Bermuda: Bidirectional de novo assembly of transcripts with new insights for handling uneven coverage

Qingming Tang
Toyota Technological Institute
at Chicago
6045 S. Kenwood Ave.
Chicago, Illinois 60637
qmtang@ttic.edu

Sheng Wang
Toyota Technological Institute
at Chicago
6045 S. Kenwood Ave.
Chicago, Illinois 60637
shengwang@ttic.edu

Jian Peng
University of Illinois at
Urbana-Champaign
2118 Siebel Center
201 N Goodwin Ave Urbana,
IL, 61801
jianpeng@illinois.edu

Jianzhu Ma
Toyota Technological Institute
at Chicago
6045 S. Kenwood Ave.
Chicago, Illinois 60637
majianzhu@ttic.edu

Jinbo Xu
Toyota Technological Institute
at Chicago
6045 S. Kenwood Ave.
Chicago, Illinois 60637
jinbo.xu@gmail.com

ABSTRACT

Motivation: RNA-seq has made feasible the analysis of a whole set of expressed mRNAs. Mapping-based assembly of RNA-seq reads sometimes is infeasible due to lack of high-quality references. However, de novo assembly is very challenging due to uneven expression levels among transcripts and also the read coverage variation within a single transcript. Existing methods either apply de Bruijn graphs of single-sized k-mers to assemble the full set of transcripts, or conduct multiple runs of assembly, but still apply graphs of single-sized k-mers at each run. However, a single k-mer size is not suitable for all the regions of the transcripts with varied coverage.

Contribution: This paper presents a de novo assembler Bermuda with new insights for handling uneven coverage. Opposed to existing methods that use a single k-mer size for all the transcripts in each run of assembly, Bermuda self-adaptively uses a few k-mer sizes to assemble different regions of a single transcript according to their local coverage. As such, Bermuda can deal with uneven expression levels and coverage not only among transcripts, but also within a single transcript. Extensive tests show that Bermuda outperforms popular de novo assemblers in reconstructing unevenly-expressed transcripts with longer length, better contiguity and lower redundancy. Further, Bermuda is computationally efficient with moderate memory consumption.

Availability: The software is available for review upon request and will be publicly available once the paper is accepted.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*Sequence Analysis*

General Terms

Bioinformatics

Keywords

RNA-Seq, De novo assembly, Uneven Coverage

1. INTRODUCTION

The advent of RNA-seq technique has revolutionized the study of transcriptomics as it promises a comprehensive transcriptional landscape. Compared to microarrays, RNA-seq can be used to annotate novel genes [28, 33], refine the 5' and 3' ends of genes [12], identify the boundary of exons and introns and analyze differential gene and transcript expression [15, 27, 32, 35]. Further, RNA-seq can be applied to construct the whole set of expressed transcripts, which is one of the computational challenges in the field [10, 21, 24]. Several reference-based assemblers (i.e., mapping assemblers) [10, 13, 32, 38] are developed to reconstruct the full set of expressed mRNAs. With high-quality references, even some erroneous reads can be correctly mapped and thus, expressed transcripts can be accurately reconstructed.

However, a high-quality reference genome is not always available. In fact we do not have reference genomes for the majority of the organisms. Further, due to the genomic difference among the sub-populations within one organism [4, 34], the sample under assembly may be quite different from its reference at some regions. That is, sometimes we cannot rely on the reference even if it does exist. Therefore, de novo assemblers, which do not rely on references, are needed for transcript assembly. A few de novo assemblers [3, 8, 11, 18, 25, 29, 30, 31, 36] have been developed, but they still underperform mapping assemblers (e.g., Cufflinks [32]) when a good reference is available.

1.1 Challenges of de novo transcript assembly

Although significant advance has been made to de novo genome assembly [2, 5, 24, 39], de novo transcriptome assembly is quite different from genome assembly and faces the following challenges.

1.1.1 Isoform identification

Isoforms pose challenge to both de novo and mapping based as-sembly [14, 37]. Due to alternative splicing in the genome of eukaryotes, one gene may be spliced into multiple transcripts (i.e., isoforms) sharing exons. Paired-end sequencing, which has been applied in the study of genome sequences [2, 6, 7, 17], can help resolve isoforms by making use of sequential order (or even sequential distance) of two short reads in the same molecule. In Figure 1A, the read $[r_1(\text{gap})r_2]$ clearly conveys the information that exon A is a prefix of exon D in some transcript. However, since the insert size (i.e., genomic distance between the 5 and 3 sequences of one paired-end read) is usually limited, merely utilizing paired-end information may not resolve isoforms. For example, still shown in Figure 1A, ACD and BCE are two isoforms. Suppose that two paired-end reads $[a_1(\text{gap})a_2]$ and $[b_1(\text{gap})b_2]$ are sequenced from ACD and two paired-end reads $[c_1(\text{gap})c_2]$ and $[d_1(\text{gap})d_2]$ are from BCE respectively. Only based upon these four reads we cannot infer that 'A' must connect to 'D' and 'B' to 'E'. Instead, "ACE" and "BCD" may also be feasible transcripts.

1.1.2 Uneven expression levels

RNA-seq data has unevenly distributed coverage on transcripts under assembly due to a large range of expression. This leads to:

- It is challenging to select a good k-mer size for a de Bruijn graph assembler to conduct assembly. Highly-expressed tran-scripts or isoform-shared exons tend to have higher coverage while lowly-expressed transcripts lower coverage. De Bruijn graphs of a single k-mer size may not be suitable for the reconstruction of transcripts with varied expression levels. The reads for the transcripts with low expression tend to have less overlap and thus, these transcripts are less likely to be represented as a connected path in a de Bruijn graph of a large k-mer size. However, a smaller k may reduce the assembly specificity. As shown in Figure 1B, there is a one-to-one correspondence between the sequence **ACGAACGCACGGACG** and the 4-mer de Bruijn graph, but this is not true for the 3-mer de Bruijn graph.

