On *de novo* Bridging Paired-end RNA-seq Data

Xiang Li
xpl5168@psu.edu
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, Pennsylvania, USA

Mingfu Shao
mxs2589@psu.edu
Department of Computer Science and Engineering
Huck Institutes of the Life Sciences
The Pennsylvania State University
University Park, Pennsylvania, USA

ABSTRACT

The high-throughput short-reads RNA-seq protocols often produce paired-end reads, with the middle portion of the fragments being unsequenced. We explore if the full-length fragments can be computationally reconstructed from the sequenced two ends in the absence of the reference genome—a problem here we refer to as *de novo* bridging. Solving this problem provides longer, more informative RNA-seq reads, and benefits downstream RNA-seq analysis such as transcript assembly, expression quantification, and splicing differential analysis. However, *de novo* bridging is a challenging and complicated task owing to alternative splicing, transcript noises, and sequencing errors. It remains unclear if the data provides sufficient information for accurate bridging, let alone efficient algorithms that determine the true bridges. Methods have been proposed to bridge paired-end reads in the presence of reference genome (called reference-based bridging), but the algorithms are far away from scaling for *de novo* bridging as the underlying compacted de Bruijn graph (cdBG) used in the latter task often contains millions of vertices and edges. 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 cdBG with millions of vertices. Our experiments showed that paired-end RNA-seq reads can be accurately bridged to a large extent. The resulting tool is freely available at https://github.com/Shao-Group/rnabridge-denovo.

CCS CONCEPTS

• **Theory of computation → Graph algorithms analysis**; • **Applied computing → Molecular sequence analysis**.

KEYWORDS

RNA-seq, *de novo* bridging, de Bruijn graph, transcript assembly, alternative splicing

ACM Reference Format:
Xiang Li and Mingfu Shao. 2023. On *de novo* Bridging Paired-end RNA-seq Data. In *14th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '23), September 3–6, 2023, Houston, TX, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3584371.3612987

BCB '23, September 3–6, 2023, Houston, TX, USA

ACM ISBN 979-8-4007-0126-9/23/09.
https://doi.org/10.1145/3584371.3612987

1 INTRODUCTION

The high-throughput RNA sequencing technologies (RNA-seq) enable accurate measurement of isoform-level gene activities and have been widely used in biological and biomedical research. The second generation short-reads RNA-seq, which remains *de facto* standards for most expression studies, often produces paired-end reads. Such data reports sequences of the two ends of a fragment of an RNA molecule, but misses the middle portion of the fragment. The fact that 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 [5], HISAT2 [10], SpliceMap [1]), isoform-level expression quantification (e.g., Salmon [17], kallisto [2], RSEM [12]), transcript assembly (e.g., StringTie [18], StringTie2 [11], TransComb [14], Scallop [20], and Scallop2 [22]), gene fusion detection (e.g., FuSeq [21], STAR-Fusion [7], SQUID [16]), and splicing quantification (e.g., DARTS [23], leafCutter [13]), among many others.

We explore computationally inferring the full-length fragments from paired-end RNA-seq reads, a problem we refer to as *bridging*. Solving this problem can substantially benefit downstream RNA-seq analysis such as transcript assembly, isoform quantification, and splicing quantification. Specifically, the inferred full-length 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 likely to be ambiguously located to transcripts expressed from the same gene or from homologous genes, and hence improves isoform quantification. The reconstructed full-length fragments may reveal missing junctions in the unsequenced portion, which will likely lead to a more accurate estimation of junction abundance, and hence improves splicing quantification.

The above bridging problem has been studied in a reference-based setting, implemented as part of the Scallop2 assembler [22]. In that method, the reads alignments of a gene locus are first organized by a *splice graph*, and reads are then presented as paths in the graph. Scallop2 proposed a formulation that seeks a path connecting the two ends of a paired-end read such that the *bottleneck* weight (defined as the smallest edge-weight) of the path is maximized. Scallop2 designed a dynamic programming algorithm,

and it was demonstrated to be efficient in improving the accuracy of reference-based assembly.

