The Pennsylvania State University
The Graduate School

BRIDGING PAIRED-END RNA-SEQ READS

A Thesis in
Computer Science and Engineering
by
Xiang Li

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2021

The thesis of Xiang Li was reviewed and approved by the following:

Mingfu Shao
Assistant Professor of Computer Science and Engineering
Thesis Advisor

Paul Medvedev
Associate Professor of Computer Science and Engineering

Chita R. Das
Professor of Computer Science and Engineering
Head of the department of Computer Science and Engineering

Abstract

The widely-used high-throughput RNA-sequencing technologies (RNA-seq) usually produce paired-end reads. We explore if full fragments can be computationally reconstructed from the sequenced two ends—a problem here we refer to as *bridging*. Solving this problem provides longer, more informative RNA-seq reads, and hence benefits downstream RNA-seq analysis such as transcriptome assembly and expression quantification. However, bridging is a challenging and complicated task owing to alternative splicing, transcript noises, and sequencing errors. It remains unclear if the data itself provides sufficient information for accurate bridging, let alone proper models and efficient algorithms that characterize and determine the true bridges.

We proposed a novel mathematical formulation to seek a path in a compacted de Bruijn graph for bridging such that its bottleneck weight is maximized. This formulation characterizes true bridges and is efficient in filtering out false bridges. This formulation admits optimal substructure property, and hence efficient dynamic programming algorithms can be designed. We designed a new truncated Dijkstra's algorithm for this problem, and proposed a novel algorithm that reuses the shortest path tree to avoid running the truncated Dijkstra's algorithm from scratch for all vertices for further speeding up. These innovative techniques result in scalable algorithms that can bridge all paired-end reads in a compacted de Bruijn graph with millions of vertices.

Our experiments showed that paired-end RNA-seq reads can be accurately bridged to a large extend. High precision was observed with varied sensitivity in both simulation and real data.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This work is partly supported by the US National Science Foundation DBI-2019797 and the US National Institutes of Health R01HG011065. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation and National Institutes of Health.

Chapter 1
Introduction

The established high-throughput RNA sequencing technologies (RNA-seq) enables global and accurate measurement of isoform-level gene activities. The second generation short-reads RNA-seq, which remains *de facto* standards for most expression studies, produces paired-end reads. Such data reports sequences of the two ends of a fragment of a RNA molecule, but misses the middle portion of the fragment. The fact that the two ends are from the same molecule and that the length of fragments follows a certain distribution (through fragment size selection) provides valuable long-range information in determining complicated splicing variants, and has been incorporated into various RNA-seq analysis tools and software to improve accuracy, including splicing-aware alignment (e.g., STAR [1], HISAT2 [2], SpliceMap [3]), expression quantification (e.g., Salmon [4], kallisto [5], RSEM [6]), transcript assembly (e.g., StringTie [7], TransComb [8], Scallop [9]), fusion detection (e.g., FuSeq [10], STAR-Fusion [11], SQUID [12]), and alternative splicing quantification (e.g., DARTS [13], leafCutter [14]), among many others.

We explore computationally inferring full fragments from given paired-end RNA-seq reads, a problem we refer to as *bridging*. We believe solving this problem could benefit various downstream RNA-seq analysis such as transcript assembly, isoform-level expression quantification, and splicing quantification. The inferred full fragments likely contain more splicing junctions than individual reads, and hence provide additional long-range information that helps resolve more complicated splicing variants in transcript assembly. Longer sequences will be less ambiguously located to transcripts expressed from the same gene or homologous genes, and hence improves isoform quantification. The reconstructed full fragments may reveal missing junctions in the unsequenced portion, which may lead to more accurate estimation of junction abundance, and hence improves splicing quantification.

However, bridging is a challenging task. It remains unclear if the reads themselves

contain sufficient information to enable accurate bridging. Due to the complicated mechanism of alternative splicing and the dynamic nature of transcription and mRNA degradation, different types of splicing junctions will be captured by RNA-seq reads. According to our experiments, in many gene loci, thousands of different splicing junctions can be observed (a majority of them are with low abundance though). For those loci the possible ways of bridging a pair of reads will be enormous, and it's unclear what signal can be used to determine the true bridge. Besides, sequencing errors also produce false bridges, making this task more challenging to solve.