- It is very challenging to correct sequencing errors. Ideally, the full set of expressed transcripts can be represented as many small disconnected de Bruijn graphs [25], each connected graph may correspond to isoforms of one gene. However, due to the existence of erroneous k-mer nodes, these disconnected components may be coupled together, and traversing such a graph with chimeric edges may generate chimeric transcripts. Most de Bruijn graph genome assemblers, such as Velvet [39] and ALLPATHS [5], correct erroneous nodes by simply removing those k-mers with low support since they are very likely to be erroneous. However,

such a method is not suitable for RNA-seq due to uneven expression levels. The support of an erroneous k-mer from a highly-expressed transcript may be higher than that of a correct k-mer from a lowly-expressed transcript. Therefore, simply removing k-mers with weak support will not only remove errors, but also many correct k-mers extracted from lowly-expressed transcripts.

1.1.3 Coverage variation within a single transcript

To better handling the uneven coverage on transcripts, some transcriptome assemblers conduct multiple runs of assembly, and apply a single but different k-mer de Bruijn graph at each run [3, 25, 30]. However, this strategy cannot deal well with coverage variation in a single transcript. Coverage variation not only exists among transcripts, but also appears within even a single transcript. As shown in Figure 1C, the coverage of the exon AT3G12580-E.2 shows big variation. Actually, the sequencing process can be understood as a random process [9, 22], and each base pair can be hit by the random sequencing process with a certain probability in one run. Thus, for those transcripts with not very high expression levels, the variance of coverage would probably be more significant.

1.2 Related work

Several de novo transcript assemblers have been developed, including the Trinity [11], Oases-M [30], Trans-ABySS [29], IDBA-Tran [25] and SOAPdenovo-Trans [36]. All these assemblers are de Bruijn graph assemblers.

Trinity is developed to assemble highly-expressed transcripts with large support. Trinity uses a simple clustering-based error correction method and thus, may remove many correct k-mers sequenced from the transcripts with low expression. Further, Trinity uses only one de Bruijn graph of single and large-sized k-mer and thus, may miss the transcripts with low expression.

SOAPdenovo-Trans is another de Bruijn graph based assembler derived from the SOAPdenovo2 assembler [20], with an effective scaffolding module for transcript assembly. SOAP denovo-Trans utilizes the Trinity error correction strategy, a carefully designed de Bruijn graph simplification method and the Oases graph traversal strategy to generate high quality transcript candidates.

Unlike Trinity and SOAPdenovo-Trans, both Trans-ABySS and Oases-M use multiple de Bruijn graphs of different-sized k-mers to conduct assembly in multiple runs. In each run only one graph is used and the assembly results of all the runs are combined by a consensus method. Since they still use a single graph to construct each transcript fragment, they may not deal well with a transcript with large coverage variation across different regions [9, 22]. Further, after combining the reconstructed transcripts from all the runs, they may produce du-plicated (but different) copies of the same transcript and thus, generate many more transcripts than expected.

Similar to Oases-M, IDBA-Tran also uses multiple de Bruijn graphs with distinct k to handle transcripts with different expression levels. IDBA-Tran employs a probabilistic progressive approach to iteratively remove erroneous k-mers with local thresholds. In each run, IDBA-Tran deals with the output of last round rather than the raw reads, which is different from Oases-M and Trans-ABySS, but IDBA-Tran
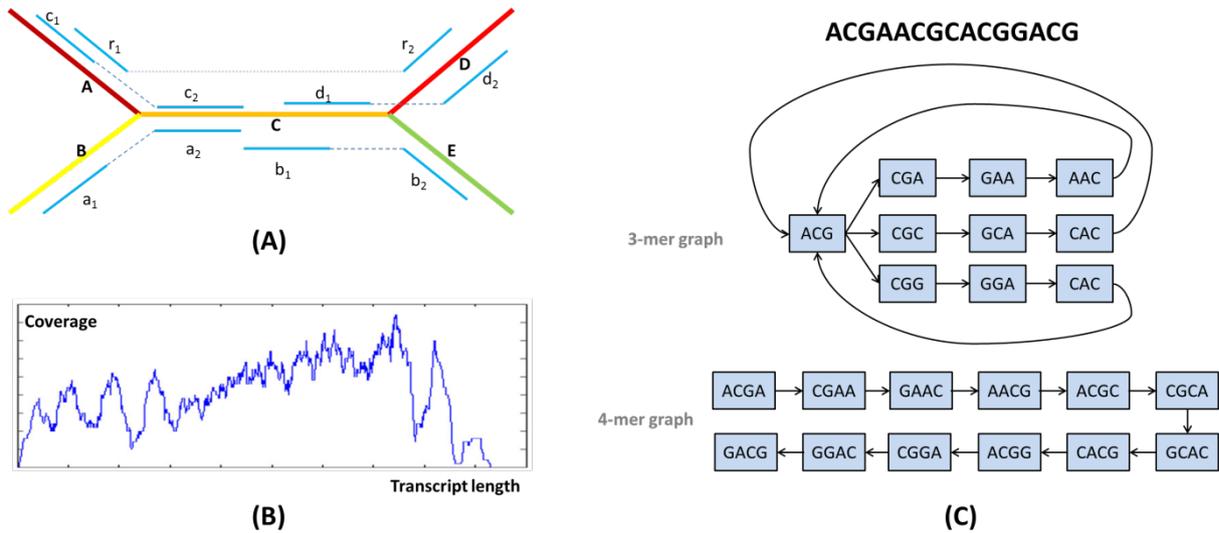
Figure 1: Illustration of challenges of de novo transcript assembly. (A) isoform deconvolution. Five exons (A, B, C, D and E) form a splicing graph and $[r_1(\textbf{gap})r_2]$, $[a_1(\textbf{gap})a_2]$, $[b_1(\textbf{gap})b_2]$, $[c_1(\textbf{gap})c_2]$ and $[d_1(\textbf{gap})d_2]$ are all paired-end short reads. (B) Impact of k-mer size to de Bruijn graph. (C) Read coverage (y-axis) at each position (x-axis) of exon AT3G12580-E.2. Only those reads of SRR391052 that can align to this exon with at least 140 matched bases are considered.

still uses single k-mer de Bruijn graph to handle a group of transcripts.

All the methods described above can be regarded as "transcript-level" methods. That is, in each run of assembly, a single k-mer size is applied globally to all the transcripts.

1.3 Contribution

In this study, we present a new de novo transcript assembler Ber-muda with the following features:

- Bermuda uses dynamic k-mer sizes to assemble different regions of a single transcript fragment in a single run. Consequently, it can deal with not only uneven expression levels among transcripts, but also coverage bias along a single transcript resulting from sequencing at random positions. By contrast, existing methods use single strategy (e.g. a single k-mer graph) to assemble the whole set of transcripts in one run. To the best of our knowledge, Bermuda is the first assembler that employs dynamic strategies for transcript assembly in a single run.

