SPATA: A Seeding and Patching Algorithm for *de novo* Transcriptome Assembly

Zhiyu Zhao[1,3], Tin Nguyen Chi[2], Nan Deng[2], Kristen Marie Johnson[3] and Dongxiao Zhu[2,*]
[1]The LONI Institute, Baton Rouge, LA, USA; zzhao5@uno.edu
[2]Department of Computer Science, Wayne State University, Detroit, MI 48202, USA; dzhu@wayne.edu
[3]Department of Computer Science, University of New Orleans, New Orleans, LA 70148, USA
* To whom correspondence should be addressed.

***Abstract*—RNA-seq reads are sampled from the underlying human transcriptome sequence, consisting of hundreds of thousands of mRNA transcripts. *De novo* transcriptome reconstruction from RNA-seq reads is a promising approach but facing with algorithmic and computational challenges derived from nonlinear transcript structures and ultra high-throughput read counts. To tackle this issue, we designed a divide-and-conquer strategy to read localization, followed by a novel algorithm to *de novo* read assembly. Using simulation studies, we have demonstrated a high accuracy in transcriptome structures reconstruction.**

***Index Terms*—RNA-seq; transcriptome; *de novo* assembly; read mapping;**

I. Introduction

Human transcriptomes are highly diverse, overlapping, complex, and dynamic. Alternative splicing and structural variations play important roles in enhancing the range of transcriptome complexity ([1]–[5]). For example, it is reported that over 90% of human genes are alternatively spliced, and up to 5% of structural variations, such as insertions, deletions and translocations, are present within exons. Moreover, all these events are tissue-specific that lead to diversity of human transcriptomes. Tissue-specific gene modeling aimed at deciphering the structures of all expressed transcripts is central to transcriptomics research. The ever-declining cost and increasing depth of RNA-seq provide new opportunities for tissue-specific gene modeling while creating high demand for informatics tools to automate the analysis. Particularly, mapping and assembling of RNA-seq short reads is quite challenging due to the two-level complexity introduced by mRNA splicing and structural variation.

The first level complexity is introduced by splicing and alternative splicing of the annotated exons, including Untranslated Regions (UTR's) ([1], [5]). It creates enormous challenges for mapping reads from exon junctions, termed junction reads, to the reference genome. In response to this challenge, much research has been done to map the junction reads, such as Mapsplice ([6]), SpliceMap ([7]), TopHat ([8]) and HMMSplicer ([9]). These approaches have been proven useful in that they have discovered novel splicing events between known exons. However, it is still challenging to infer the entire set of transcripts due to the nonlinear relationship between the number of exons and the number of possible isoforms. Assumptions such as single alternative splicing event for each transcript must be made. Thus the discoverable splicing mechanisms are mostly limited to a single exon skipping or mutually exclusive exon event (e.g. [10], [11]). Additional limitations for these approaches include their exon-level resolution. Hence, these tools fail to infer transcripts with structural variations, such as truncated or fused exons.

The second level complexity is a more general problem that is introduced by ubiquitous structural variations, such as deletions, insertions, and translocations, in human genomes and transcriptomes ([2]–[4]). It created enormous challenges for both mapping and assembling reads from altered genomic regions. Human genome structural variations have been linked to diverse human diseases and other physiological processes such as aging. Structural variations in the transcriptome intimately correlate with those in the genome, thus can be discovered with related computational approaches. Larger-sized structural variations in the genome can be discovered probabilistically using paired-end reads ([12]–[14]). However, detection of medium to small sized structural variations in both genome and transcriptome remains a challenging problem ([3]).

