand RNA which can be translated to a viral protein. Currently, this model is largely believed to be true due to the considerable experimental support from genetic studies detecting minus-strand subgenomic RNAs [44–48].

We now describe the procedure to simulate transcripts and their abundances following the negative-sense model of discontinuous transcription for a given segment graph. The model is parameterized by the function $p : E \to [0, 1]$. According to the *negative-sense discontinuous transcription* model, the transcription process is modeled as an $\mathbf{t} - \mathbf{s}$ walk in the reverse graph $\bar{G}$ where the direction of each original edge is reversed. At each node the RdRP randomly chooses an outgoing edge to traverse in the reverse graph $\bar{G}$ (which would be an incoming edge to the node in the original graph G) where the probabilities are given by the function p. Hence, the corresponding constraint on p under the negative-sense discontinuous transcription model is $\sum_{e \in \delta^-(\mathbf{v})} p(e) = 1$. The probabilities are drawn from a Dirichlet distribution with concentration parameter α set to 10 for edges that are present in the path corresponding to any of the canonical transcripts and 1 otherwise. This is done to ensure that canonical transcripts are generated with high enough abundance, making the simulations similar to real data.

The next step of our simulation pipeline is to generate transcripts $\mathcal{T}$ and their abundances $\mathbf{c}$ for the given segment graph. We simulate the transcription process by generating 100,000 $\mathbf{s} - \mathbf{t}$ paths on the segment graph and report the number of unique paths/transcripts $\mathcal{T}$ and their abundances $\mathbf{c}$. We repeat this process to generate 5 independent sets of transcripts and abundances for the positive and the negative model each. Fig. 3b shows the number of transcripts generated from each simulation using the negative-sense discontinuous transcription model. To contrast, the total number of $\mathbf{s} - \mathbf{t}$ paths in the underlying segment graph is 3440.

Once the transcripts are generated, the next step in our pipeline is to simulate the generation and sequencing of RNA-seq data. We use `polyester` [24] for this step as it allows the user to provide the number of reads generated from each transcript. For a given total number n of reads, the number of reads generated from transcript T_i is given by $nc_iL_i/\sum_{j=1}^{k} c_jL_j$ where L_i is the length of the transcript T_i. We use the default parameters for read length ($\ell = 100$) and fragment length distribution (Gaussian with mean $\mu_r = 250$ and standard deviation $\sigma_r = 25$) to generate 3,000,000 reads. For each set of transcript and abundances

generated in the previous step of the pipeline, we simulate 5 replicates of the sequencing experiment.

The final step of the simulation pipeline is to align the generated reads to the reference genome NC_045512.2 using `STAR` [15]. The resulting BAM file serves as the input for the transcription assembly methods. To summarize, we generated 5 independent pairs $(\mathcal{T}, \mathbf{c})$ of transcripts and abundances under the negative-sense discontinuous transcription model. For each pair $(\mathcal{T}, \mathbf{c})$ we run 5 simulated sequencing experiments using `polyester` [24]. Therefore, we generated a total of $5 \times 5 = 25$ simulated instances.

C.2 SCALLOP arguments

We use the following arguments.

```
scallop -i ${input_bam} -o ${output_assembled}
```

C.3 STRINGTIE arguments

We run STRINGTIE in de novo transcript assembly mode. That is, we do not provide a GFF file to guide assembly. We use the following arguments.

```
stringtie  -o ${output_assembled} -A ${output_abundance} ${input_bam}
```