- Bermuda uses de Bruijn graphs for only pre-assembly error correction and then directly joins overlapping short reads into transcripts, as opposed to existing methods that search connected paths in the graphs as the assembly results. By doing so, we can significantly reduce false positives (e.g., chimeric paths) introduced by de Bruijn graphs.

- Bermuda uses a bi-directional assembly process to join overlapping short reads, which can significantly reduce the impact of insert size.
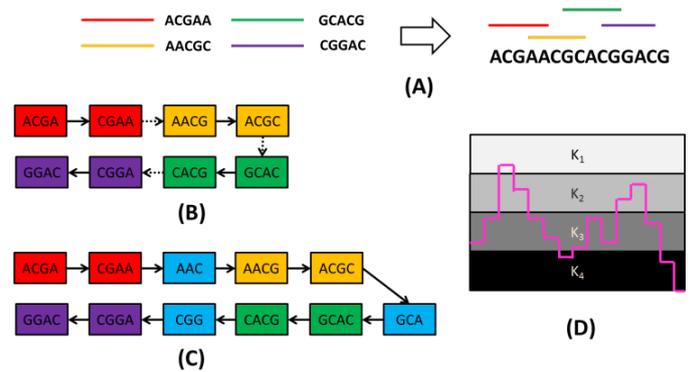
2. MAERIALS AND METHODS

Figure 2: Illustration of applying multiple k-mer sizes to assemble a single transcript. (A) The reads. (B) The corresponding 4-mer de Bruijn graph built from reads. (C) The $3,4$-mer de Bruijn graph. (D) Applying four k ($k_1 > k_2 > k_3 > k_4$) to assemble a transcript with varied coverage.

2.1 New insights

Existing de Bruijn graph assemblers either apply a de Bruijn graph of a carefully-chosen k-mer size to assemble the full set of transcripts, or conduct multiple runs of assembly and use a single-sized k-mer graph in each run. However, in some situations, it is not easy to find such a single k so that high-quality assembly can be con-ducted using that k-mer graph. As shown in Figure 2A, supposing there are four reads for the sequence **ACGAACGCACGGACG**. Although the 3-mer graph built from these reads is connected, it contains a few loops (see Figure 2B), which makes it hard to identify the true sequence among so many possible paths without extra information. If 4-mers are used, as shown in Figure
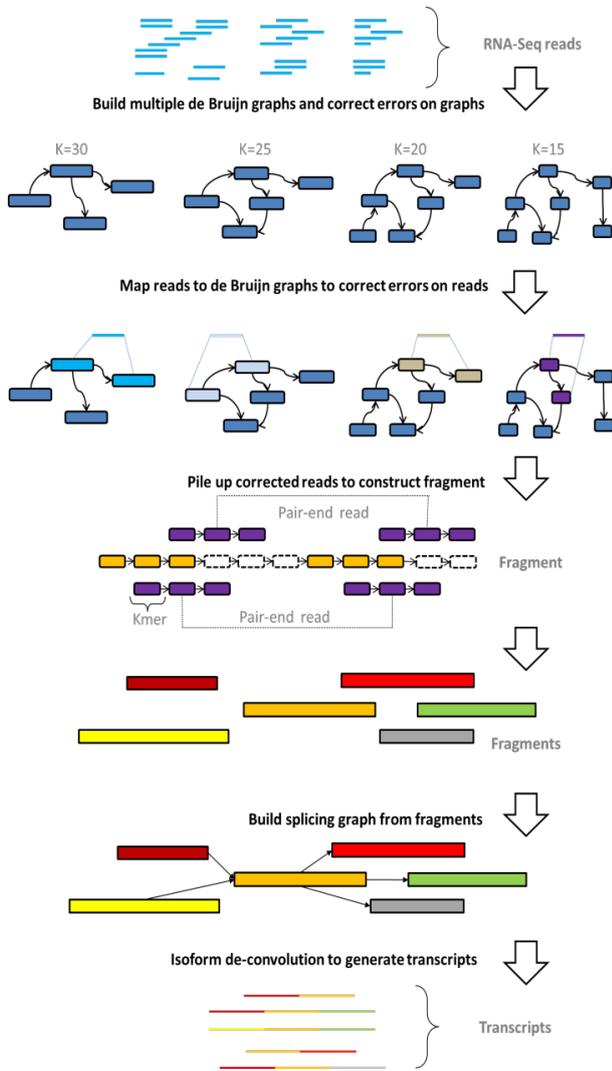
Figure 3: Workflow of Bermuda.

2B, it will result in four disconnected small de Bruijn graphs (marked by four different colors). That is, there is no single k suitable for this case. However, if we use both 3-mers and 4-mers, then we can build a simple, linear and connected graph from the four reads, as shown in Figure 2C.

The intuition is that we shall choose a k-mer size considering the read coverage at each position. In particular, given a sequence $\{a_1, a_2, ..., a_N\}$ and M reads, the optimal k for a_i shall be the maximum number s, such that there exists a path $[a_{i-s}, ..., a_{i-1}] \rightarrow [a_{i-s+1}, ..., a_i] \rightarrow [a_{i-s+2}, ..., a_{i+1}]$ in the s-mer de Bruijn graph. The larger read coverage at one point, it is more likely we can use a larger k-mer size.

Due to read coverage variation, it is infeasible to choose a specific k-mer size optimal for all positions. Instead, we may choose several distinct k-mer sizes, and adaptively choose one based on the local read coverage. In particular, we may divide the whole coverage range into four levels and at each level, we use a specific k, as shown in Figure 2D. Generally speaking, larger k is used for regions with higher coverage and smaller k for regions with low coverage.

