MultiTrans: a novel algorithm for transcriptome assembly through mixed integer linear programming

Jin Zhao[1,2], Haodi Feng[1,*], Daming Zhu[1], and Yu Lin[2,*]

[1] *School of Computer Science and Technology, Shandong University, Qingdao, 266237, China*
[2] *Research School of Computer Science, Australian National University, Canberra, 2601, Australia*
[*] *Correspondence should be addressed to Haodi Feng (fenghaodi@sdu.edu.cn) or Yu Lin (yu.lin@anu.edu.au)*

Recent advances in RNA-seq technology have made identification of expressed genes affordable, and thus boosting repaid development of transcriptomic studies. Transcriptome assembly, reconstructing all expressed transcripts from RNA-seq reads, is an essential step to understand gene, protein, and cell functions. Transcriptome assembly remains a challenging problem due to complications in splicing variants, expression level, uneven coverage and sequencing errors. Here, we formulate the transcriptome assembly problem as path extraction on splicing graph (or assembly graph), and propose a novel algorithm MultiTrans for path extraction using mixed integer linear programming. MultiTrans is able to take into consideration coverage constraints on vertices and edges, the number of paths and the paired-end information simultaneously. We benchmarked MultiTrans against two state-of-the-art transcriptome assemblers, TransLiG and rnaSPAdes. Experimental results show that MultiTrans generates more accurate transcripts compared to TransLiG (using the same splicing graph) and rnaSPAdes (using the same assembly graph). MultiTrans is freely available at https://github.com/jzbio/MultiTrans.

1 Introduction

Alternative splicing allows a gene to express multiple transcripts, which plays an important role in regulating gene expression and producing diversity of proteins (Baralle and Giudice, 2017; Kelemen *et al.*, 2013). Studies show that transcripts from more than 95% of multiexon genes in human undergo alternative splicing (Pan *et al.*, 2008). Besides, some diseases such as cancer are related to abnormal splicing events (Biamonti *et al.*, 2019; Climente-González *et al.*, 2017; Paronetto *et al.*, 2016). Therefore, the identification of all expressed transcripts plays an important role in diseases researches and transcriptomic studies.

High-throughput RNA sequencing (RNA-seq) technology has provided an unprecedented opportunity to transcriptomic studies (Ozsolak and Milos, 2011; Liu *et al.*, 2016c; Safikhani *et al.*, 2017). The RNA-seq protocol takes the expressed transcripts as input and outputs millions of short reads. In principle, such short reads can allow us to recover all expressed transcripts. However, reconstructing all expressed transcripts from RNA-seq reads remains a substantial computational challenge. This task is complicated by highly similar paralogs, various alternative splicing variants, different expression levels of isoforms of the same gene, and sequencing errors and bias.

Existing work on transcriptome assembly problem can be mainly divided into two categories: reference-based approaches and de novo assembly approaches. The reference-based assemblers such as StringTie2 (Kovaka *et al.*, 2019), Ryūtō (Gatter and Stadler, 2019), Scallop (Shao and Kingsford, 2017a), TransComb (Liu *et al.*, 2016b), Cufflinks (Trapnell *et al.*, 2012), Bayesember (Optimization, 2014), CIDANE (Canzar *et al.*, 2016), iReckon (Mezlini *et al.*, 2013), Traph (Tomescu *et al.*, 2013), and Scripture (Guttman *et al.*, 2010) usually achieve higher accuracy than de novo assemblers when a high-quality reference genome is available. However, the reference-based strategies depend heavily on the quality of alignment tools, and they are seriously limited in practice for the absence of high-quality reference genomes. Although de novo assemblers may generate more artificial transcripts than reference-based assemblers, they are desired when reference genome is unknown, incomplete, or substantially altered as in cancer tissues (Liu *et al.*, 2019).

De novo approaches for transcriptome assembly such as rnaSPAdes (Bushmanova *et al.*, 2019), TransLiG (Liu *et al.*, 2019), IsoTree (Zhao *et al.*, 2018), BinPacker (Liu *et al.*, 2016a), Bridger (Chang *et al.*, 2015), SOAPdenovo-trans (Xie *et al.*, 2014), IDBA-Tran (Peng *et al.*, 2013), Oases (Schulz *et al.*, 2012), ABySS (Birol *et al.*, 2009), and Trinity (Grabherr *et al.*, 2011) first construct graphs to represent splicing variants and then extract paths from graphs as recovered transcripts. TransLiG and rnaSPAdes are two state-of-the-art de novo transcriptome assemblers. TransLiG constructs splicing graphs (Heber *et al.*, 2002) while rnaSPAdes builds assembly graphs (simplified version of de Bruijn graphs (Pevzner *et al.*, 2001)) to model splicing variants for future path extraction.

Various path extraction algorithms have been proposed to identify paths as transcripts. For example, Trinity (Grabherr *et al.*, 2011) enumerates all paths in a brute-force manner to recover as more transcript candidates as possible. The enumeration strategy ensures as many correct transcripts as possible to be obtained but may introduce a large number of artificial transcripts. Cufflinks (Trapnell *et al.*, 2012), TRIP (Mangul *et al.*, 2012), MLIP (Mangul *et al.*, 2013), and Bridger (Chang *et al.*, 2015) aim to find the minimum number of transcript candidates, which achieves a high accuracy but may miss some true transcripts. Many other algorithms, such as SSP (Safikhani *et al.*, 2013), CLASS (Song and Florea, 2013), and CLIIQ (Lin *et al.*, 2012), adopt a linear programming model with different objectives and constrains to extract paths. However, the paired-end information has not been explicitly included in these linear programming models. More recently, BinPacker (Liu *et al.*, 2016a) extracts the paths by solving a series of bin-packing problem, which makes a full use of sequencing depth but still ignores the paired-end information. rnaSPAdes (Bushmanova *et al.*, 2019) and TransLiG (Liu *et al.*, 2019) are two state-of-the-art de novo transcriptome assemblers, which take into consideration both the sequencing depth and paired-end information. rnaSPAdes adopts a path extension framework to construct paths by selecting the edge that is best supported by paired-end reads (Prjibelski *et al.*, 2014). TransLiG looks for paths that best balance the weights between all the incoming and out-going edges. However, both rnaSPAdes and TransLiG recover paths tend to make locally optimal choices, which cannot guarantee to find a global optimal solution.