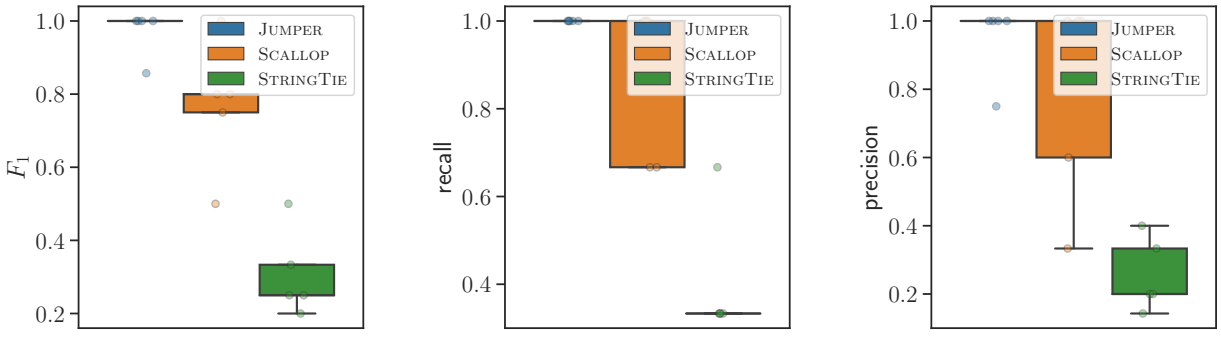
We noted that STRINGTIE produces incomplete transcripts, *i.e.* all the assembled transcripts did not map to the 5' and 3' end of the reference genome. In our simulations, STRINGTIE was not penalized for this as our evaluation metrics considered only discontinuous edges.

C.4 Human gene simulations

We evaluate the performance of JUMPER, SCALLOP and STRINGTIE on simulated samples of a single human gene as well. We use a selected region (from position 5001 to 30255) of the FAS gene as the reference genome[1] with is located on the long arm of chromosome 10 in humans and encodes the Fas cell surface receptor which leads to programmed cell death if it binds its ligand (Fas ligand).

We include the following 3 distinct isoforms of this gene (P25445-1, P25445-6 and P25445-7) with equal proportions in the ground truth.

[1]NCBI reference sequence NG_009089.2: `https://www.ncbi.nlm.nih.gov/nuccore/NG_009089.2?from=5001&to=30255&report=fasta`



(a) **(b)** **(c)**

Figure C1: JUMPER outperforms SCALLOP and STRINGTIE for all simulation instances of the FAS gene (on human chromosome 10) in terms of F_1 score, recall and precision while maintaining a modest running time. (a) F_1 score (b) recall and (c) precision of the three methods for the simulated instances. The ground truth contained three isoforms of the FAS gene with uniform relative abundances.

1. P25445-1 with length of 335aa
   `https://www.ncbi.nlm.nih.gov/CCDS/CcdsBrowse.cgi?REQUEST=CCDS&DATA=CCDS7393.1`

2. P25445-6 with length of 314aa
   `https://www.ncbi.nlm.nih.gov/CCDS/CcdsBrowse.cgi?REQUEST=CCDS&DATA=CCDS7394.1`

3. P25445-7 with length of 220aa
   `https://www.ncbi.nlm.nih.gov/CCDS/CcdsBrowse.cgi?REQUEST=CCDS&DATA=CCDS7395.1`

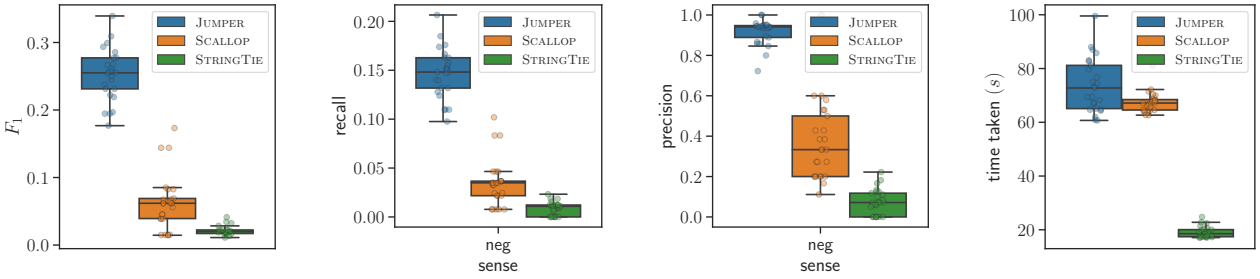
We add a poly-A tail of length 85 at the end of the reference genome as well as each of the isoforms to emulate the transcription process.

We use polyester [24] to simulate the sequencing of 30,000,000 paired-end reads of the sample with a Gaussian fragment length distribution with mean 250 and standard deviation of 25. We simulate 5 replicates of the sequencing experiment. The simulated reads are aligned to the selected region of the FAS gene using `STAR` [15]. The resulting BAM file serves as the input for the transcription assembly methods. Figure C1 shows that JUMPER outperforms both SCALLOP and STRINGTIE in terms of both recall and precision.