Figure 1.1. The full Fragment and paired-end reads.

We explore modeling above bridging problem as a graph problem. Sequencing reads can be organized by a graph data structure de Bruijn graph (dBG) or compacted de Bruijn graph (cdBG) [15]. Naturally, the sequenced two ends of a fragment can be mapped to the graph and then represented as a pair of paths (two lists of vertices or edges) in the graph. Therefore, the bridging task becomes to find a path that connects the two paths. (Our work follows this framework for bridging; see Chapter 2.) However, the resulting graphs are often erroneous due to transcript noises and sequencing errors. It remains open that what is a good characterization of the true path in bridging—in other words, what objective function should be in formulating bridging as a graph optimization problem. Besides, the resulting dBG or cdBG may contain millions of vertices. This urges scalable algorithms for bridging.

Although many existing RNA-seq analysis tools use paired-end information, limited efforts have been made to directly reconstruct full fragments. Existing method for reference-based bridging includes MapPER [16]. MapPER implemented a probabilistic framework: starting from splicing junctions of all end reads, MapPER constructed potential splicing paths connecting paired-end reads; an expectation maximization method then assigned likelihood values to all splice junctions and assigned the most probable alignment for each fragment. And to our best knowledge, no available tool for RNA-seq bridging without using any reference information exists.

Chapter 2
Algorithms

We formulate bridging as a new optimization problem (Section 2.1). We then introduce our method which is divided into three modules: the construction of cdBG in section 2.2, mapping reads to cdBG in Section 2.3 and the bridging algorithms in Section 2.4.

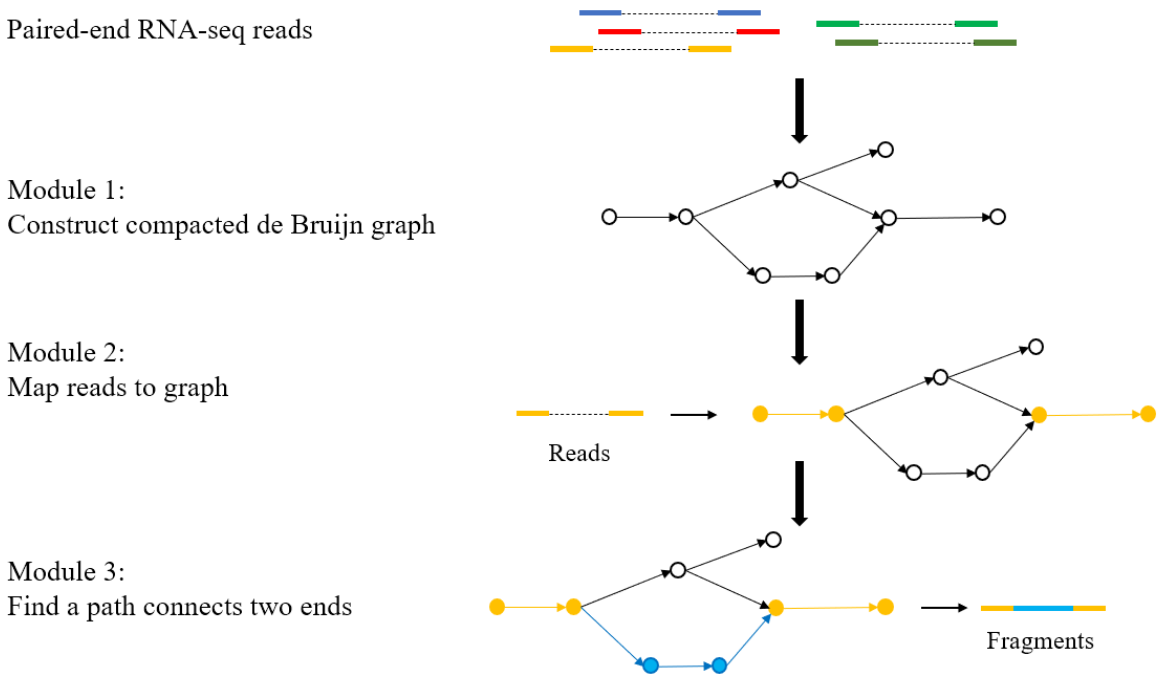

Figure 2.1. The pipeline of the method.