2.2 Overview of Bermuda

Figure 3 illustrates the overall flowchart of Bermuda. Bermuda first corrects errors in reads, then it directly joins reads to build fragments. Afterwards, Bermuda links these fragments to form splicing graphs, each of which may correspond to isoforms of one or several genes, and finally traverses the graphs to generate candi-date transcripts. To correct errors in reads, Bermuda first builds de Bruijn graphs, and conduct graph reduction in each graph. Then Bermuda maps reads back to the reduced de Bruijn graphs to correct sequencing errors in the reads. By default, Bermuda builds four types of de Bruijn graphs of $30-$, $25-$, $20-$, and $15-$mers, respectively. After read correction, Bermuda bi-directionally constructs transcript fragments one by one. For each transcript fragment under construction, Bermuda adaptively uses graphs of different k-mer sizes to join overlapping paired-end reads until the whole fragment has been constructed or no compatible reads are available. Bi-direction construction helps with low expression regions of a transcript and also isoform detection. Bermuda repeats the above construction procedure for each possible transcript fragment and also builds (alternative) splicing graphs for isoform detection. A splicing graph encodes all expressed isoforms of one gene or all transcripts of some similar genes. Finally, Bermuda employs the graph traversal algorithm of Oases [30] to identify transcripts.

2.3 De Bruijn graphs construction and reduction

Bermuda uses four types of k-mer de Bruijn graphs where k equals to 15, 20, 25, and 30. A k-mer graph is represented as a set of $(k + 1)$-mers, each representing an edge in the k-mer graph. For example, an edge "$ACG \rightarrow CGT$" in a 3-mer graph can be represented by a 4-mer $ACGT$. The graph edges are stored in hash tables and balanced trees. See Supplementary Method for more technical details.

After building the de Bruijn graphs, Bermuda starts the graph reduction process to remove erroneous nodes in each of the de Bruijn graphs. Typically, a graph reduction process includes removing short tips and parallel paths that share the same start and end nodes with a true path [16, 18, 23, 26, 29, 30].

However, a typical parallel path search algorithm may detect an erroneous path that sharing only its start and end nodes with a true path. It would be time-consuming to detect an erroneous path that shares its internal nodes with other true paths. Such kind of parallel paths may be more easily detected in a de Bruijn graph of larger k-mers as the internal nodes in an erroneous path may no longer match with nodes in other true paths when k increases. Also, it would be impossible to correct errors in an isolated short path. However, in de Bruijn graph of small k-mers, such kind of isolated path may become either a short tip or an erroneous path parallel to some true path if the isolated short path is erroneous. So, by simultaneously examining multi-k-mer graphs, we can better remove erroneous nodes. For each single graph, we use a heuristic method to efficiently define the parallel paths. See Supplementary Method for more technical details about our parallel path searching algorithm for a de Bruijn graph.
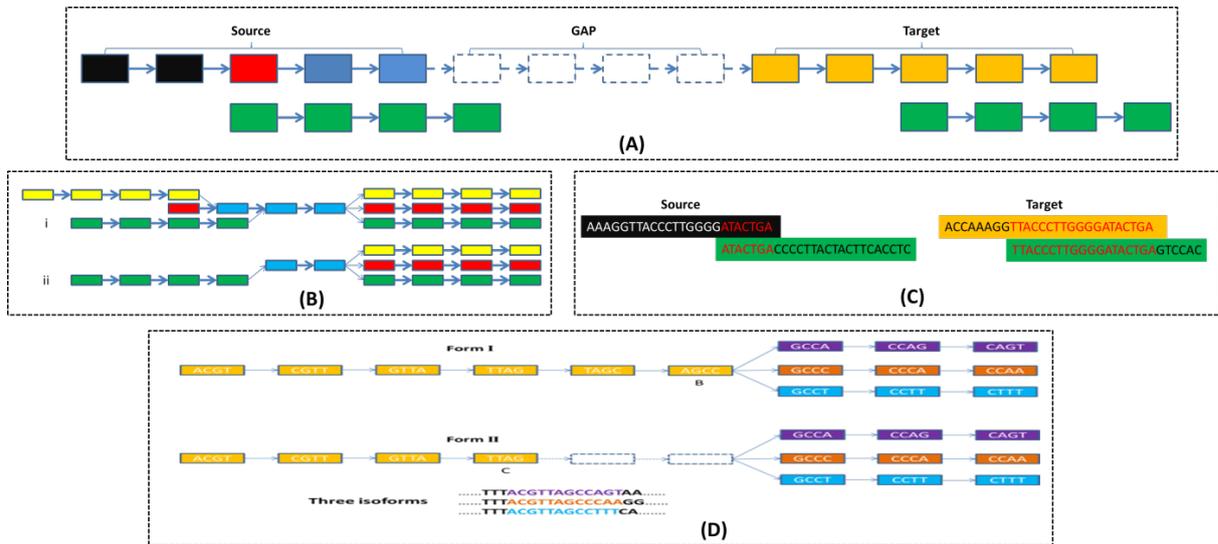
2.4 Read error correction

Figure 4: Illustration of transcript fragment construction and splicing graph building. (A) Illustration of fragment growth process ($5'$ to $3'$). (B) Forks tracking (C) An example of reverse direction construction. (D) Illustration of two forms of forks. In Form I, node B directly connects to three branches of the fork. In Form II, a gap exists between node C and the three branches.

We correct errors in the $5'$ and $3'$ sequences of one raw paired-end read separately, but with the same procedure. To correct errors in a $5'$ or $3'$ sequence, we use the reduced k-mer de Bruijn graphs in the decreasing order of k ($k = 30$, 25, 20, and 15) and execute the following two steps.

- If the first k-mer of the $5'$ sequence appears in the reduced k-mer graph, in the graph find the most similar path starting with this k-mer node for the sequence. Replace the $5'$ sequence by the path if they differ from each other by 1 or 2 positions. Otherwise, go to the next step.

- Randomly select one k-mer (other than the first one) from the $5'$ sequence and use it as seed to find the most similar path for the sequence. Replace the $5'$ sequence by the path if they differ from each other by 1 or 2 bases. Otherwise, use reduced graph with smaller k-mer size and goes to step one.

2.5 Transcript fragment construction

Here we use "transcript fragment" to denote the concatenation of some compatible reads. A fragment may correspond to a single transcript, or multiple isoforms.

2.5.1 Basic fragment construction procedure

We construct transcript fragments in both directions: from $5'$ to $3'$ and from $3'$ to $5'$. We grow a fragment by gradually adding and aligning compatible reads, using paired-end information and the context of both the $5'$ and $3'$ sequences. In the $5'$ to $3'$ direction, we say that one read is compatible with one node in the fragment under construction if the $5'$ sequence of this read starts from this node. Similarly, in the $3'$ to $5'$ direction, a read is compatible with one node if its $3'$ sequence starts from this node.