An intuitive way to tissue-specific gene modeling is to *de novo* assemble transcriptome. Due to the highly diverse and nonlinear transcriptome structure, straightforward application of genome assembling approaches is often unsatisfactory. Recent global transcriptome assembly based approach, Trans-Abyss ([15], [16]), filter and merge the contigs assembled using the Abyss genome assembler ([17]), to form a much smaller set of non-redundant contigs. It is followed by mapping the processed contigs to known transcripts. Other local transcriptome assembly based approach, Cufflinks ([18]), first maps the reads, and then *de novo* assembles a minimum set of compatible transcripts from the cDNA fragments identified by paired-end reads. Both approaches attempt to *de novo* reconstruct mRNA transcripts at the base-pair resolution thus improving over the exon-based approaches. However, the former is limited by parsimonious assumptions and heavy computational burden; the latter does not map and assemble reads with structural variations. Moreover, their assembly accuracies have not been systematically evaluated by simulation studies.

Here we design a novel divide-and-conquer strategy and

develop a new *de novo* assembly algorithm to tackle both levels of complexities in transcriptome reconstruction. We carry out simulation studies to show that our algorithm is able to accurately reconstruct *in vivo* transcript sequences at the base-pair resolution with a minimum of assumption. In particular, our algorithm reconstructs transcript sequences directly from short reads and exhaustively finds the sequences of all expressed mRNA transcripts. The only required input from users is the raw read sequence file in standard fastq or fasta format; and it will export sequences of each expressed mRNA transcript and report tissue-specific splicing and structural variations events.

II. Short Reads Localization

De novo transcriptome assembly represents a computationally daunting task, brought by tens of millions of short reads. Moreover, the non-linear structures of the splicing transcripts compromise the performance of the genome assemblers, which were designed to assemble longer linear genome sequences. The existing global and local transcriptome assemblers, such as Trans-Abyss ([15], [16]) and Cufflinks ([18]), are among the first efforts to reconstruct transcriptome. However, their performance has not been systematically evaluated using simulation studies. An innovation of our approach is first to localize reads, with or without structural variations, to each annotated gene or un-annotated transcriptional unit. After this localization, we *de novo* reconstruct transcript structures within each annotated gene or un-annotated transcriptional unit. We localize reads as follows:

Step 1: Initial alignment of all the reads. We first aligned the entire set of reads, single-end or paired-end, uniquely to the reference genome using Bowtie ([19]). For single-end reads, we initially aligned most of the human exonic reads, e.g. $60\% - 70\%$, to the reference genome (left panel of the Figure 1). For paired-end reads, we localize the reads if at least one read of the pair is aligned to the reference genome, including reads with structural variations (right panel of the Figure 1).

Step 2: Follow-up localization of split reads. For the remaining reads from the step 1, if they are single-end, we equally split each read into three partitions, and localize the split reads to annotated genes or un-annotated transcriptional units using anchors (left panel of the Figure 1). Anchors refer to any partition read(s) that are aligned to the reference genome. For paired-end reads, we equally split each read into two partitions, four partitions for a read pair, and localized those using anchors (right panel of Figure 1).

III. *De Novo* Transcriptome Assembly Algorithm

We propose an efficient algorithm to *de novo* reconstruct transcript sequences at the base-pair resolution within each annotated gene or un-annotated transcriptional unit. The central idea of our algorithm is to use overlaps between a pair of reads. The algorithm proceeds into two consecutive stages. In the Seeding and Growing Stage, we first grow a full-length backbone transcript sequence starting from an

Fig. 1. Short Reads Localization. Reads in the gray-shaded regions correspond to un-localized reads and reads in the clear regions correspond to localized reads. Left panel: single-end reads; Right panel: paired-end reads

arbitrarily selected "seed" short read (see Figure 2), and then label all the reads covered by the backbone as discovered (e.g. $S1$ in Figure 2). For those un-discovered reads, the algorithm grows a sequence from another "seed" as long as possible, and then label all the reads covered by that sequence as discovered ($S2$ or $S3$ in Figure 2). The process continues until there is no read remaining undiscovered. In the Patching and Cutting Stage, when applicable, all the sequences (e.g. $S2$ and $S3$ in Figure 2) passing a minimum overlap cut-off are patched to another sequence ($S1$ in Figure 2) to form the whole set of transcripts (S4, S5 and S6 in Figure 1c). After the sequences have been patched to each other, they are cut into small segments, redundant segments are removed and remaining segments are organized into an isoform graph to reconstruct isoform structures and their corresponding contigs.