2.1 Formulation

In the first step of our framework, the given RNA-seq data can be organized by a compacted de Bruijn graph (cdBG) (see Section 2.2 for their constructions). Let $G = (V, E)$ be the compacted de Bruijn graph. Let f be a fragment, for which we know its two sequenced ends. Each end of f can be represented as a path in G, and f can then be represented as a pair of paths in G. Note that multiple fragments may correspond to the same pair of paths in G; we cluster them into *equivalent classes*. Formally, an equivalent class is a pair of paths (p_1, p_2) in G that represent all fragments with two ends being corresponding to p_1 and p_2 respectively.

Let $F = (p_1 = (a_1, a_2, \cdots, a_m), p_2 = (b_1, b_2, \cdots, b_n))$ be an equivalent class, where path p_1 is represented as a list of vertices $(a_1, \cdots, a_m)$ in the graph (the same for p_2). The problem of bridging fragments in F becomes to find a path in G from a_m to b_1; we call such path as a *bridging path*. We assume that, all fragments in an equivalent class have the same true bridging path. This is because fragments in an equivalent class are similar, as their two ends are mapped to exactly the same list of vertices in the graph. Algorithmically, this assumption allows us to reduce computational load, as all fragments can be bridged in a single run.

We explore what's a good formulation to find the best bridging path. The main signal we have is the abundances (i.e., the numbers of reads that support vertices/edges of the cbdg, modeled as weights of vertices/edges; see Section 2.2). Intuitively, a bridging path supported by more reads are more likely to be the true bridge. We determined that, a formulation that seeks a bridging path with maximized *bottleneck* weight is appropriate for bridging problem, and performs well on experimental comparisons. Below we describe this formulation.

We define a full ordering of all bridging paths w.r.t. an equivalent class $F = (p_1 = (a_1, a_2, \cdots, a_m), p_2 = (b_1, b_2, \cdots, b_n))$. Let q_1 and q_2 be two arbitrary paths from a_m to b_1 in G. Let w_1^i (resp. w_2^i) be the ith smallest weight in path q_1 (resp. q_2). We say q_1 is *more reliable* than q_2, if there exists an integer k such that $w_1^i = w_2^i$ for all $1 \leq i < k$, and $w_1^k > w_2^k$. We now formulate the bridging problem as to find the most reliable path. Intuitively, we seek a path q from a_m to b_1 in G such that the smallest weight in this path is maximized, and in case there are multiple paths with maximized smallest weight, among them we seek the one whose second smallest weight is maximized, and so on.

We believe this formulation is appropriate for bridging paired-end RNA-seq reads. First, by maximizing bottleneck weight, the weakest point of the bridging path is

supported strongest, which avoids being dominated by vertices/edges with large weights. Second, through this formulation, false paths coming from sequencing errors can be efficiently excluded, as they usually exhibits low bottleneck abundance. We also explored formulations such as maximizing the *sum* of weights in a path but they didn't perform as well as this one.

We emphasize that this formulation satisfies the *optimal substructure* property. Specifically, if $a_m \to v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k} \to b_1$ is the most reliable path, then $a_m \to v_{i_1} \to v_{i_2} \to \cdots \to v_{i_k}$ is the most reliable path from a_m to v_{i_k}. This implies that polynomial-time dynamic programming algorithms can be designed to find the optimal solution.

The bottleneck weight (smallest weight of the optimal path) in this formulation can be used as a filtering criterion to decide if a bridging path is true. We set this threshold with different numbers to balance sensitivity and precision in bridging (see Table 3.1 and Table 3.2).

2.2 Constructing Compacted de Bruijn Graph (cdBG)

We use cdBG to represent the given paired-end RNA-seq reads. See Figure 2.2. In the de Bruijn graph (dBG), each vertex represents a distinct k-mer, and its weight equals to the number of appearance this k-mer in the reads. The corresponding cdBG is defined as concatenating each simple path (i.e., every vertex in it except the first and the last one has in-degree of 1 and out-degree of 1) of the dBG as a single vertex (the resulting sequence is called a *unitig*). To comply with our formulation of finding the most reliable path, we assign the weight of each vertex in cdBG as the smallest weight of the corresponding simple path in dBG.