In this work, we explore the bridging problem in the *de novo* setting, i.e., without using a reference genome, termed *de novo* bridging. This is motivated by the RNA-seq analysis for non-model species, for which a good-quality reference genome is not yet available. The *de novo* bridging problem can also be modeled as a graph problem. Specifically, all RNA-seq reads can be organized by a de Bruijn graph (dBG) or compacted de Bruijn graph (cdBG) [3]. (In this paper we use cdBG.) Similar to the aforementioned reference-based bridging, the sequenced two ends of a fragment can be mapped to the cdBG, and the bridging problem amounts to finding a path that can connect the two ends in the graph. Given the proven efficiency of the formulation proposed in Scallop2 for reference-based bridging, here we adopt it for *de novo* bridging, i.e., to seek a connecting path in the cdBG such that the bottleneck weight is maximized. However, the dynamic programming algorithm in Scallop2 cannot be applied for *de novo* bridging, as it is designed for splice graphs which is acyclic (while cdBGs have cycles) and that it cannot scale for cdBGs which contain millions of vertices (while splice graphs typically contain hundreds of vertices or less).

We propose two algorithmic innovations to enable *de novo* bridging that scales to graphs with millions of vertices. First, we design a *truncated Dijkstra's algorithm* which can find the optimal path while taking into account the fragment length to speed up searching. Second, we propose to use shortest path tree to store the optimal solutions for a single vertex and propose an efficient algorithm to construct such shortest path tree for the next vertex by reusing the nodes and edges on previous trees. This allows us to avoid running the above truncated Dijkstra's algorithm from scratch for all possible vertices. Combined, the resulting algorithm can accurately bridge all paired-end reads in a couple of hours on typical RNA-seq samples (with millions of vertices in the underlying cdBG).

2 ALGORITHMS

Our approach for *de novo* bridging consists of three modules (see Figure 1): constructing the cdBG (Section 2.1), mapping reads to the cdBG (Section 2.2), and bridging the reads. The last module is organized as a formulation (Section 2.3) followed by the algorithms for solving the formulation (Section 2.4).

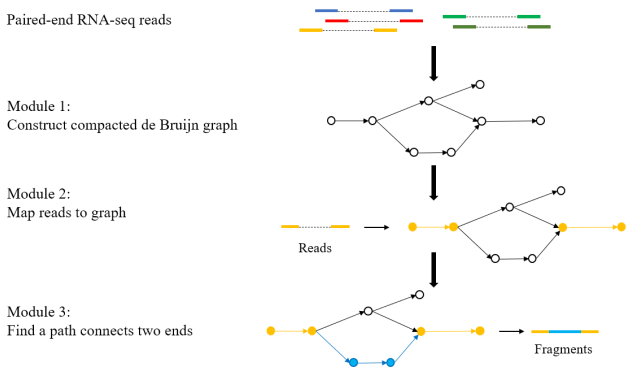

Figure 1: The pipeline of our method for *de novo* bridging.

2.1 Constructing Compacted de Bruijn Graph (cdBG)

We use cdBG to represent the given paired-end RNA-seq reads. See Figure 2. In the de Bruijn graph (dBG), each vertex represents a distinct k-mer, and its weight is equal to the number of the appearance 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 (see details in Section 2.3), we assign the weight of each vertex in cdBG as the smallest weight of the corresponding simple path in dBG.

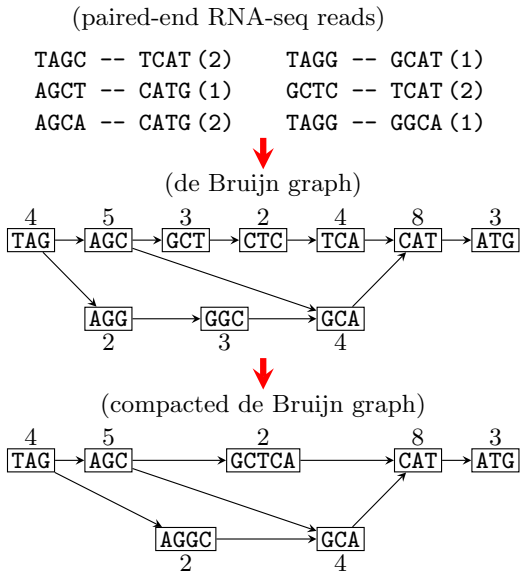

Figure 2: Example of a cdBG constructed from paired-end RNA-seq reads.

In implementation, we directly construct the cdBG and calculate vertex weights, instead of explicitly constructing dBG as an intermediate. Specifically, we use the library Bifrost [8] 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.

2.2 Mapping Reads and Constructing Equivalent Classes

After constructing the cdBG, we map all paired-end reads to the graph. For each end, we first call the findunitig function provided in the library Bifrost to fetch the vertex that matches the beginning of this end. Then we can traverse the graph by extending the following characters to get a list of vertices that this end is matched to. This procedure gives the path for each end in the cdBG.