C.5 Supplementary results figures

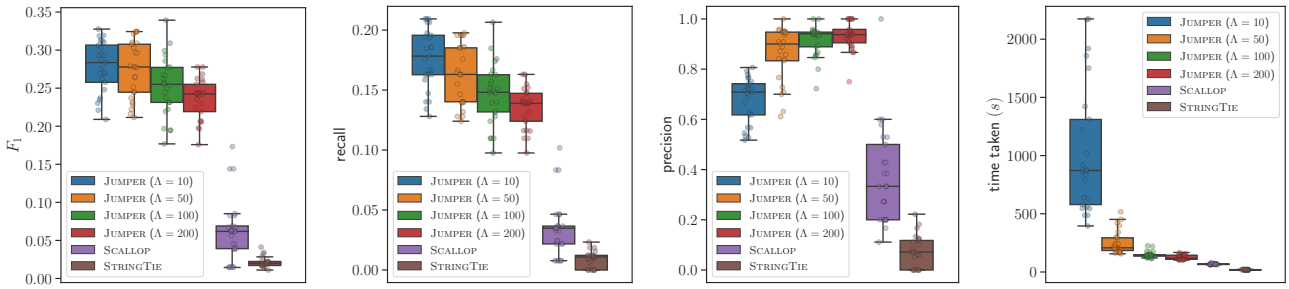
We have the following supplementary figures.

- Fig. C2 shows that JUMPER outperforms SCALLOP and STRINGTIE for all simulation instances in terms of F_1 score, recall and precision while maintaining a modest running time.
- Fig. C3 shows that JUMPER outperforms SCALLOP and STRINGTIE for varying values of thresholding parameter Λ.
- Fig. C4 shows that JUMPER produces better recall and precision when compared to SCALLOP and STRINGTIE for every simulation instance $(\mathcal{T}, \mathbf{c})$.
- Fig. C5 shows that the core sequence observed in the reference genome potentially explaining a non-canonical discontinuous transcription event, and the core sequence corresponding to transcript X is conserved across *Sarbecovirus* species.
- Fig. C6 shows an example of a supporting read for a transcript with two discontinuous edges.
- Fig. C7 shows that transcript X is supported in both long-read and short-read samples deposited in SRA.
- Fig. C8 shows the number of *supporting reads* with the 5' end mapping to the leader sequence in the short and long read sequencing data.
- Fig. C9 shows the abundances of the predicted transcripts by JUMPER in two SARS-CoV-1 infected samples.
- Table C1 shows summary of the results from the simulations.



(a) **(b)** **(c)** **(d)**

Figure C2: JUMPER outperforms SCALLOP and STRINGTIE for all simulation instances in terms of F_1 score, recall and precision while maintaining a modest running time. (a) F_1 score (b) recall, (c) precision and (d) time taken by the three methods for the simulated instances.



(a) **(b)** **(c)** **(d)**

Figure C3: JUMPER outperforms SCALLOP and STRINGTIE for varying values of thresholding parameter Λ. (a) F_1 score (b) recall, (c) precision and (d) time taken by the JUMPER for different values of Λ compared to SCALLOP and STRINGTIE on the simulated instances. As expected, the recall value drops for increasing Λ while the precision increases. We set the default value of Λ to 100 which incurs runtime comparable to SCALLOP while producing higher recall and precision solutions.