In this paper, we introduce MultiTrans, a linear programming model with more comprehensive consideration (such as the sequencing depth, the paired-end information, the assembly graph, etc) for the transcriptome assembly problem. The main contributions of this work include:

(i) Propose a unified path-extraction model for transcriptome assembly problem that considers coverage constraints on vertices and edges, the number of paths and the paired-end information simultaneously.

(ii) Find a global optimal solution for the path-extraction model through a mixed integer linear programming (MILP). MultiTrans also provides efficient preprocessing to reduce the size of MILP instance.

(iii) Apply the MultiTrans algorithm to the graphs constructed by TransLiG and rnaSPAdes, respectively. Experimental results show that MultiTrans outperforms rnaSPAdes and TransLiG on both simulated and real datasets.

2 Methods

In this section, we will show how MultiTrans takes splicing graphs (or assembly graphs) built from reads as input and extracts paths from these graphs to output transcripts. In the following, the term "splicing graph" (Liu *et al.*, 2019) also refers to a connected component in the assembly graph (Bushmanova *et al.*, 2019).

2.1 Preliminaries

A transcript corresponds to a path in a splicing graph $G(V, E)$. The coverage of a transcript is represented by the flow of the corresponding path. A vertex $v \in V$ usually represents continuous transcript sequence without any alternative splicing events. Two vertices are connected by an edge if they are consecutive in one of transcripts. The coverage of a vertex (or an edge) denotes the total coverage of the corresponding sequence.

The transcriptome assembly problem can thus be modeled as extracting a set of paths with respect to the following objectives:

Objective (1): all the vertices and edges are covered in at least one path.

Objective (2): all the paired-end reads are covered in at least one path.

Objective (3): the coverage of each vertex is as close as possible to the sum of flows of paths that pass through this vertex.

Objective (4): the number of paths is as small as possible.

2.2 Mixed integer linear programming

MultiTrans formulates the above path extraction problem as a mixed integer linear programming which aims to find an optimal path set to meet the four objectives in the above section.

Similar to Shao and Kingsford, 2017b; Zhao *et al.*, 2019, MultiTrans first adds a source vertex and a sink vertex to the graph. MultiTrans then connects the source vertex to vertices without incoming edges and connects vertices without outgoing edges to the sink vertex. Assume P is the set of all paths from the source vertex to the sink vertex in the above graph[1]. Note that not all the paths in P correspond to a real transcript.

MultiTrans introduces a binary variable s_i to indicate whether the i^{th} path is selected ($s_i = 1$) or not ($s_i = 0$). The vertices and edges in the path p_i is denoted by $V(p_i)$ and $E(p_i)$, respectively. The integer variables $\alpha(v, p_i)$ and $\beta(e, p_i)$ indicate the number of times that v appears in $V(p_i)$ and the number of times that e appears in $E(p_i)$, respectively. The flow on the i^{th} path is denoted by f_i. Note the following constraints guarantee that a path is selected ($s_i = 1$) if and only if its flow is larger than 0 ($f_i > 0$):

$$\lambda s_i \geq f_i \quad 1 \leq i \leq |P|, \tag{1}$$

$$\lambda f_i \geq s_i \quad 1 \leq i \leq |P|, \tag{2}$$

where λ is a large positive number.

MultiTrans employs $c(v)$ and $c(e)$ to represent the observed (input) coverage of vertex v and edge e, respectively. MultiTrans adds the following constraints to meet objective (1), *i.e.,* all the vertices and edges are covered by at least one path:

$$\sum_{i=1}^{|P|} \alpha(v, p_i) s_i \geq 1 \quad v \in V, \tag{3}$$

$$\sum_{i=1}^{|P|} \beta(e, p_i) s_i \geq 1 \quad e \in E. \tag{4}$$

Note that each paired-end read may span multiple vertices and these vertices are expected to appear in a same transcript. Let PE denote the set of all paired-end constraints, where each constraint pe ($pe \in PE$) consists of a set of vertices $V(pe)$ that must occur in at least one transcript simultaneously. Therefore, each paired-end constraint in objective (2) can be modelled through the following constraints:

$$\lambda(|V(pe)| - \sum_{j \in V(pe) \cap V(p_i)} s_i) \geq \mu_{pe,i} \quad pe \in PE, \tag{5}$$

$$|V(pe)| - \sum_{j \in V(pe) \cap V(p_i)} s_i \leq \lambda \mu_{pe,i} \quad pe \in PE, \tag{6}$$

$$\sum_{i=1}^{|P|} \mu_{pe,i} < |P|, \quad pe \in PE. \tag{7}$$

Note that the binary variable $\mu_{pe,i} = 0$ if and only if the i^{th} path is selected and covers all the vertices in this paired-end constraint pe. Thus

[1] Although there are no cycles in the splicing graph (Liu *et al.*, 2019), it is possible that a connected component in the assembly graph (Bushmanova *et al.*, 2019) contains cycles. If there are cycles in the graph, MultiTrans considers each path that contains at most one cycle.