Let $G = (V, E)$ be the cdBG and let f be a fragment. Each end of f can now 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*. In other words, an equivalent class is a pair of paths (p_1, p_2) in G that represent all fragments with two ends 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. The problem of bridging fragments in F becomes finding a path in G from a_m to b_1; such path is defined as a *bridging path*. We assume that all fragments in an equivalent class have the same true bridging path. 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, it allows us to reduce computational load, as all fragments can be bridged in a single run.

2.3 Formulation

We now formulate the *de novo* bridging as an optimization problem, using the same idea proposed in Scallop2 [22]. 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 graph 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 $1 \le i < k$, 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. This formulation has been showed to be accurate when applied for reference-based bridging.

We note 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 formulation forms the basis for designing efficient algorithms. 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 1 and Table 2).

2.4 Bridging Algorithms

Given the *optimal substructure* property, a straightforward dynamic programming can be designed (which essentially was used in Scallop2). However, as the cdBG constructed from a typical RNA-seq sample may consist of millions of vertices (see Table 3), above dynamic programming algorithm simply cannot scale. We propose to adopt the *Dijkstra's algorithm*. Dijkstra's algorithm is primarily used for solving the shortest path problem, but we modify it to find the most reliable path defined above. Specifically, we maintain an array $d[]$ to store the bottleneck weight on the most reliable path from the source to other vertices; each time we select the vertex with highest d-value using a priority queue. Then for each vertex j adjacent to i, $d[j]$ can be updated as $d[j] = \max(d[j], \min(d[i], e(i, j)))$ where $e(i, j)$ is the weight of the edge from vertex i to vertex j.

A single run of the above Dijkstra's algorithm starting from a vertex in the cdBG $G = (V, E)$ will find the most reliable path connecting it to all other vertices. It runs in $O(|V| \log |V|)$ time, already faster than the dynamic programming algorithm used in Scallop2, which runs in $O(|V'| \cdot |E'|)$ time where V' and E' are the vertices and edges of the splice graph.

To further speed it up, we propose two algorithmic innovations. 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. 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 further extend from v to any other vertices in the Dijkstra's algorithm.

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 the next starting vertex a'_m. This allows us to avoid running the 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 3. This tree can be constructed in linear time while 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 the 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.

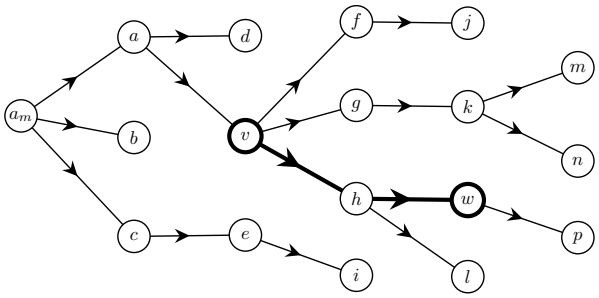

Figure 3: Illustrating the shortest path tree starting from vertex a_m. Consider subtree rooted at v. Then for any vertex in it, say w, the path from v to w in this subtree is the most reliable path from v to w in the cdBG.

3 RESULTS

3.1 Resulting Tool

The above algorithm was implemented, available at https://github.com/Shao-Group/rnabridge-denovo. The input files for this tool is the paired-end RNA-seq data in fastq/fasta format, and it generates sequences of full fragments again in fasta format. Since so far our method is the only one for *de novo* bridging, we could not compare it with other methods in the following experiments. In our experiments, we choose $k = 31$ when constructing cdBG. The upper bound of the bridging path D can be picked according to the distribution of the insert size so as to cover most of the fragments. We use a default value of $D = 400$, based on the ENCODE RNA-seq Data Standards [19].

3.2 Datasets

We use two datasets to evaluate the accuracy of bridging. The first dataset includes 80 paired-end RNA-seq samples simulated using Flux-Simulator [6]. We vary two parameters in the simulation: the average length of fragments (flen; 300 and 500) and the read length (rlen; 75 and 100). 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 [20]: it contains 10 biological RNA-seq samples.

3.3 Results on Simulation Data

We first evaluate the accuracy of our algorithm using simulation data, for which the ground-truth is available. A bridged fragment is correct only if it is 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 1. The bottleneck threshold used to filter bridges is set to 5 for all simulated samples. Our algorithm exhibits high accuracy, suggesting that the missing portion can be accurately bridged given solely the reads information and that our algorithm is able to use such information to bridge. The accuracy drops with long fragment length. This is expected as in this case the missing portion is longer and therefore harder to bridge. Higher accuracy is observed when the read length is longer at the same fragment length, again as expected.