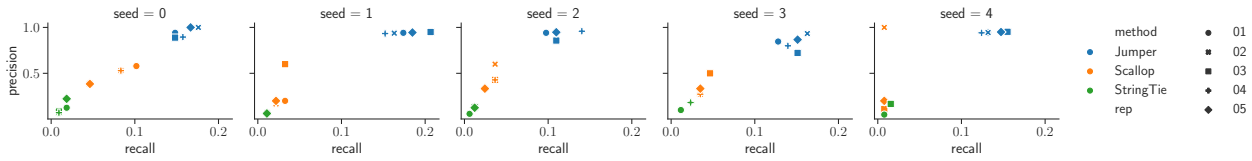
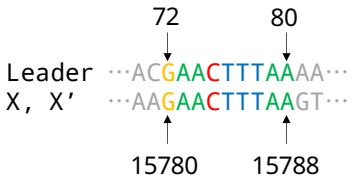
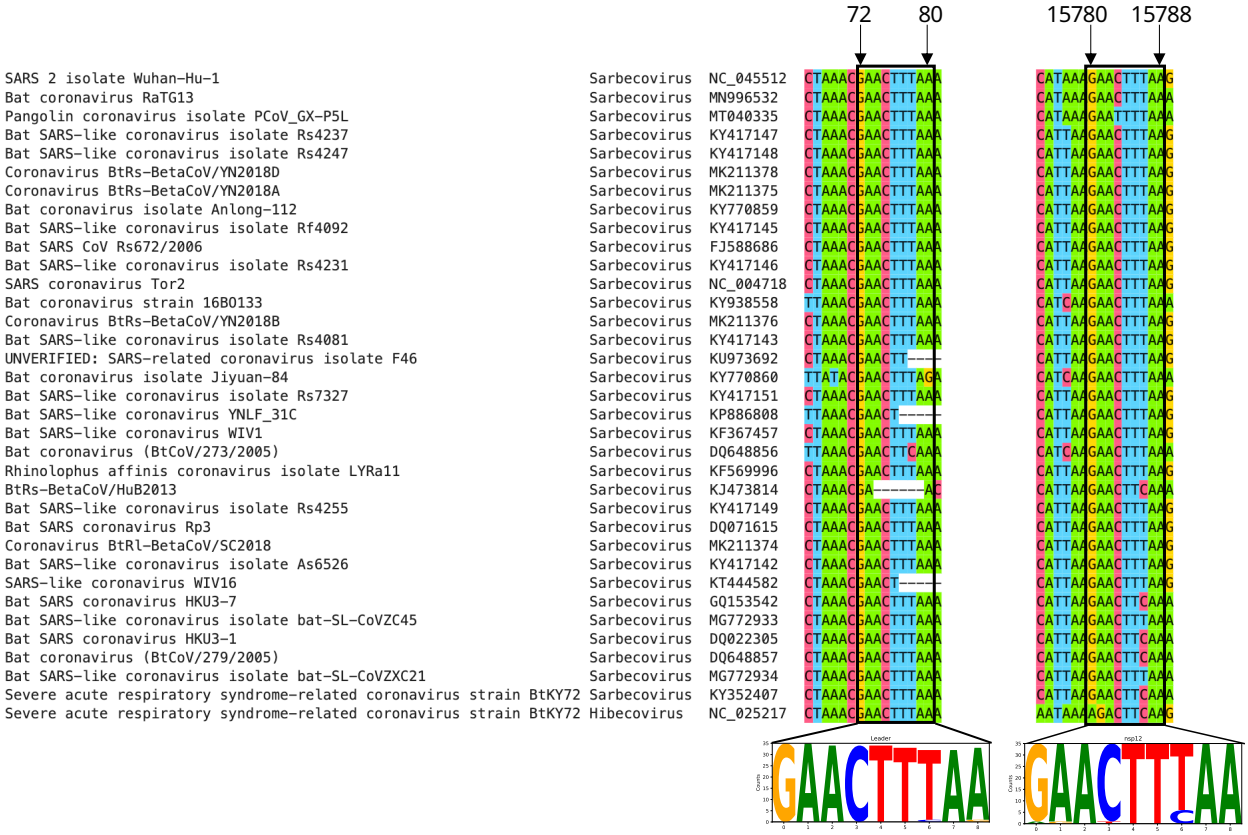


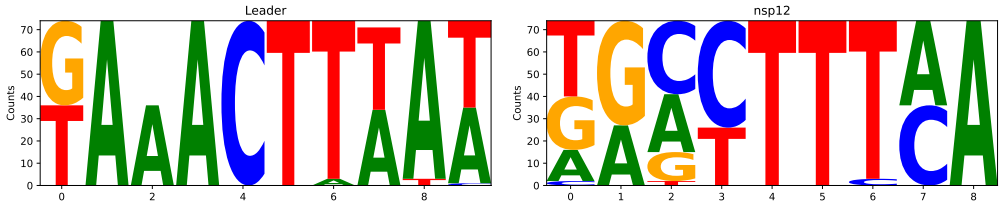
Figure C4: While all three methods return consistent results when generating technical sequencing replicates, JUMPER produces better recall and precision when compared to SCALLOP and STRINGTIE for every simulation instance $(\mathcal{T}, \mathbf{c})$. Varying simulation instances $(\mathcal{T}, \mathbf{c})$ correspond to distinct panels. Each panel shows the recall and precision of the three methods for 5 sequencing experiments of the same simulated instance $(\mathcal{T}, \mathbf{c})$.