In implementation, we directly construct the cdBG and calculate vertex weights, instead of explicitly constructing dBG as an intermediate. Specifically, we use library Bifrost [17] to build the cdBG. In order to assign vertex weights, we first build a table that stores the frequency of each k-mer; we then examine all k-mers in this vertex (a unitig of length l contains $l - k + 1$ k-mers) and assign the smallest frequency as the weight of the vertex.

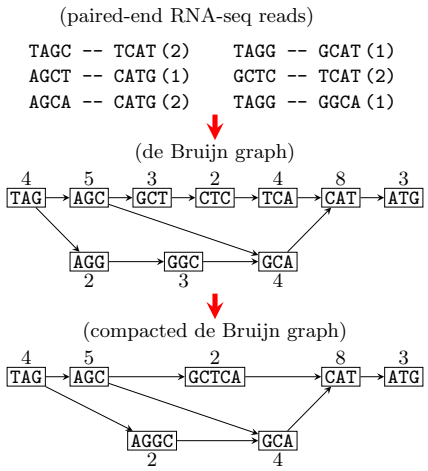

Figure 2.2. Example of a cdBG constructed from paired-end RNA-seq reads.

2.3 Mapping Reads to cdBG

After constructing the cdBG, we need to map all the paired-end reads to the graph, which means for each read, we want to find the corresponding path in the graph that represents the read. Every pair of reads can be mapped to a pair of paths in cdBG. As we described in Section 2.1, all the pairs of paths can be clustered into equivalent classes to decrease computing burdens. More specifically, here we used the pair of the last vertex a_m in p_1 and the first vertex b_1 in p_2 to represent an equivalent class.

Therefore, after mapping all the reads to cdBG, we will get a list of pairs of vertices, in each pair, there are one source vertex a_m and one sink vertex b_1 which are needed for the bridging algorithm.

2.4 Bridging Algorithms

Following our formulation, we aim to find the most reliable bridging path for all equivalent classes. From Section 2.1, we know that our formulation satisfies the *optimal substructure* property, which allows us to use dynamic programming algorithm to calculate the most reliable path. More precisely, we can use *Dijkstra's algorithm* to solve our problem but change the shortest path to our formulation the most reliable path.

However, as the cdBG constructed from a typical RNA-seq sample may consist of millions vertices (see Table 3.3), it is simply not affordable to run a standard *Dijkstra's algorithm* for every possible source vertices (note that a single run of *Dijkstra's algorithm* starting from a vertex will find the most reliable path connecting it to all other vertices).

We propose two algorithmic innovations to address this main challenge in *de novo* bridging. First, we implemented a *truncated Dijkstra's algorithm* to find the most reliable bridging path starting from a starting vertex a_m to any other vertex up to a certain length D (default value is $400 + 2L$ where L is read length). This is to incorporate the prior knowledge that fragment length follows an empirical distribution and an upper bound of fragment length can be assumed. In our truncated Dijkstra's algorithm, for each vertex v we maintain the total length of the most reliable path from the current starting a_m to v, and when such length for v reaches D, we won't extend v in the Dijkstra's algorithm. Similar to the algorithm in reference-based bridging, a list that stores the smallest M weights also needs to be maintained.

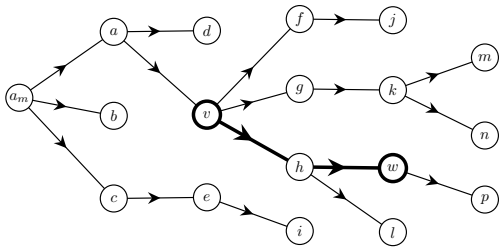

Figure 2.3. Illustrating the shortest path tree starting from vertex a_m. Consider subtree rooted at v. Then for any vertex in it, say w, the path from v to w in this subtree is the most reliable path from v to w in the cdBG.