Since fragment construction in both directions is quite similar, we only describe the construction procedure from $5'$ to $3'$, as shown in Figure 4A. A fragment under construction consists of two parts: Source and Target. The Source and Target are initially formed by a pair of $5'$ and $3'$ sequences, respectively. They are disconnected and may become connected as the construction goes on. Each node in the Source has one of the three colors: black, red and blue, indicating the status of the reads compatible with the node. A black, red and blue node indicates that the compatible reads have been examined for the fragment growth, are currently under examination and are yet to be examined, respectively. At any time there is at most one red node. When current red node is done, we move to the blue node right next to the red and set its color to red if blue nodes exist. As shown in Figure 4A, if the $5'$ and $3'$ sequences of one read compatible with the red node can be exactly aligned to the Source and Target respectively, the fragment can be linearly extended. By "exact alignment" we mean that the prefix of one sequence is exactly same as the suffix of the Source or Target. Otherwise, alternative paths (i.e., branches) are introduced to the Source or/and the Target, which results in a nonlinear structure called fork. The branches in a fork may stand for a true path or erroneous/chimeric paths. We merge a pair of branches if they differ by at most 2 bases. Assume x is the branch with the largest support, and let $|x|$ denote its support. Then any other branch with supports at least $0.05 \times |x|$ is treated as a true path. Otherwise, it is treated as an erroneous path and accordingly removed. A fork with true branches encodes isoforms or paralogous genes.

Note that all the new nodes added to the Source are marked as blue. After examining all the nodes in the Source, we will also examine the nodes in the Target and continue fragment construction.

2.5.2 Using multiple k-mer size

To handle transcripts with uneven expression levels or coverage bias along a single transcript, Bermuda makes use of four types of k-mer sizes, say $30-$, $25-$, $20-$, and $15-$mers, respectively and by default. Bermuda always chooses larger k-mers when there is sufficient coverage. Otherwise, it uses a smaller k-mers to increase sensitivity. More specifically, let K denotes the size of the k-mers currently used. Whenever Bermuda finishes examining one red node (see Figure 4A), supposing that there are x blue nodes right after the red node, Bermuda switches to the k_1-mer de Bruijn graph where k_1 is the largest integer (among 15, 20, 25, and 30) smaller than $K + x - 1$. When $K = 15$, and $x = 0$, our construction algorithm terminates because we believe that overlapping smaller than 15 base pairs are not reliable. In such kind of case, fragment construction in reverse direction may be applied to continue the fragment construction as described in the following section.

2.5.3 Fragment construction in reverse direction

Fragment growth in reverse direction can help detect isoforms since fragment construction along a single direction may not be able to fully capture the complex structure of isoforms. For example, as in Figure 4B(i), we have a de Bruijn graph representation of three transcripts, which share a blue path in the middle. If we only use $5'$ to $3'$ construction starting from the green path, then we can only build a transcript fragment shown in Figure 4B(ii), which misses the red and yellow paths in the $5'$ side (see Figure 4B(i)). Nevertheless, if we conduct the construction along both directions, we may build the three transcripts shown in Figure 4B(i). Transcript construction in reverse direction is also useful when the Source and Target of one fragment under construction is still disconnected after the $5'$ to $3'$ construction is done. As shown in Figure 4C, assume that there is one read (in green) overlapping with both the Source and Target. Due to the large variation of the insert size, the $5'$ sequence of this read overlaps with the Source by only 6 bases (the 6 bases overlap can be identified by brute force search or applying suffix tree), which is not a strong evidence indicating that the $5'$ sequence shall be joined to the Source. By contrast, the $3'$ sequence overlaps with the Target by many more bases. Therefore, if we extend the fragment from the Target to Source, the Source and Target may be connected. That is, reverse direction construction may help connect the Source to the Target or reduce the gap size between them.

2.6 Building splicing graphs

As mentioned before, we use splicing graphs for isoform deconvolution. We build splicing graphs after constructing one transcript fragment in both directions ($5'$ to $3'$ and $3'$ to $5'$). When there is no fork in the constructed transcript fragment, there are no isoforms and hence, it is easy to deal with. If there are branches in either the Source or Target, there may be two types of forks, as shown in Figure 4D. When there is a common junction node connecting the branches as shown in Form I, we split the fragment at this junction node. The boundary of one fragment is defined by junction node. As in Figure 4D, node B is the junction node for Form I and it defines the boundary for the sequence marked by yellow. When there is a gap (the blank nodes in Figure ??D) as shown in Form II of Figure 4D, we do not have a junction node as in Form I. As such, we find the nearest node (e.g., node C in Form II) such that the reads passing

through it would support at least two branches. We treat the nearest node as the junction node. Form II appears due to lack of reads to support the two blank nodes as shown in the graph. Finally, we split the frag-ment at all the junction nodes, treat each resultant linear segment as a vertex of the splicing graph and add one directed edge (from $5'$ to $3'$ls) between two segments that share a junction node and may belong to the same transcript.

3. RESULTS

3.1 Datasets

We benchmark our algorithm using the Human, Mouse and Arabidopsis Thaliana data sets. In total we have tested our method on 12 datasets. However, due to space limit, here we present experimental results on three real data sets: SRR364830 (human), SRR391051 (thaliana) and SR-R404355 (mouse). We compare our method Bermuda with four popular de novo transcript assemblers, including Trinity, Oases-M, IDBA-Tran and SOAPdenovo-Trans. To save space, we denote Oases-M as Oases, IDBA-Tran as IDBA, and SOAPdenovo-Trans as SOAP. All these assemblers use default or recommended parameter settings. We ran Oases using the supplied python pipeline (oases_pipeline.py) with $k_{min} = 23$ and $k_{max} = 31$. Such a pipeline will conduct a postprocessing to reduce redundancy in the results. For Bermuda, we use the 30, 25, 20 and 15-mer for error correction and fragment construction. We pre-process all the raw reads by removing the reads containing little information evaluated as follows: for a continuous segment of 'N's, we assign a score of 20 to the first 'N' and a score of 2 to each of the remaining 'N's. We assign a score of 1 to each base other than 'N'. Let $P(N)$, $P(A)$, $P(C)$, $P(G)$ and $P(T)$ denote the total scores over 'N', 'A', 'C', 'G' and 'T' respectively, and S be $P(A)+P(C)+P(G)+P(T)+P(N)$. We calculate the information content of one sequence as follows:

$$\frac{P(A)}{S} \ln \frac{S}{P(A)} + \frac{P(C)}{S} \ln \frac{S}{P(C)}$$
$$+ \frac{P(G)}{S} \ln \frac{S}{P(G)} + \frac{P(T)}{S} \ln \frac{S}{P(T)} + \frac{P(N)}{S} \ln \frac{S}{P(N)} \quad (1)$$

We remove a paired-end read if either its $5'$ or $3'$ sequence has information content less than 0.5. Some assemblers use the insert size information to assist transcript assembly, but we do not do so because the insert size may vary a lot.