(a) Core sequence



(b) MSA and sequence logo with 34 *Sarbecoviruses* and 1 *Hibecovirus*.



(c) Sequence logo of 11 *Nobecoviruses*, 27 *Merbecoviruses* and 36 *Embecoviruses*

Figure C5: The core sequence of transcript X is conserved within the *Sarbecovirus* subgenus but not in other subgenera of the *Betacoronavirus* genus. (a) Core sequence for the transcript X and X'. (b) Sequence logo for the positions 15780 to 15788 in SARS-CoV-2 genome built from the multiple sequence alignment of the leader sequence and ORF1ab of 34 *Sarbecovirus* and a *Hibecovirus*. (c) Sequence logo for positions 15780 to 15788 in SARS-CoV-2 genome built from multiple sequence alignment with the remaining subgenera of *Betacoronaviruses*.

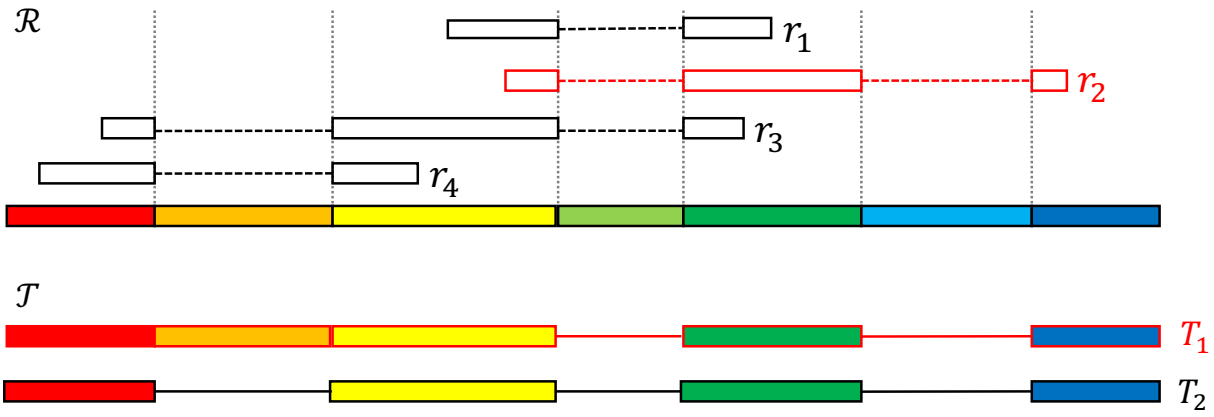
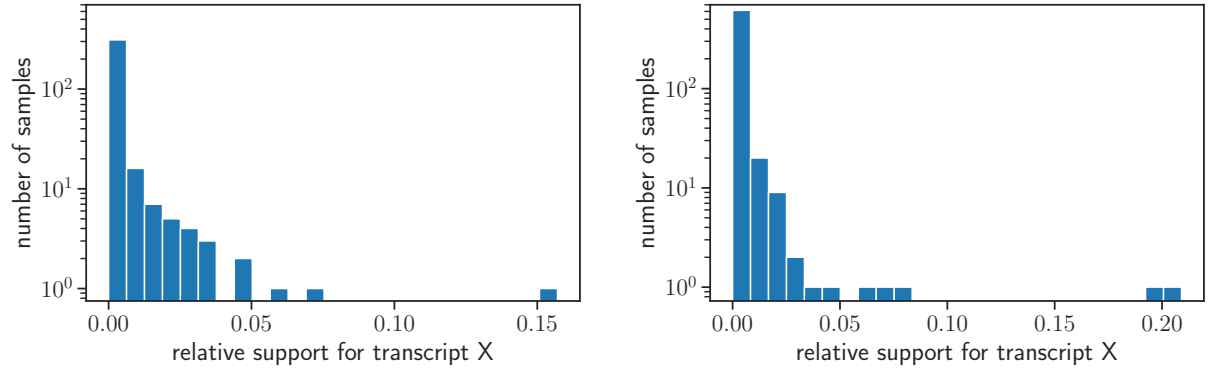