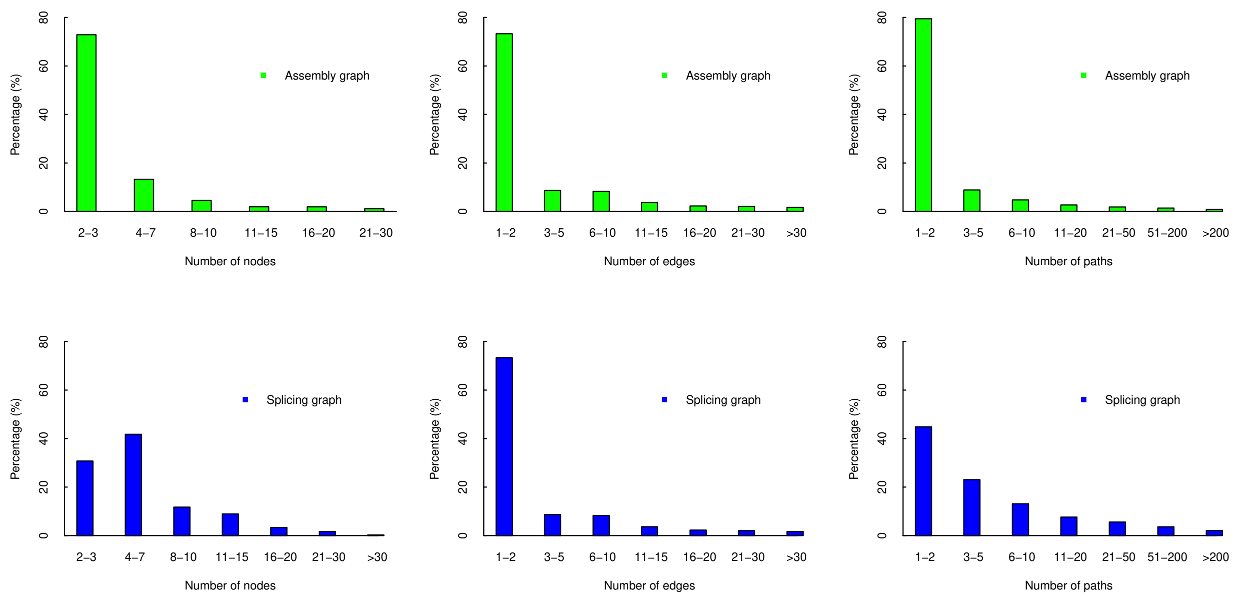

Figure 1: The percentages of graphs (splicing graphs or connected components in assembly graphs) with different number of nodes, edges, and paths out of all the graphs with at least two vertices on real human dataset (SRR5133163).

Equation (7) makes sure that the paired-end constraint is satisfied by at least one path.

In an ideal case, the observed coverage of each vertex should be equal to the predicted coverage, *i.e.,* the sum of flows of selected paths that pass through it. However, due to sequencing bias and other factors, it is unrealistic to match exactly the observed coverage and the predicted coverage for all vertices and edges. MultiTrans introduces a binary variable $l(v)$ to denote the discordance of a vertex v, *i.e.,* the minimum value of $l(v)$ is 1 if the ratio between the predicted coverage and the observed coverage falls outside the interval $[(1-\epsilon),(1+\epsilon)]$ (*i.e.,* v is a discordant vertex with respect to its coverage) otherwise is 0.

$$\lambda l(v) \geq \sum_{i=1}^{|P|} \alpha(v,p_i) f_i - (1+\epsilon)c(v) \quad v \in V, \tag{8}$$

$$\lambda l(v) \geq (1-\epsilon)c(v) - \sum_{i=1}^{|P|} \alpha(v,p_i) f_i \quad v \in V. \tag{9}$$

MultiTrans aims to meet four objectives simultaneously. The first two objectives are satisfied through formulae (3)-(9), and the last two objectives are optimized through the following objective function:

$$min \quad \sum_{v \in V} l(v) + \frac{1}{|P|}\sum_{i=1}^{|P|} s_i. \tag{10}$$

In the above objective function, $min \sum_{v \in V} l(v)$ is to minimize the number of discordant vertices with significant deviation between observed and predicted coverage while $min \sum_{i=1}^{|P|} s_i$ is to minimize the number of selected paths, where $|P|$ is the number of possible paths (refer to Figure 3 for a distribution of $|P|$). Note that $\frac{1}{|P|}$ is introduced to make sure that the mixed linear programming prefers solutions with the minimum number of discordant vertices or solutions with the minimum number of paths if there are multiple solutions with a same minimum number of discordant vertices.

2.3 Preprocessing

As shown in Figure 3, the size of input splicing graphs (or connected components in the assembly

Table 1: Real RNA-Seq datasets selected for comparison of different transcriptome assembly tools.

Dataset name	Organism	No. of reads (millions)	Read length (bp)	Accession No.
Human	Homo sapiens	60	150	SRR5133163
Mouse	Mus musculus	100	76	SRX062280
Rice	Oryza Sativa	73	150	SRR12147608

graph) is usually very small, and thus an optimal solution for the mixed integer linear programming can be derived efficiently by software Gurobi (Optimization, 2014). However, there may still exist large splicing graphs which could lead to high computational cost.

To resolve this issue, MultiTrans introduces a restriction on the number of paths in each splicing graph (similar to Maretty *et al.*, 2014). If the number of paths in a splicing graph is larger than 500, MultiTrans iteratively removes an edge supported by the least number of paired-end reads (by checking the upstream and downstream 500bp from this edge). If multiple such edges exist, MultiTrans removes the edge with the smallest coverage. The iterative removal of edges terminates when the number of paths in the splicing graph becomes less than 500. According to objective (1), each edge (including the deleted one) should be covered by at least one path. Therefore, for each deleted edge (v, v'), MultiTrans retrieves a path to cover (v, v') by concatenating a sub-path from the source vertex to v, the edge (v, v'), and a sub-path from v' to the sink vertex. Note that the above two sub-paths are constructed in a greedy way in which MultiTrans iteratively selects a vertex with the maximum coverage from v (v') to the source (sink) vertex, respectively.