3.2 Experiment design and performance metrics

To evaluate the predicted transcripts, we use BLAST [1] to align them to the corresponding cDNA databases. Human and mouse cDNA databases are downloaded from and Arabidopsis Thaliana cDNA database is from Only the best BLAST match for a predicted transcript is used to evaluate the performance.

We consider both completeness of a real transcript and correctness of a constructed candidate transcript. The completeness of one constructed candidate transcript is defined as the number of matched bases normalized by the length of the most similar cDNA. The completeness of one cDNA is the maximum completeness value of all the related alignments. The correctness of one constructed candidate is defined as the number of matched bases normalized by its

Table 1: Statistics of assembly results by five assemblers.

	#Recovered	#Candidates	#Corrects	Pre1(%)	Aligned length	Unaligned length	Pre2
			SRR364830 (human)				
Bermuda	10171	89032	34233	36.4	101445435	28502762	3.56
Trinity	8330	120974	34484	28.5	104505218	33668196	3.56
Oases	-	-	-	-	-	-	-
IDBA	9194	87483	26898	30.7	93398029	26190201	3.57
SOAP	7017	65562	37902	57.8	59375913	24522156	2.42
			SRR391051 (thaliana)				
Bermuda	5752	38043	18071	47.5	27854983	15119906	1.84
Trinity	4658	48128	20692	43.0	22169248	13177044	1.68
Oases	3642	112009	62435	55.7	33887243	14028743	2.42
IDBA	5439	34121	15893	46.6	23212486	12599930	1.84
SOAP	4123	60918	21528	35.3	20793459	14103059	1.47
			SRR404355 (mouse)				
Bermuda	1943	18876	13207	69.9	20032435	3132349	6.40
Trinity	973	24570	16746	68.2	13514258	1986079	10.09
Oases	1277	80359	57983	71.1	25825409	2443274	10.57
IDBA	1788	17733	10984	61.9	17046935	3253578	5.24
SOAP	1443	35051	16529	47.2	15978392	4930091	3.24

Table 2: Performance (i.e., #Recovered) with respect to expression levels.

	[0-20]	[20-40]	[40-60]	[60-80]	[80-100]
			SRR364830 (human)		
Bermuda	632	2462	4637	4455	1032
Trinity	417	1847	3571	3948	1019
Oases	-	-	-	-	-
IDBA	524	2155	4001	4263	952
SOAP	350	1536	3047	3531	886
			SRR391051 (thaliana)		
Bermuda	15	102	1403	3112	3478
Trinity	2	51	710	2422	3211
Oases	3	60	730	2317	3102
IDBA	3	92	1228	3052	3428
SOAP	1	42	794	2613	2897
			SRR404355 (mouse)		
Bermuda	6	61	261	912	1723
Trinity	1	9	60	362	1191
Oases	1	17	92	491	1528
IDBA	1	33	234	879	1693
SOAP	0	26	215	828	1545

length.

To further evaluate the performance of Bermuda, we analyze the constructed candidate transcripts at a given completeness level. Let A denotes the set of candidates with completeness level larger than a given threshold. For each element $x \in A$, denote the best match of x as $M(x)$. For one real cDNA t, if there are r elements $a_1, a_2, ..., a_r$ in A, such that $M(a_i) = t$ for $1 \le i \le r$, then the redundancy of t with respect to A is $r - 1$. Then the redundancy of A is calculated by summing over the redundancy of all the cDNAs in the database.

To examine the performance of Bermuda at different expression levels, we align each RNA-Seq read set to the corresponding gene database using the Burrows-Wheeler Alignment tool [19] and calculate the coverage of each gene as described in [15]. We divide all the expression levels in-

to 5 quantiles: $[0, 20\%]$, $[20, 40\%]$, $[40, 60\%]$, $[60, 80\%]$, and $[80, 100\%]$, and calculate the number of recovered real transcripts in each quantile.

Table 3: Computational resources.

	Time (seconds)			Memory
	1 core	2 cores	4 cores	GB
		SRR364830 (human)		
Bermuda	35982	21337	11023	20.3
Oases	>86400	>86400	>86400	> 126
IDBA	107520	60323	32234	16.3
SOAP	32880	17141	8823	24
		SRR391051 (thaliana))		
	Time (seconds)			Memory
	1 core	2 cores	4 cores	GB
Bermuda	9157	6231	4075	14.2
Oases	140400	80565	49978	57.5
IDBA	22380	14640	7680	8.5
SOAP	10175	5973	3998	8.2
		SRR404355 (mouse)		
	Time (seconds)			Memory
	1 core	2 cores	4 cores	GB
Bermuda	1395	832	510	0.5
Oases	1875	1043	605	30.9
IDBA	6135	3857	2713	2.1
SOAP	1081	669	491	7.0