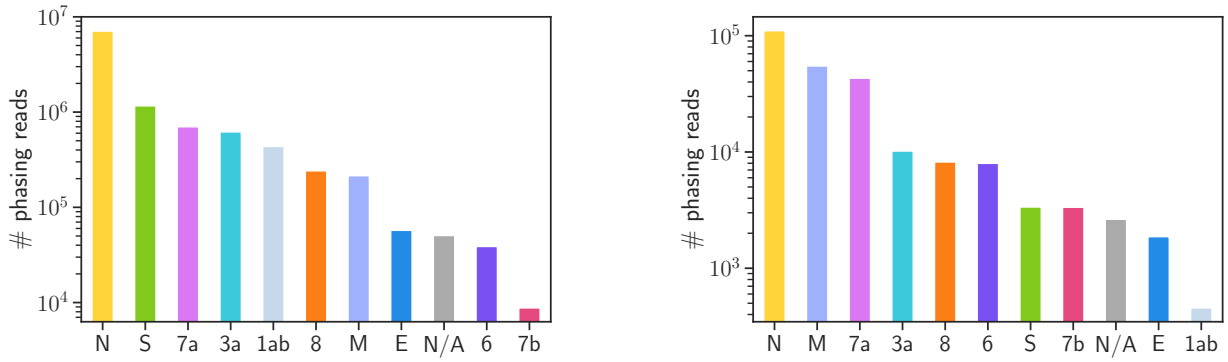
Figure C6: A schematic showing an example of a supporting read for a transcript T_1 with $\sigma^{\oplus}(T_1) = 2$. Transcript T_1 is supported by r_2 because $\pi(r_2) = \pi(T_1)$ and $|\sigma^{\oplus}(r_1)| = |\sigma^{\oplus}(T_1)| = 2$. Reads r_1, r_3 and r_4 do not support T_1 since $|\sigma^{\oplus}(r_1)| < |\sigma^{\oplus}(T_1)|$ and $\pi(r_3), \pi(r_4) \nsubseteq \pi(T_1)$. No reads support T_2 since $|\sigma^{\oplus}(r_j)| < |\sigma^{\oplus}(T_2)|$ for all reads r_j.



(a) short-read samples

(b) long-read samples

Figure C7: Transcript X has supporting reads in multiple independent SRA samples of SARS-CoV-2. Distribution of number of (a) short-read and (b) long-read SRA samples with varying proportion of leader-sequence spanning reads that support transcript X. All the short-read samples were aligned using `STAR` [15] while the long-read samples were aligned using `minimap2` [27]. In this plot we only consider samples with more than 100 reads that map to the leader-sequence (position 55 to 85 in the SARS-CoV-2 reference genome).



(a) **(b)**

Figure C8: Supporting phasing reads with 5' end mapping to the leader sequence in short and long-read sequencing samples of SARS-CoV-2 infected Vero cells [3]. Supporting phasing reads have at most one discontinuous edge with the 5' end occurring in the leader sequence (*i.e.* between positions 55 and 85) and the first occurrence of 'AUG' downstream of the 3' end position coinciding with the start codon of a known ORF. Supporting phasing reads corresponding to '1ab' start in the leader sequence but do not contain a discontinuous edge. Supporting phasing reads corresponding to 'N/A' start in the leader sequence but have a 3' end such that the first occurrence of 'AUG' downstream of the 3' end position does *not* coincide with the start codon of any known ORFs. (a) Supporting phasing reads in the short-read sequencing sample. (b) Supporting phasing reads in the long-read sequencing sample.