3 Experimental Setting

3.1 Dataset Information

We benchmarked MultiTrans against other assemblers on the following datasets:

- Simulated datasets. There are eight simulated samples $\{SIM_2, SIM_3, \ldots, SIM_9\}$ produced by Flux Simulator (Griebel *et al.*, 2012). Each sample SIM_i contains 1 million 150bp paired-end reads drawn from 5,040 transcripts, and i is the ratio of the number of expressed transcripts to genes (*i.e.*,, each gene expresses i transcprits).
- Real datasets. There are four datasets retrieved from the NCBI Sequence Read Archive (SRA) database with accession code SRR5133163, SRX110318, SRR12147608, and SRX062280, respectively. A detailed description of these datasets can be found in Table 1.

3.2 Assessment metrics

For transcriptome assembly, one of the most important indicator of assembly quality is the number of full-length reconstructed transcripts. Here, we first aligned assembled transcript candidates to the annotated transcripts using BLAST+ (Camacho *et al.*, 2009). We then used the analysis script provided by Trinity (Grabherr *et al.*, 2011) to examine the percentage of the target being aligned to by the best matching assembled transcript candidates. Note that if an annotated transcript matches multiple transcript candidates as their best hits, this annotated transcript is counted only once along with the transcript candidate that provides the highest BLAST bit score and longest match length (Grabherr *et al.*, 2011). An annotated transcript is called as a $X\%$*-reconstructed transcript* if at least $X\%$ of its length is covered by the above best matching. Specially, 95%-reconstructed transcripts are referred to as *full-length reconstructed transcripts.*

When the ground-truth transcripts are known in simulated datasets, we employ the $F1\text{-}sore = \frac{2*recall*precision}{recall+precision}$ to measure the accuracy of reconstructed transcripts, where *recall* is defined as the fraction of full-length reconstructed transcripts out of all ground-truth transcripts, and *precision* is defined as the fraction of full-length reconstructed transcripts out of all reconstructed transcript candidates.

When the ground-truth transcripts are not known in real datasets, all the annotated transcripts from NCBI databases are considered as the ground truth transcripts. The annotated transcriptome of human and mouse consists of 165,009 and 120,304 transcripts with length larger than 200*bp*, respectively. However, the number of real expressed transcripts is usually far less than the number of annotated transcripts, and some novel transcripts may not be included in the annotation. Therefore, we employ the number of different percentage reconstructed transcripts when evaluating results of real datasets.

3.3 Implementation and running environment

We applied the MultiTrans algorithm on the assembly graphs (simplified de Bruijn graphs) constructed by rnaSPAdes (Bushmanova *et al.*, 2019) as well as the splicing graphs constructed by TransLiG (Liu *et al.*, 2019). Note that rnaSPAdes (version 3.13.1) and TransLiG (version 1.3) are applied with their default parameters. All experiments were run on a server with 256GB of RAM and E5-2620V3*2 CPU processor.

Regarding the paired-end information, we used the BWA aligner Li, 2013 to align paired-end reads to the vertices of the assembly graphs constructed by rnaSPAdes (Bushmanova *et al.*, 2019), and we inherited the default paired-end information provided within the splicing graphs constructed by TransLiG (Liu *et al.*, 2019).

When running Gurobi (Optimization, 2014) for the mixed integer linear programming, we set a time limit of 5 minutes of each instance, *i.e.*, the current best solution from Gurobi will be returned if Gurobi does not terminate in 5 minutes. If the best solution derived from Gurobi still contains more than 10% of discordant vertices of a splicing graph, MultiTrans ignores the Gurobi solution and outputs all paths in this splicing graph when the total number of such paths is less than 10. For each vertex in the splicing graph, its *embedded read number* is defined as the multiplication of its length and coverage divided by the average read length. MultiTrans filters isolated vertices in the splicing graph if its length is less than 200 or its embedded read number is less than 10% of the average value across all splicing graphs.

4 Results

4.1 Evaluation on simulated datasets of different complexities

We tested MultiTrans, rnaSPAdes and TransLiG on eight simulated datasets with different complexities. The number of transcripts expressed by each gene in these eight datasets is from 2 to 9, respectively. Table 2 summarizes F1-score, precision and recall of assemblers on these simulated datasets.

As shown in Table 2, with the same input graphs, MultiTrans obtained higher F1-score and precision than rnaSPAdes and TransLiG on all simulated datasets. For example, the average precision achieved by MultiTrans based on assembly graphs was 0.434, which increased the average precision of rnaSPAdes (0.283) with 53.4%. Although TransLiG achieved higher recall than MultiTrans on datasets SIM_2 and SIM_3, MultiTrans performed better recall than TransLiG on other six simulated datasets. Besides, MultiTrans produced a higher average recall than TransLiG. The improvement made by MultiTrans over rnaSPAdes or TransLiG became more significant when the number of expressed transcripts by each gene got larger. For example, MultiTrans improved the F1-socres of rnaSPAdes and TransLiG on dataset SIM_2 with 7.0% and 3.0%, respectively, while it increased the F1-scores of rnaSPAdes and TransLiG on dataset SIM_9 with 50.3% and 11.8%, respectively. When more transcripts by each gene are expressed, the graphs are expected to get more complicated, and thus MultiTrans benefits from its application of the mixed integer linear programming to derive more accurate paths on non-trivial graphs (splicing graphs with at least two vertices or connected components with at least two vertices in assembly graphs).