3.3 Statistics of assembly results

Table 1 lists the statistic information on the output of the five assemblers. Meanwhile, "#Recovered" is the number of real transcripts that can be covered by some candidates at $\ge 90\%$ completeness level. Since the cDNA databases are fixed, so "#Recovered" can be interpreted as the recall. "#Candidates" is the number of the constructed candidate transcripts and "#Corrects" is the number of candidates with 95% correctness. The contig level precision (Pre1) is defined as "#Corrects" normalized by "#Candidates". "Aligned
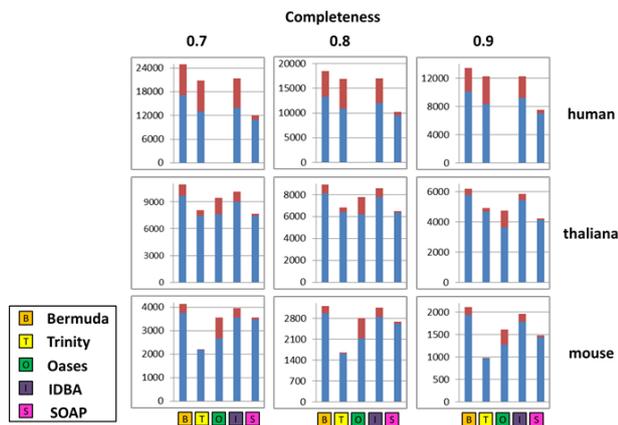
Figure 5: Redundancy at different completeness levels. The nine figures show the comparison of redundancy of the five assemblers over human, thaliana and mouse data sets at completeness levels of 0.7, 0.8 **and** 0.9**, respectively.**

length" is the total number of bases in the predicted transcripts that can be aligned to the cDNA database using BLAST and "Unaligned length" is the number of bases that cannot be aligned. The nucleotide level precision (Pre2) is defined as aligned length normalized by unaligned length. Oases failed to run on the SRR364830 (human) data set because it consumes more than $126G$ RAM, which is unavailable on all the computers to which we can access.

As shown in Table 1, Bermuda outperforms all the other assemblers in terms of the number of reconstructed transcripts (i.e., "#Recovered"). Oases has the best performance in terms of the contig and nucleotide level precision, but very low recall. This may be because Oases generates many short contigs, which can be easily aligned to cDNA databases very well and thus increase both the contig and nucleotide level precision. SOAP also shows very high contig level precision in the human dataset because it generates many very short candidates on this dataset. SOAP in total generates 153618 predicted transcripts, but only 65562 of them are longer than $100bps$. Excluding these exceptions, Bermuda has better precision than the other three assemblers (Trinity, IDBA and SOAP) at both contig and nucleotide levels.

3.4 Completeness distribution of predicted transcripts

Figure 5 shows the redundancy of assembly results at three different completeness levels: 0.7, 0.8 and 0.9. In this figure, red color indicates the number of redundant occurrences, and the blue color indicates the total number of candidates excluding redundancy. As show in Fig 5, Bermuda outperforms the other assemblers in gen-erating non-redundant candidates at each completeness level (marked by blue color). This also indicates that Bermuda outperforms the other assemblers in terms of recovering a real transcript to a better completeness level.

3.5 Relationship with expression level

Table 2 shows the number of reconstructed real transcripts (with completeness level 0.8) by the five assemblers with respect to expression levels. As described in 3.2, we divide all the cDNAs into five groups according to their expression levels: $0-20\%$, $20-40\%$, $40-60\%$, $60-80\%$ and $80-100\%$. As shown in Table 2, Bermuda outperforms all the other assemblers at each expression level on all the three data sets. This shows that Bermuda can deal with unevenly distributed RNA-Seq data pretty well.

3.6 Computational Resources

Table 3 shows the running time and peak memory usage of the four assemblers, tested on three datasets SRR364830 (human), SRR391051 (thaliana) and SRR404355 (mouse). All the assemblers are run with 1, 2 and 4 cores, respectively, on the same Linux compute node with 126G RAM. As shown in Table 3, Bermuda is slightly slower than SOAP, but much faster than the others. Ber-muda consumes moderate amount of memory while Trinity and Oases consume too much memory, which may not be available to many users. The memory consumption by SOAP and IDBA also increase fast along with data size.

4. DISCUSSION

This paper has presented a new method Bermuda for de novo tran-script assembly. This method is designed to handle transcripts with uneven and low expression levels as well as biased read coverage. Experimental results show that our method not only outperforms a few popular de novo assemblers in terms of the number of recov-ered transcripts, but also has favorable running time and moderate memory usage. We will extend our work along two lines. One is to design a bet-ter method for isoform de-convolution by better utilizing the quan-tification information (i.e., the observed number of reads mapping to each node of the splicing graph) and formulating it as a regularized network flow problem. We will study an efficient algorithm that can decompose the flow of one splicing graph into several paths, each corresponding to one transcript, subject to the sparsity constraint. Currently our method is mainly designed for short read assembly, we will develop a new method that can combine both short and long reads(Koren, et al., 2012) to further increase assembly accuracy.

5. ACKNOWLEDGMENTS

6. REFERENCES

[1] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[2] P. N. Ariyaratne and W.-K. Sung. Pe-assembler: de novo assembler using short paired-end reads. *Bioinformatics*, 27(2):167–174, 2011.

[3] I. Birol, S. D. Jackman, C. B. Nielsen, J. Q. Qian, R. Varhol, G. Stazyk, R. D. Morin, Y. Zhao, M. Hirst, J. E. Schein, et al. De novo transcriptome assembly with abyss. *Bioinformatics*, 25(21):2872–2877, 2009.

[4] K. Bryc, C. Velez, T. Karafet, A. Moreno-Estrada, A. Reynolds, A. Auton, M. Hammer, C. D. Bustamante, and H. Ostrer. Genome-wide patterns of population structure and admixture among hispanic/latino populations. *Proceedings of the National Academy of Sciences*, 107(Supplement 2):8954–8961, 2010.

[5] J. Butler, I. MacCallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum, and D. B. Jaffe. Allpaths: de novo assembly of whole-genome shotgun microreads. *Genome research*, 18(5):810–820, 2008.

[6] P. J. Campbell, P. J. Stephens, E. D. Pleasance, S. O'Meara, H. Li, T. Santarius, L. A. Stebbings, C. Leroy, S. Edkins, C. Hardy, et al. Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel paired-end sequencing. *Nature genetics*, 40(6):722–729, 2008.

[7] M. J. Chaisson, D. Brinza, and P. A. Pevzner. De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome research*, 19(2):336–346, 2009.

[8] H.-T. Chu, W. W. Hsiao, J.-C. Chen, T.-J. Yeh, M.-H. Tsai, H. Lin, Y.-W. Liu, S.-A. Lee, C.-C. Chen, T. T. Tsao, et al. Ebardenovo: highly accurate de novo assembly of rna-seq with efficient chimera-detection. *Bioinformatics*, 29(8):1004–1010, 2013.