The second algorithmic innovation is that we reuse the optimal solutions obtained for the current starting vertex a_m to construct the optimal solutions for next starting vertex a'_m. This allows us to avoid running above truncated Dijkstra's algorithm from scratch for all possible starting vertices. Specifically, for the current starting vertex a_m, we maintain a *shortest path tree*, denoted as $T(a_m)$, to store the most reliable paths from a_m to all other vertices (up to length D), i.e., the unique path in T from root a_m to any vertex v gives the most reliable path from a_m to v in the cdBG. See Figure 2.3. This tree can be constructed in linear time with a tracing back procedure after running the truncated Dijkstra's algorithm. We note that, again according to the property of optimal substructures, any subtree of $T(a_m)$, say the one rooted at u, also gives the most reliable path from u to any other vertex w in the subtree. This suggests we can reuse the subtree rooted at u to calculate all reliable paths starting from u (and then construct the corresponding shortest path tree). More specifically, in implementation we directly load the subtree rooted at u to the priority queue (recall that the core data structure of Dijkstra's algorithm is a priority queue that gets updated iteratively). In other words, for next starting vertex u we run the truncated Dijkstra's algorithm in the

middle rather than from scratch, as we already know the optimal solutions for a subset of vertices (i.e., those in the subtree of $T(a_m)$ rooted at u). To benefit from this property to the largest extent, we determine the starting vertex whose subtree is largest as the next one to bridge.

Chapter 3
Results

3.1 Resulting Tools

We developed a tool of the bridging algorithm, available at https://github.com/Shao-Group/rnabridge-denovo. The input for it is sequencing reads in fastq/fasta format, and it generates sequences of full fragments again in fasta format.

3.2 Datasets

We use 2 datasets to evaluate the accuracy of bridging and its effectiveness in improving downstream RNA-seq analysis. The first dataset includes 80 paired-end RNA-seq samples simulated using Flux-Simulator [18]. We vary two parameters in the simulation: the average length of fragments (300 and 500 were used) and the read length (75 and 100 were used). For each combination of parameters, we independently simulated 20 samples. The number of reads in samples with fragment length being 300 and 500 are roughly 150M and 90M, respectively. The second dataset was previously used in the Scallop paper [9]: it contains 10 biological RNA-seq samples.

3.3 Simulation Data Results

We first evaluate the accuracy of our algorithm using simulation data, for which the ground-truth are available. We define a bridged fragment is correct only if it's exactly the same as the ground-truth fragment. The sensitivity is defined as the number of correctly bridged fragments divided by the total number of fragments (i.e., paired-end reads), and precision is defined as the number of correctly bridged fragments divided by the total

number of bridged fragments.

The results are summarized in Table 3.1. The bottleneck threshold used to filter bridges is set to 5 for all simulated samples. Overall our tool exhibits high accuracy. The accuracy drops with long fragment length. This is expected as in this case the missing portion is longer and therefore harder to bridge. This may also be partly due to that the coverage of these simulated samples is low. Higher accuracy is observed when the read length is longer at the same fragment length, again as expected.

Table 3.1. Averaged accuracy of rnabridge-denovo on the simulated RNA-seq samples in dataset 1.

flen	rlen	bottleneck	sensitivity (%)	precision (%)
300	75	5	80.7	91.8
300	100	5	85.7	92.5
500	75	5	66.6	85.4
500	100	5	76.3	86.9

3.4 Real Data Results

We then evaluate using the biological data in real dataset 2. As we don't have ground-truth for them, we again use the sequences in the reference transcriptome to evaluate. We align all the bridged fragments to reference using BLAT [19]. We define a bridged fragment is correct only if it is hit by one of the reference sequences with at least 95% sequence identity. The sensitivity and precision is defined the same as in evaluating with simulation data.

The results are summarized in Table 3.2. Overall the precision keeps high but the sensitivity varies quite a lot. This is because we prioritize precision by setting a high bottleneck-threshold: 20 for samples with less than 50M paired-end reads and 100 otherwise. Comparing with simulated data, biological data are more noisy and harder to bridge.

3.5 Running Time Analysis

As it's mentioned in Chapter 2, Our tool consists of 3 modules. Module 1 uses Bifrost [17] to build the cdBG; module 2 aligns all paired-end reads to the cdBG and calculates the weight of unitigs; module 3 implements our core bridging algorithm described in

Table 3.2. The accuracy of rnabridge-denovo on the 10 RNA-seq samples in dataset 2. The number of paired-end reads are given in unit of million.