Figure 2 indicates that the performance of MultiTrans is related to the average numbers of vertices, edges and paths in non-trivial graphs. In most cases, MultiTrans based on assembly graphs (or splicing graphs) achieved a higher improvement over rnaSPAdes (or TansLiG) on the dataset with complicated non-trivial graphs.

Table 2: Benchmarking of rnaSPAdes, TransLiG, and MultiTrans on small simulated datasets

Dataset	Parameter	rnaSPAdes (assembly graph)	MultiTrans (assembly graph)	TransLiG (splicing graph)	MultiTrans (splicing graph)
SIM_2	F1-score	0.214	**0.229**	0.233	**0.240**
	precision	0.190	**0.215**	0.202	**0.220**
	recall	0.246	0.246	**0.275**	0.264
SIM_3	F1-score	0.222	**0.252**	0.243	**0.258**
	precision	0.231	**0.298**	0.240	**0.273**
	recall	0.213	**0.2175**	**0.247**	0.244
SIM_4	F1-score	0.274	**0.320**	0.300	**0.329**
	precision	0.346	**0.492**	0.331	**0.384**
	recall	0.227	**0.238**	0.274	**0.288**
SIM_5	F1-socre	0.211	**0.248**	0.245	**0.272**
	precision	0.282	**0.412**	0.286	**0.338**
	recall	0.168	**0.177**	0.214	**0.228**
SIM_6	F1-score	0.205	**0.248**	0.249	**0.284**
	precision	0.287	**0.450**	0.301	**0.358**
	recall	0.159	**0.171**	0.212	**0.236**
SIM_7	F1-score	0.221	**0.285**	0.268	**0.311**
	precision	0.344	**0.566**	0.341	**0.404**
	recall	0.163	**0.190**	0.220	**0.253**
SIM_8	F1-score	0.169	**0.213**	0.217	**0.248**
	precision	0.264	**0.442**	0.288	**0.344**
	recall	0.124	**0.140**	0.174	**0.194**
SIM_9	F1-score	0.199	**0.299**	0.246	**0.275**
	precision	0.322	**0.600**	0.327	**0.389**
	recall	0.143	**0.199**	0.197	**0.213**

4.2 Evaluation on real datasets

We benchmarked rnaSPAdes, TransLiG, and MultiTrans on three real publicly available RNA-seq datasets: Human dataset (SRR5133163), Mouse dataset (SRX0662280), and Rice dataset (SRR12147608). The numbers of reconstructed transcripts and assembled transcript candidates are presented in Table 3-5.

Table 3 shows the reconstructed transcripts reported by different assemblers on Human dataset (SRR5133163). Taking the same input graphs, MultiTrans recovered more reconstructed transcripts than rnaSPAdes and TransLiG. For example, MultiTrans based on assembly graphs produced 1,446 more full-length (95%) reconstructed transcripts than rnaSPAdes, and MultiTrans based on splicing graphs recovered 88 more full-length reconstructed transcripts than TransLiG. Note that MultiTrans made a more significant improvement over rnaSPAdes than TranLiG. This is consistent with the observation that rnaSPAdes outputs more non-trivial graphs than TransLiG (refer to Table 6), which offers more chances for MultiTrans to use the mixed integer linear programming to recover transcripts. While TransLiG already used both sequencing depth and paired-end information in its path extraction, MultiTrans still obtained better reconstructed transcripts. The number of transcript candidates was largely reduced, as MultiTrans took into consideration of the number of paths in the mixed integer linear programming in addition to sequencing depth and paired-end information.

Table 4 presents the resulting reconstructed transcripts on the Mouse dataset (SRX0662280).