Table 1: Averaged bridging accuracy on the simulated RNA-seq datasets.

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 Results on Real Data

We then evaluate the accuracy of our algorithm on real dataset. The paired-end reads are first processed by an error correction tool RECKONER [4]. As we do not have ground-truth for them, we use the sequences in the reference transcriptome to evaluate. We align all the bridged fragments to reference using BLAT [9]. 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 are defined the same as in evaluating with simulation data.

The results are summarized in Table 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 noisier and harder to bridge. We note that high precision is more desirable for downstream analysis in order not to introduce false positives. Given the high precision, even a small portion of correctly bridged paired-end reads are able to improve the accuracy of downstream applications. Users may also choose to adjust the bottleneck-threshold to balance the precision and sensitivity.

Table 2: The bridging accuracy on the 10 RNA-seq samples. The number of paired-end reads are given in unit of million.

SRA ID	#paired-reads	bottleneck	sensitivity (%)	precision (%)
SRR307903	36.0M	20	59.5	91.5
SRR307911	41.4M	20	52.8	91
SRR315323	30.3M	20	39.7	88.5
SRR315334	39.5M	20	60.2	93.3
SRR545723	38.9M	20	44.3	75.5
SRR387661	124M	100	59.9	93.7
SRR534291	114M	100	67.4	93.0
SRR534307	165M	100	65.6	89.5
SRR534319	76.6M	100	29.5	71.2
SRR545695	119M	100	43.4	77.0

3.5 Analysis of Running Time

We show the breakdown of the CPU time of the 3 modules (Section 2) of our tool together with the size of the resulting cdBGs on the 10 biological samples in Table 3. Note that module 3 takes the least CPU time among 3 modules, which proves the efficiency and scalability of the bridging algorithms. Performance was tested on a server with 40 cores (CPU model: Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz) and 760 GB memory.

4 CONCLUSIONS AND DISCUSSION

We study the problem of reconstructing the full-length fragments from paired-end reads without a reference. The experimental results showed that the simulated RNA-seq data does provide sufficient information for accurate bridging. As for the real RNA-seq data, it provides enough information to bridge a large part of the reads.

Table 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 our bridging tool on the 10 biological RNA-seq samples.

SRA ID	size of cdBG		CPU time (in minutes)		
	#vertices	#edges	1	2	3
SRR307903	3.09M	6.17M	96	26	22
SRR307911	5.24M	10.77M	138	48	135
SRR315323	5.22M	10.14M	102	31	112
SRR315334	3.04M	7.27M	124	46	45
SRR387661	10.69M	26.06M	413	162	118
SRR534291	7.83M	21.66M	679	244	95
SRR534307	15.44M	42.92M	938	512	418
SRR534319	10.23M	23.22M	298	81	97
SRR545695	11.04M	25.65M	415	128	133
SRR545723	9.51M	19.6M	226	64	303

We invented two algorithmic innovations for *de novo* bridging, including a novel truncated Dijkstra's algorithm and a new technique that reuses optimal path trees to speed up. Experimental results proved that the resulting tool is efficient on real data. We also proved that the formulation used for reference-based bridging is also accurate for *de novo* bridging when applied on the cdBG; this conclusion was unclear before we conducted this research.

We explored if bridging could improve *de novo* transcript assembly. To this end, we piped the bridged fragments to one leading assembler TransLiG [15], 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.

The sensitivity of our algorithm is low on some biological RNA-seq samples. One reason is that we used a high bottleneck-threshold to keep high precision, which consequently disconnects paired-end reads in low-coverage gene loci. We are therefore 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 the 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 long consecutive 0s in the profile suggest a false bridge. We are experimenting if such a supporting profile could lead to more efficient algorithms in boosting bridging accuracy.

ACKNOWLEDGMENTS

This work is supported by the US National Science Foundation (2019797 and 2145171 to M.S.) and by the US National Institutes of Health (R01HG011065 to M.S.).