SRA ID	#paired-reads	bottleneck	sensitivity (%)	precision (%)
SRR307903	36.0M	20	65.3	91.8
SRR307911	41.4M	20	42.7	91.3
SRR315323	30.3M	20	27.9	87.9
SRR315334	39.5M	20	49.5	93.3
SRR545723	38.9M	20	42.7	81.6
SRR387661	124M	100	58.2	90.1
SRR534291	114M	100	64.3	93.5
SRR534307	165M	100	50.6	74.7
SRR534319	76.6M	100	24.6	69.5
SRR545695	119M	100	37.6	75.5

Section 2.4. We show the breakdown of the CPU time of the 3 modules together with the size of the resulting cdBGs on the 10 biological samples in Table 3.3. Note that module 3 takes the least CPU time among 3 modules, which proves the efficiency and scalability of the optimized core bridging algorithms.

Table 3.3. The size (number of vertices and edges; in unit of million) of the cdBGs and the CPU time (in minutes) measured for the 3 modules of rnabridge-denovo running on the 10 samples in dataset 2.

SRA ID	size of cdBG		CPU time (in minutes)		
	#vertices	#edges	module 1	module 2	module 3
SRR307903	5.15M	6.99M	149	57	26
SRR307911	8.11M	10.71M	175	66	75
SRR315323	7.68M	9.45M	120	40	63
SRR315334	6.35M	10.16M	158	67	37
SRR387661	18.71M	27.36M	515	223	141
SRR534291	15.14M	26.67M	685	370	128
SRR534307	34.29M	56.00M	1038	561	281
SRR534319	16.48M	22.09M	311	118	121
SRR545695	19.43M	27.49M	479	201	174
SRR545723	13.47M	16.04M	243	99	159

Chapter 4
Conclusions and Future Work

4.1 Conclusions

In this thesis, we mainly focus on the bridging problem, i.e., to reconstruct the full length fragments from paired-end reads. In order to solve this problem, we build a compacted *de Bruijn* Graph from all the reads, and further change the problem into a path-finding problem in cdBG. The experimental results in chapter 3 shows that the simulated RNA-seq data do provide sufficient information for accurate bridging. As for the real RNA-seq data, it provide enough information to bridge a large part of the reads. The results also proves that our problem formulation and method pipeline works well for the bridging problem.

The two important innovation in this work are the problem formulation for bridging and the bridging algorithm. Before we conducted this research, it remains unclear that what the proper formulation is for the bridging problem. Here we define the most reliable path with highest bottleneck weight and use experimental results to show it works well for bridging RNA-seq reads. Besides, we proposed a novel truncated Dijkstra's algorithm and the optimal path tree reuse technique to speed up, which help us to solve this bridging problem efficiently.

4.2 Future Work

We explored if bridging could improve *de novo* transcript assembly. To this end we piped the bridged fragments to one leading assembler TransLiG [20], but only observed marginal improvement. This may be because TransLiG is not optimized to make use of mixed short and long sequences. Developing a new *de novo* assembler that can fully use

such bridged data is on our research agenda. Experimenting if *de novo* bridging could improve isoform quantification and splicing quantification is also an interesting future research topic for us.

The sensitivity of our method is low on some biological samples. One reason is that we use a high bottleneck-threshold to keep high precision, which consequently disconnects paired-end reads in low-coverage gene loci. We are developing a post-bridging algorithm to use full-range information in the reads to decide if a bridged fragment is correct (rather than just using a bottleneck-threshold), in a hope of keeping high precision while improving sensitivity. Specifically, note that the cdBG is not a loss-free representation of sequencing reads, as it breaks reads into k-mers, and any phasing information beyond $(k+1)$-mer is not represented. Let s be the bridged sequence of a fragment f constructed using above algorithm. We can examine each sliding window of length L, where L is the read length, and determine the number of input reads that are identical to this L-mer (i.e., these reads *support* this L-mer). This gives a *supporting profile*, a vector of length $|s| - L + 1$ for this bridged sequence. Intuitively, a profile with few 0s suggests that the bridged sequence is likely a true one, while a long consecutive 0s in the profile suggests a false bridge. We are experimenting if such supporting profile could lead to more efficient algorithms in boosting bridging accuracy.