[9] J. C. Dohm, C. Lottaz, T. Borodina, and H. Himmelbauer. Substantial biases in ultra-short read data sets from high-throughput dna sequencing. *Nucleic acids research*, 36(16):e105–e105, 2008.

[10] M. Garber, M. G. Grabherr, M. Guttman, and C. Trapnell. Computational methods for transcriptome annotation and quantification using rna-seq. *Nature methods*, 8(6):469–477, 2011.

[11] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, et al. Full-length transcriptome assembly from rna-seq data without a reference genome. *Nature biotechnology*, 29(7):644–652, 2011.

[12] B. R. Graveley. The haplo-spliceo-transcriptome: common variations in alternative splicing in the human population. *Trends in Genetics*, 24(1):5–7, 2008.

[13] M. Guttman, M. Garber, J. Z. Levin, J. Donaghey, J. Robinson, X. Adiconis, L. Fan, M. J. Koziol, A. Gnirke, C. Nusbaum, et al. Ab initio reconstruction of cell type-specific transcriptomes in mouse reveals the conserved multi-exonic structure of lincrnas. *Nature biotechnology*, 28(5):503–510, 2010.

[14] S. Heber, M. Alekseyev, S.-H. Sze, H. Tang, and P. A. Pevzner. Splicing graphs and est assembly problem. *Bioinformatics*, 18(suppl 1):S181–S188, 2002.

[15] H. Jiang and W. H. Wong. Statistical inferences for isoform expression in rna-seq. *Bioinformatics*, 25(8):1026–1032, 2009.

[16] D. R. Kelley, M. C. Schatz, S. L. Salzberg, et al. Quake: quality-aware detection and correction of sequencing errors. *Genome Biol*, 11(11):R116, 2010.

[17] J. O. Korbel, A. E. Urban, J. P. Affourtit, B. Godwin, F. Grubert, J. F. Simons, P. M. Kim, D. Palejev, N. J. Carriero, L. Du, et al. Paired-end mapping reveals extensive structural variation in the human genome. *Science*, 318(5849):420–426, 2007.

[18] S. Koren, M. C. Schatz, B. P. Walenz, J. Martin, J. T. Howard, G. Ganapathy, Z. Wang, D. A. Rasko, W. R. McCombie, E. D. Jarvis, et al. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature biotechnology*, 30(7):693–700, 2012.

[19] H. Li and R. Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[20] R. Luo, B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu, et al. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience*, 1(1):18, 2012.

[21] J. A. Martin and Z. Wang. Next-generation transcriptome assembly. *Nature Reviews Genetics*, 12(10):671–682, 2011.

[22] L. M. McIntyre, K. K. Lopiano, A. M. Morse, V. Amin, A. L. Oberg, L. J. Young, and S. V. Nuzhdin. Rna-seq: technical variability and sampling. *BMC genomics*, 12(1):293, 2011.

[23] P. Medvedev, E. Scott, B. Kakaradov, and P. Pevzner. Error correction of high-throughput sequencing datasets with non-uniform coverage. *Bioinformatics*, 27(13):i137–i141, 2011.

[24] N. Nagarajan and M. Pop. Sequence assembly demystified. *Nature Reviews Genetics*, 14(3):157–167, 2013.

[25] Y. Peng, H. C. Leung, S.-M. Yiu, M.-J. Lv, X.-G. Zhu, and F. Y. Chin. Idba-tran: a more robust de novo de bruijn graph assembler for transcriptomes with uneven expression levels. *Bioinformatics*, 29(13):i326–i334, 2013.

[26] P. A. Pevzner, H. Tang, and M. S. Waterman. An eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

[27] F. Rapaport, R. Khanin, Y. Liang, M. Pirun, A. Krek, P. Zumbo, C. E. Mason, N. D. Socci, and D. Betel. Comprehensive evaluation of differential gene expression analysis methods for rna-seq data. *Genome Biol*, 14(9):R95, 2013.

[28] A. Roberts, H. Pimentel, C. Trapnell, and L. Pachter. Identification of novel transcripts in annotated genomes using rna-seq. *Bioinformatics*, 27(17):2325–2329, 2011.

[29] G. Robertson, J. Schein, R. Chiu, R. Corbett, M. Field, S. D. Jackman, K. Mungall, S. Lee, H. M. Okada, J. Q. Qian, et al. De novo assembly and analysis of rna-seq data. *Nature methods*, 7(11):909–912, 2010.

[30] M. H. Schulz, D. R. Zerbino, M. Vingron, and E. Birney. Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–1092, 2012.

[31] Y. Surget-Groba and J. I. Montoya-Burgos. Optimization of de novo transcriptome assembly from next-generation sequencing data. *Genome research*, 20(10):1432–1440, 2010.

[32] C. Trapnell, A. Roberts, L. Goff, G. Pertea, D. Kim, D. R. Kelley, H. Pimentel, S. L. Salzberg, J. L. Rinn,

and L. Pachter. Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature protocols*, 7(3):562–578, 2012.

[33] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter. Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature biotechnology*, 28(5):511–515, 2010.

[34] B. S. Weir and C. C. Cockerham. Estimating f-statistics for the analysis of population structure. *evolution*, pages 1358–1370, 1984.

[35] Z. Xia, J. Wen, C.-C. Chang, and X. Zhou. Nsmap: a method for spliced isoforms identification and quantification from rna-seq. *BMC bioinformatics*, 12(1):162, 2011.

[36] Y. Xie, G. Wu, J. Tang, R. Luo, J. Patterson, S. Liu, W. Huang, G. He, S. Gu, S. Li, et al. Soapdenovo-trans: de novo transcriptome assembly with short rna-seq reads. *Bioinformatics*, 30(12):1660–1666, 2014.

[37] Y. Xing, A. Resch, and C. Lee. The multiassembly problem: reconstructing multiple transcript isoforms from est fragment mixtures. *Genome research*, 14(3):426–441, 2004.

[38] M. Yassour, T. Kaplan, H. B. Fraser, J. Z. Levin, J. Pfiffner, X. Adiconis, G. Schroth, S. Luo, I. Khrebtukova, A. Gnirke, et al. Ab initio construction of a eukaryotic transcriptome by massively parallel mrna sequencing. *Proceedings of the National Academy of Sciences*, 106(9):3264–3269, 2009.

[39] D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.