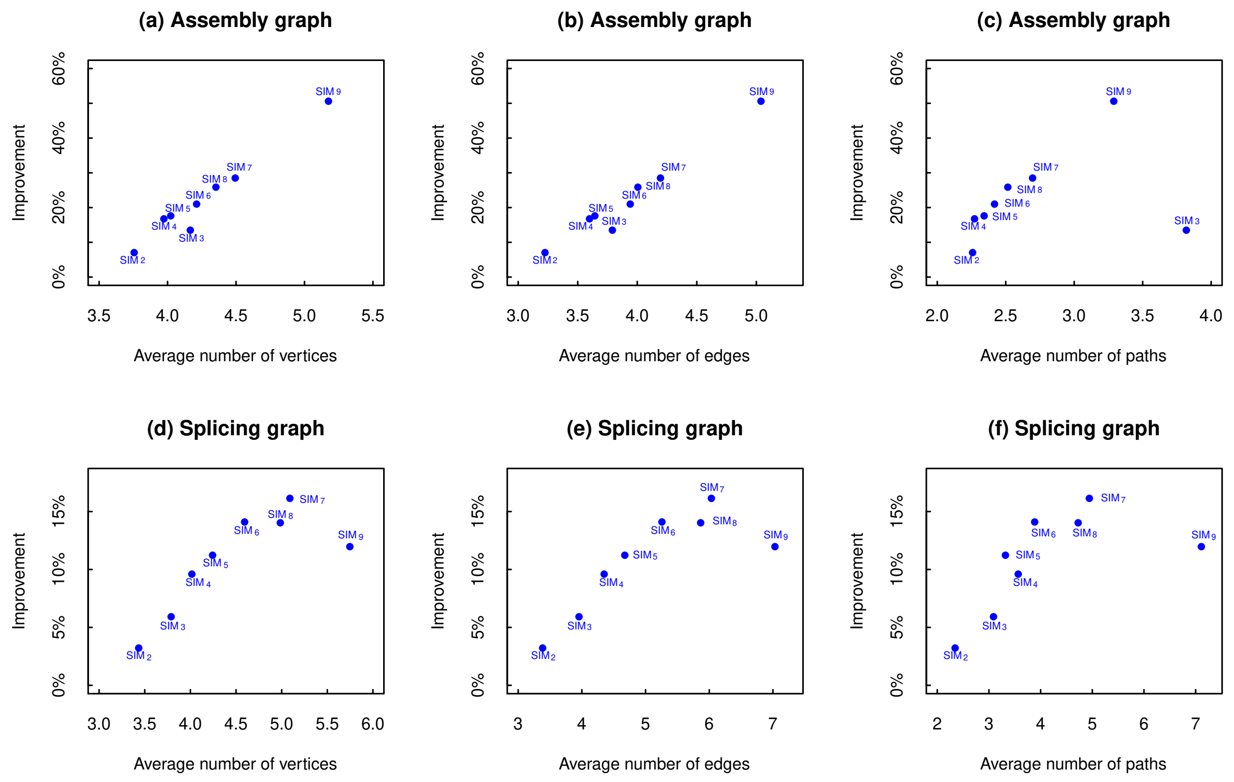

Figure 2: Correlation between the complexities of graphs and the relative improvements of F1-scores of MultiTrans over rnaSPAdes and TransLiG on simulated datasets. The complexities of graphs (assembly graphs and splcing graphs) are measured by the average numbers of vertices, edge, and paths in non-trivial graphs, where non-trivial graphs are splicing graphs (or connected components in assembly graphs) with at least two vertices.

Similar to the previous results, MultiTrans achieved better performance compared to rnaSPAdes and TransLiG. For example, MultiTrans based on assembly graphs obtained 334 more full-length reconstructed transcripts than rnaSPAdes, and MultiTrans based on splicing graphs reconstructed 202 more full-length reconstructed transcripts than TransLiG. Interestingly, the improvement by MultiTrans over rnaSPAdes on real mouse dataset was lower than it on the real human dataset. This is consistent with the observation that the average numbers of vertices, edges and paths of non-trivial graphs constructed by rnaSPAdes on the real mouse dataset are smaller than that of on real human dataset, respectively (Table 6). Although MultiTrans based on splicing graphs achieved a higher improvement in recovering full-length reconstructed transcripts on the real mouse dataset than on the real human dataset, its performance on reducing transcript candidates on real mouse dataset was not as good as on the real human dataset. MutiTrans obtained 2.8% more full-length reconstructed transcripts and 30.0% fewer candidates than TransLiG on the real mouse dataset while it produced 1% more full-length reconstructed transcripts and 39.0% fewer candidates than TransLiG on the real human dataset. Therefore, we think MultiTrans achieves a higher improvement over TransLiG on the real human dataset than on the real mouse dataset. Note that this conclusion is also consistent with the observation that the non-trivial graph on the real human dataset is more complicated than it on the real mouse dataset.

Table 5 demonstrates the assembling results on the Rice dataset (SRR12147608). When taking assembly graphs from rnaSPAdes as input,

Table 3: Benchmarking of rnaSPAdes, TransLiG, and MultiTrans on Human dataset (SRR5133163)

Parameter	rnaSPAdes (Assembly graph)	MultiTrans (Assembly graph)	TransLiG (Splicing graph)	MultiTrans (Splicing graph)
100%-reconstructed	7,381	**8,616**	7,457	**7,661**
95%-reconstructed	8,886	**10,332**	9,030	**9,118**
90%-reconstructed	9,840	**11,499**	10,157	**10,215**
85%-reconstructed	10,742	**12,556**	11,145	**11,196**
80%-reconstructed	11,555	**13,470**	12,119	**12,214**
candidates	182,400	**92,916**	101,478	**61,875**

Table 4: Benchmarking of rnaSPAdes, TransLiG, and MultiTrans on Mouse dataset (SRX0662280)

Parameter	rnaSPAdes (Assembly graph)	MultiTrans (Assembly graph)	TransLiG (Splicing graph)	MultiTrans (Splicing graph)
100%-reconstructed	6,627	**6,930**	6,230	**6,442**
95%-reconstructed	7,609	**7,943**	7,266	**7,468**
90%-reconstructed	8,263	**8,633**	7,995	**8,177**
85%-reconstructed	8,807	**9,251**	8,555	**8,773**
80%-reconstructed	9,324	**9,883**	9,110	**9,311**
candidates	81,241	**61,235**	30,941	**21,648**

MultiTrans reported more reconstructed transcripts and fewer candidates, and thus outperformed rnaSPAdes. MultiTrans based on splicing graphs reported 16.6% fewer candidates than TransLiG while maintaining a similar number of reconstructed transcripts as TransLiG (within 1.3% difference), which indicates the superior performance of MultiTrans over TransLiG in this dataset.

As shown in above experiments, MultiTrans can be successfully applied to different graphs (assembly graphs by rnaSPAdes and splicing graphs by TransLiG) to increase the number of reconstructed transcripts while reducing the number of transcript candidates. Besides, MultiTrans algorithm is very quick in most cases. For example, it only costs 30 minutes and $287Mb$ when takes the splicing graphs constructed by TransLiG on real human dataset (SRR5133163) as input.

5 Conclusion and Discussion

In this paper, we introduced an efficient transcriptome assembly algorithm, MultiTrans, which formulated the transcripts assembly problem into a mixed integer linear programming. We applied the MultiTrans algorithm to the assembly graphs constructed by rnaSPAdes and splicing graphs constructed by TransLiG, respectively. Experiments on both simulated and real datasets show that MultiTrans significantly improves the performances of rnaSPAdes and TransLiG when using the same input graphs. While rnaSPAdes and TransLiG recover paths by iteratively in a local or greedy fashion, MultiTrans seeks a global optimal solution via a mixed integer linear programming that takes into consideration of the number of paths, sequencing depth, and paired-end information, simultaneously. We believe that MultiTrans algorithm also applies to reference-based transcriptome assembly problem and may be extended to handle metagenomics assembly in the future.