Bibliography

[1] Dobin, A., C. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T. Gingeras (2013) "STAR: ultrafast universal RNA-seq aligner," *Bioinformatics*, **29**(1), pp. 15–21.

[2] Kim, D., B. Langmead, and S. Salzberg (2015) "HISAT: a fast spliced aligner with low memory requirements," *Nat. Methods*, **12**(4), pp. 357–360.

[3] Au, K., H. Jiang, L. Lin, Y. Xing, and W. Wong (2010) "Detection of splice junctions from paired-end RNA-seq data by SpliceMap," *Nucleic Acids Res.*, **38**(14), pp. 4570–4578.

[4] Patro, R., G. Duggal, M. Love, R. Irizarry, and C. Kingsford (2017) "Salmon provides fast and bias-aware quantification of transcript expression," *Nat. Methods*, **14**, pp. 417–419.

[5] Bray, N., H. Pimentel, P. Melsted, and L. Pachter (2016) "Near-optimal probabilistic RNA-seq quantification," *Nat. Biotechnol.*, **34**(5), pp. 525–527.

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

[7] Pertea, M., G. Pertea, C. Antonescu, T.-C. Chang, J. Mendell, and S. Salzberg (2015) "StringTie enables improved reconstruction of a transcriptome from RNA-seq reads," *Nat. Biotechnol.*, **33**(3), pp. 290–295.

[8] Liu, J., T. Yu, T. Jiang, and G. Li (2016) "TransComb: genome-guided transcriptome assembly via combing junctions in splicing graphs," *Genome Biol.*, **17**(1), p. 213.

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

[10] Vu, T. N., W. Deng, Q. T. Trac, S. Calza, W. Hwang, and Y. Pawitan (2018) "A fast detection of fusion genes from paired-end RNA-seq data," *BMC Genomics*, **19**(1), pp. 1–13.

[11] Haas, B. J., A. Dobin, B. Li, N. Stransky, N. Pochet, and A. Regev (2019) "Accuracy assessment of fusion transcript detection via read-mapping and de novo fusion transcript assembly-based methods," *Genome Biology*, **20**(1), p. 213.

[12] Ma, C., Shao, M., and C. Kingsford (2018) "SQUID: transcriptomic structural variation detection from RNA-seq," *Genome Biol.*, **19**(1), p. 52.

[13] Zhang, Z., Z. Pan, Y. Ying, Z. Xie, S. Adhikari, J. Phillips, R. P. Carstens, D. L. Black, Y. Wu, and Y. Xing (2019) "Deep-learning augmented RNA-seq analysis of transcript splicing," *Nature Methods*, **16**(4), pp. 307–310.

[14] Li, Y. I., D. A. Knowles, J. Humphrey, A. N. Barbeira, S. P. Dickinson, H. K. Im, and J. K. Pritchard (2018) "Annotation-free quantification of RNA splicing using LeafCutter," *Nature Genetics*, **50**(1), pp. 151–158.

[15] Chikhi, R., A. Limasset, and P. Medvedev (2016) "Compacting de Bruijn graphs from sequencing data quickly and in low memory," *Bioinformatics*, **32**(12), pp. i201–i208.

[16] Hu, Y., K. Wang, X. He, D. Y. Chiang, J. F. Prins, and J. Liu (2010) "A probabilistic framework for aligning paired-end RNA-seq data," *Bioinformatics*, **26**(16), pp. 1950–1957.

[17] Holley, G. and P. Melsted (2020) "Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs," *Genome Biology*, **21**(1), pp. 1–20.

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

[19] Kent, W. J. (2002) "BLAT—the BLAST-like alignment tool," *Genome research*, **12**(4), pp. 656–664.

[20] Liu, J., T. Yu, Z. Mu, and G. Li (2019) "TransLiG: a de novo transcriptome assembler that uses line graph iteration," *Genome Biology*, **20**(1), pp. 1–9.