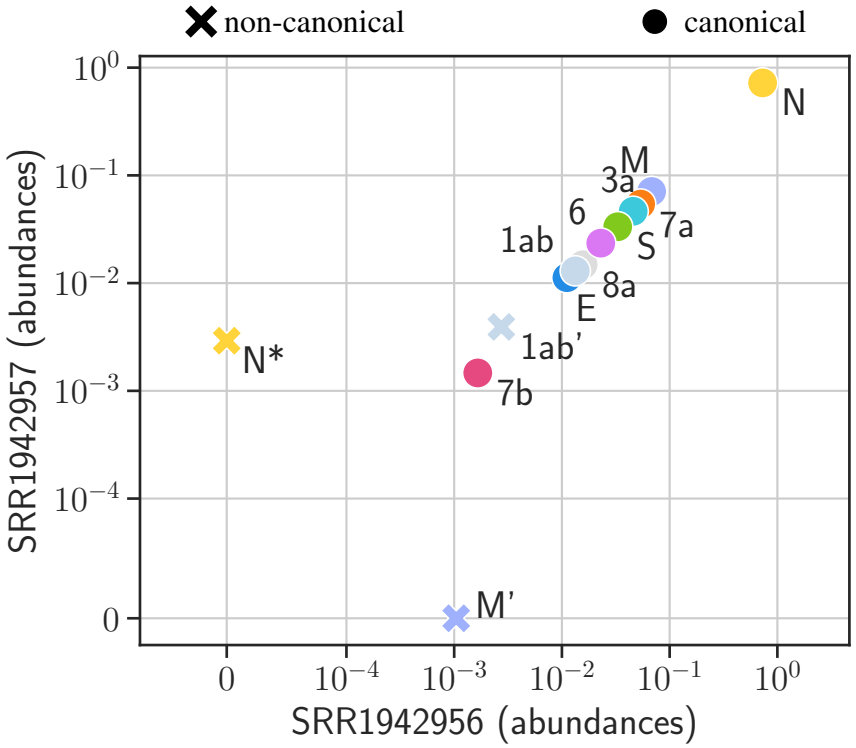


Figure C9: Abundances of the canonical and non-canonical transcripts predicted by JUMPER are consistent in the two SARS-CoV-1 infected samples (SRR194256 and SRR194257). JUMPER predicts 10 canonical and 3 non-canonical transcripts across the two samples.

Simulation				JUMPER			SCALLOP			STRINGTIE		
seed	rep	can	non-can	TP		FP	TP		FP	TP		FP
				can	non-can		can	non-can		can	non-can	
0	1	14	94	7	9	1	7	4	8	2	0	14
0	2	14	94	8	11	0	7	2	8	1	0	13
0	3	14	94	7	11	2	4	1	8	1	0	11
0	4	14	94	6	9	2	7	2	8	1	0	13
0	5	14	94	7	11	0	4	1	8	1	0	7
1	1	14	78	3	13	1	3	0	12	2	0	13
1	2	14	78	4	11	1	2	0	10	1	0	13
1	3	14	78	3	16	1	3	0	2	1	0	12
1	4	14	78	3	11	1	2	0	8	0	0	16
1	5	14	78	4	13	1	2	0	8	1	0	15
2	1	14	150	5	11	1	3	1	8	1	0	16
2	2	14	150	4	14	3	5	1	4	2	0	15
2	3	14	150	5	13	3	5	1	8	2	0	13
2	4	14	150	7	16	1	5	1	8	2	0	16
2	5	14	150	4	14	1	3	1	8	2	0	14
3	1	14	72	4	7	2	3	0	8	1	0	9
3	2	14	72	6	8	2	3	0	4	0	0	8
3	3	14	72	7	6	4	4	0	8	0	0	20
3	4	14	72	4	8	3	3	0	8	2	0	9
3	5	14	72	4	9	2	3	0	6	0	0	4
4	1	14	115	4	13	1	1	0	4	1	0	19
4	2	14	115	5	12	1	1	0	0	0	0	12
4	3	14	115	6	14	1	1	0	8	2	0	10
4	4	14	115	6	10	1	1	0	4	0	0	16
4	5	14	115	6	13	1	1	0	4	0	0	12

Table C1: Simulation results for the three methods JUMPER, SCALLOP and STRINGTIE. Each distinct value in the column 'seed' is a unique instance of $(\mathcal{T}, \mathbf{c})$ and each distinct value in the column 'rep' is a unique sequencing experiment for the given $(\mathcal{T}, \mathbf{c})$. (rep: replicate, can: canonical, non-can: non-canonical, TP: true positives, FP: false positives)