Author Contributions

Yu Lin, Jin Zhao, Haodi Feng, and Daming Zhu contributed to the design of the study. Jin Zhao implemented MultiTrans and performed experiments. Jin Zhao, Yu Lin, and Haodi Feng wrote the manuscript. All authors read and approved the final manuscript.

Table 5: Benchmarking of rnaSPAdes, TransLiG, and MultiTrans on Rice dataset (SRR12147608)

Parameter	rnaSPAdes (Assembly graph)	MultiTrans (Assembly graph)	TransLiG (Splicing graph)	MultiTrans (Splicing graph)
100%-reconstructed	10,379	**11,086**	9,645	**9,652**
95%-reconstructed	12,483	**13,195**	**11,699**	11,649
90%-reconstructed	13,696	**14,329**	**12,893**	12,819
85%-reconstructed	14,584	**15,141**	**13,853**	13,689
80%-reconstructed	15,415	**15,849**	**14,750**	14,556
candidates	64,853	**47,051**	60,818	**50,698**

Table 6: Information of assembly graphs and splicing graphs on Human dataset and Mouse dataset. The average numbers of vertices, edges and paths correspond to the values in non-trivial graphs.

Parameter	Human dataset		Mouse dataset	
	assembly graph	splicing graph	assembly graph	splicing graph
Num of non-trivial graphs	25,162	11,085	21,189	4,474
average number of vertices	4.533	6.423	4.163	4.869
average number of edges	3.842	7.602	3.148	5.360
average number of paths	6.888	16.504	4.041	6.751

Funding

This work is supported by National Natural Science Foundation of China under No. 61672325, No. 61472222, No. 61732009, and No. 61761136017.

References

Baralle, F. E. and Giudice, J. (2017). Alternative splicing as a regulator of development and tissue identity. *Nature Reviews Molecular Cell Biology*, **18**(7), 437.

Biamonti, G. *et al.* (2019). Alternative splicing in alzheimer's disease. *Aging clinical and experimental research*, pages 1–12.

Birol, I. *et al.* (2009). De novo transcriptome assembly with abyss. *Bioinformatics*, **25**(21), 2872–2877.

Bushmanova, E. *et al.* (2019). rnaspades: a de novo transcriptome assembler and its application to rna-seq data. *GigaScience*, **8**(9), giz100.

Camacho, C. *et al.* (2009). Blast+: architecture and applications. *BMC bioinformatics*, **10**(1), 421.

Canzar, S. *et al.* (2016). Cidane: comprehensive isoform discovery and abundance estimation. *Genome biology*, **17**(1), 16.

Chang, Z. *et al.* (2015). Bridger: a new framework for de novo transcriptome assembly using rna-seq data. *Genome biology*, **16**(1), 30.

Climente-González, H. *et al.* (2017). The functional impact of alternative splicing in cancer. *Cell reports*, **20**(9), 2215–2226.

Gatter, T. and Stadler, P. F. (2019). Ryūtō: network-flow based transcriptome reconstruction. *BMC bioinformatics*, **20**(1), 190.

Grabherr, M. G. *et al.* (2011). Full-length transcriptome assembly from rna-seq data without a reference genome. *Nature biotechnology*, **29**(7), 644.

Griebel, T. *et al.* (2012). Modelling and simulating generic rna-seq experiments with the flux simulator. *Nucleic acids research*, **40**(20), 10073–10083.

Guttman, M. *et al.* (2010). Ab initio reconstruction of cell type–specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincrnas. *Nature biotechnology*, **28**(5), 503.

Heber, S. *et al.* (2002). Splicing graphs and est assembly problem. *Bioinformatics*, **18**(suppl_1), S181–S188.

Kelemen, O. *et al.* (2013). Function of alternative splicing. *Gene*, **514**(1), 1–30.

Kovaka, S. *et al.* (2019). Transcriptome assembly from long-read rna-seq alignments with stringtie2. *Genome Biology*, **20**(1), 1–13.

Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*.

Lin, Y.-Y. *et al.* (2012). Cliiq: Accurate comparative detection and quantification of expressed isoforms in a population. In B. Raphael and J. Tang, editors, *Algorithms in Bioinformatics*, pages 178–189, Berlin, Heidelberg. Springer Berlin Heidelberg.

Liu, J. *et al.* (2016a). Binpacker: packing-based de novo transcriptome assembly from rna-seq data. *PLoS computational biology*, **12**(2), e1004772.

Liu, J. *et al.* (2016b). Transcomb: genome-guided transcriptome assembly via combing junctions in splicing graphs. *Genome biology*, **17**(1), 213.

Liu, J. *et al.* (2019). Translig: a de novo transcriptome assembler that uses line graph iteration. *Genome biology*, **20**(1), 81.

Liu, P. *et al.* (2016c). Integrative analysis with chip-seq advances the limits of transcript quantification from rna-seq. *Genome research*, **26**(8), 1124–1133.

Mangul, S. *et al.* (2012). An integer programming approach to novel transcript reconstruction from paired-end rna-seq reads. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 369–376.

Mangul, S. *et al.* (2013). Flexible approach for novel transcript reconstruction from rna-seq data using maximum likelihood integer programming. In *5th International Conference on Bioinformatics and Computational Biology 2013, BICoB 2013*, pages 25–33.

Maretty, L. *et al.* (2014). Bayesian transcriptome assembly. *Genome biology*, **15**(10), 501.