REFERENCES

[1] K.F. Au, H. Jiang, L. Lin, Y. Xing, and W.H. Wong. 2010. Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Res.* 38, 14 (2010), 4570–4578.
[2] N.L. Bray, H. Pimentel, P. Melsted, and L. Pachter. 2016. Near-optimal probabilistic RNA-seq quantification. *Nat. Biotechnol.* 34, 5 (2016), 525–527.
[3] Rayan Chikhi, Antoine Limasset, and Paul Medvedev. 2016. Compacting de Bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics* 32, 12 (2016), i201–i208.
[4] Maciej Długosz and Sebastian Deorowicz. 2017. RECKONER: read error corrector based on KMC. *Bioinformatics* 33, 7 (2017), 1086–1089.
[5] A. Dobin, C.A. Davis, F. Schlesinger, J. Drenkow, C. Zaleski, S. Jha, P. Batut, M. Chaisson, and T.R. Gingeras. 2013. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* 29, 1 (2013), 15–21.
[6] T. Griebel, 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 (2012), 10073–10083.
[7] Brian J Haas, Alexander Dobin, Bo Li, Nicolas Stransky, Nathalie Pochet, and Aviv Regev. 2019. Accuracy assessment of fusion transcript detection via read-mapping and de novo fusion transcript assembly-based methods. *Genome Biology* 20, 1 (2019), 213.
[8] Guillaume Holley and Páll Melsted. 2020. Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. *Genome Biology* 21, 1 (2020), 1–20.
[9] W James Kent. 2002. BLAT—the BLAST-like alignment tool. *Genome research* 12, 4 (2002), 656–664.
[10] D. Kim, B. Langmead, and S.L. Salzberg. 2015. HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* 12, 4 (2015), 357–360.
[11] Sam Kovaka, Aleksey V Zimin, Geo M Pertea, Roham Razaghi, Steven L Salzberg, and Mihaela Pertea. 2019. Transcriptome assembly from long-read RNA-seq alignments with StringTie2. *Genome Biology* 20, 1 (2019), 1–13.
[12] Bo Li and Colin N Dewey. 2011. RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC Bioinformatics* 12, 1 (2011), 323.
[13] Yang I Li, David A Knowles, Jack Humphrey, Alvaro N Barbeira, Scott P Dickinson, Hae Kyung Im, and Jonathan K Pritchard. 2018. Annotation-free quantification of RNA splicing using LeafCutter. *Nature Genetics* 50, 1 (2018), 151–158.
[14] J. Liu, T. Yu, T. Jiang, and G. Li. 2016. TransComb: genome-guided transcriptome assembly via combing junctions in splicing graphs. *Genome Biol.* 17, 1 (2016), 213.
[15] Juntao Liu, Ting Yu, Zengchao Mu, and Guojun Li. 2019. TransLiG: a de novo transcriptome assembler that uses line graph iteration. *Genome Biology* 20, 1 (2019), 1–9.
[16] C. Ma, Shao, M., and C. Kingsford. 2018. SQUID: transcriptomic structural variation detection from RNA-seq. *Genome Biol.* 19, 1 (2018), 52.
[17] R. Patro, G. Duggal, M.I. Love, R.A. Irizarry, and C. Kingsford. 2017. Salmon provides fast and bias-aware quantification of transcript expression. *Nat. Methods* 14 (2017), 417–419. https://doi.org/10.1038/nmeth.4197
[18] M. Pertea, G.M. Pertea, C.M. Antonescu, T.-C. Chang, J.T. Mendell, and S.L. Salzberg. 2015. StringTie enables improved reconstruction of a transcriptome from RNA-seq reads. *Nat. Biotechnol.* 33, 3 (2015), 290–295.
[19] ENCODE project. 2019. *Bulk RNA-seq Data Standards and Processing Pipeline.* https://www.encodeproject.org/data-standards/rna-seq/long-rnas/
[20] Shao, M. and C. Kingsford. 2017. Accurate assembly of transcripts through phase-preserving graph decomposition. *Nature Biotechnology* 35, 12 (2017), 1167–1169.
[21] Trung Nghia Vu, Wenjiang Deng, Quang Thinh Trac, Stefano Calza, Woochang Hwang, and Yudi Pawitan. 2018. A fast detection of fusion genes from paired-end RNA-seq data. *BMC Genomics* 19, 1 (2018), 1–13.
[22] Qimin Zhang, Qian Shi, and Mingfu Shao. 2022. Accurate assembly of multi-end RNA-seq data with Scallop2. *Nature Computational Science* 2, 3 (2022), 148–152.
[23] Zijun Zhang, Zhicheng Pan, Yi Ying, Zhijie Xie, Samir Adhikari, John Phillips, Russ P Carstens, Douglas L Black, Yingnian Wu, and Yi Xing. 2019. Deep-learning augmented RNA-seq analysis of transcript splicing. *Nature Methods* 16, 4 (2019), 307–310.