Mezlini, A. M. *et al.* (2013). ireckon: simultaneous isoform discovery and abundance estimation from rna-seq data. *Genome research*, **23**(3), 519–529.

Optimization, G. (2014). Inc.,"gurobi optimizer reference manual," 2015.

Ozsolak, F. and Milos, P. M. (2011). Rna sequencing: advances, challenges and opportunities. *Nature reviews genetics*, **12**(2), 87–98.

Pan, Q. *et al.* (2008). Deep surveying of alternative splicing complexity in the human transcriptome by high-throughput sequencing. *Nature genetics*, **40**(12), 1413.

Paronetto, M. P. *et al.* (2016). Alternative splicing and cell survival: from tissue homeostasis to disease. *Cell Death & Differentiation*, **23**(12), 1919–1929.

Peng, Y. *et al.* (2013). Idba-tran: a more robust de novo de bruijn graph assembler for transcriptomes with uneven expression levels. *Bioinformatics*, **29**(13), i326–i334.

Pevzner, P. A. *et al.* (2001). An eulerian path approach to dna fragment assembly. *Proceedings of the national academy of sciences*, **98**(17), 9748–9753.

Prjibelski, A. D. *et al.* (2014). Exspander: a universal repeat resolver for dna fragment assembly. *Bioinformatics*, **30**(12), i293–i301.

Safikhani, Z. *et al.* (2013). Ssp: An interval integer linear programming for de novo transcriptome assembly and isoform discovery of rna-seq reads. *Genomics*, **102**(5), 507 – 514.

Safikhani, Z. *et al.* (2017). Gene isoforms as expression-based biomarkers predictive of drug response in vitro. *Nature communications*, **8**(1), 1126.

Schulz, M. H. *et al.* (2012). Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. *Bioinformatics*, **28**(8), 1086–1092.

Shao, M. and Kingsford, C. (2017a). Accurate assembly of transcripts through phase-preserving graph decomposition. *Nature biotechnology*, **35**(12), 1167.

Shao, M. and Kingsford, C. (2017b). Theory and a heuristic for the minimum path flow decomposition problem. *IEEE/ACM transactions on computational biology and bioinformatics*, **16**(2), 658–670.

Song, L. and Florea, L. (2013). Class: constrained transcript assembly of rna-seq reads. In *BMC bioinformatics*, volume 14, page S14. Springer.

Tomescu, A. I. *et al.* (2013). A novel min-cost flow method for estimating transcript expression with rna-seq. In *BMC bioinformatics*, volume 14, page S15. Springer.

Trapnell, C. *et al.* (2012). Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature protocols*, **7**(3), 562.

Xie, Y. *et al.* (2014). Soapdenovo-trans: de novo transcriptome assembly with short rna-seq reads. *Bioinformatics*, **30**(12), 1660–1666.

Zhao, J. *et al.* (2018). Isotree: A new framework for de novo transcriptome assembly from rna-seq reads. *IEEE/ACM transactions on computational biology and bioinformatics*.

Zhao, J. *et al.* (2019). Dta-sist: de novo transcriptome assembly by using simplified suffix trees. *BMC bioinformatics*, **20**(25), 1–12.

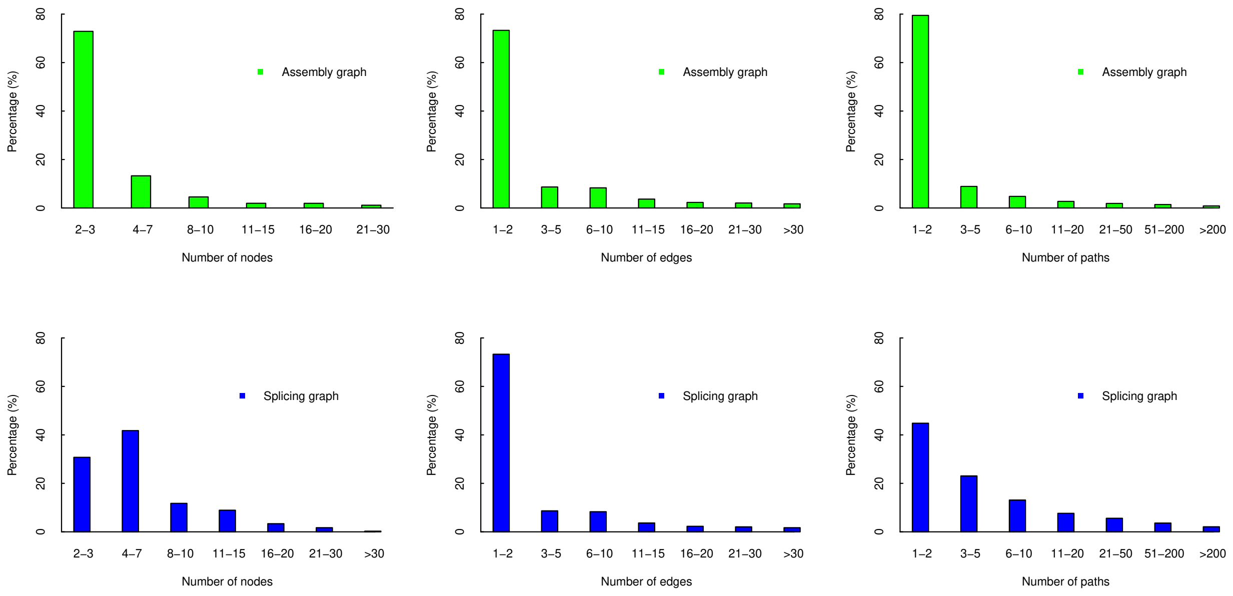

Figure 3: 3-strain).