A. The Seeding and Growing Stage

1) Definitions:

Definition 1 (Short Read)**.** *A short read R is a sequence of l letters, $(r_1 r_2 ... r_l)$, where $r_i \in \{A, T, C, G\}$ for $1 \leq i \leq l$. For the convenience of algorithm description, denote any sequence of letters taken from $\{A, T, C, G\}$ in the same manner. Also denote an empty sequence as $()$.*

Definition 2 (Overlap)**.** *Short read $P = (p_1 p_2 ... p_l)$ overlaps with $Q = (q_1 q_2 ... q_l)$ if $(p_{l-l_o+1} p_{l-l_o+2} ... p_l) = (q_1 q_2 ... q_{l_o})$ for $l_o \geq k$, where l_o is the length of overlapped letters and k is the minimum overlap cut-off. Denote the overlap length as $overlap(P, Q) = l_o$. Also denote $overlap(Q, P) = -l_o$. If short reads P and Q do not overlap with each other, let $overlap(P, Q) = overlap(Q, P) = 0$. Without loss of generality, we define the overlap between any two sequences in the same manner.*

Definition 3 (Extension)**.** *Given short read P and a set of short reads $T = \{T_1, T_2, ..., T_t\}$, short read $T_r \in T$ is called a right extension of P if $0 < overlap(P, T_r) \leq overlap(P, T_i)$ for every $overlap(P, T_i) > 0$ and $1 \leq i \leq t$. Similarly, $T_l \in T$ is called a left extension of P if $0 < overlap(T_l, P) \leq overlap(T_i, P)$ for every $overlap(T_i, P) > 0$ and $1 \leq i \leq t$. Denote the right extension as $T_r = ext_r(P, T)$ and the left extension as $T_l = ext_l(P, T)$. If P is not extendable toward left or right, denote it as $ext_l(P, T) = ()$ or $ext_r(P, T) = ()$.*

Definition 4 (Merger)**.** *Two short reads can be merged to make a longer sequence if they overlap with each other. Given short*

reads $P = (p_1p_2...p_l)$, $Q = (q_1q_2...q_l)$ and $overlap(P, Q) = l_o$, the merged sequence is $M = (p_1p_2...p_{l-l_o}q_1q_2...q_l)$. Denote it as $M = merge(P, Q)$. Without loss of generality, we define the merger between any two sequences in the same manner.

Definition 5 (Cover and Subsequence)**.** *Given two sequences $S = (s_1s_2...s_l)$ and $S' = (s'_1s'_2...s'_{l'})$, S' is covered by S if $l \geq l'$ and $(s_{i+1}s_{i+2}...s_{i+l'}) = (s'_1s'_2...s'_{l'})$ for some $1 \leq i \leq l$. Denote it as $cover(S', S) = True$. We also say that S' is a subsequence of S and denote it as $S' = sub(S, i, l')$.*

2) A seeding and growing algorithm: Given a set of short reads, the following algorithm uses the first read in the set as a seed and extends it toward left and right, respectively, until no extension reads can be found. A merged sequence is generated and all the short reads covered by the merged sequence are removed from the input set while the seed grows. If there are uncovered short reads after the first round of growth, another seed (the first read in the set) is used to grow a second sequence. This process is repeated until all the reads are covered by some sequence and thus the input set becomes empty. See Figure 2 for an illustration of this algorithm.

Algorithm Seeding and Growing

Input: A set of short reads $T = \{T_1, T_2, ..., T_t\}$ and the minimum overlap length k

Output: A set of sequences S

```
S ← ∅ /* Initialize S */
while T ≠ ∅ do
    P ← T_1 /* Set P as the first short read in T */
    S_P ← P /* Initialize S_P as the sequence of P */
    repeat
        T_l ← ext_l(P, T)
        S_P ← merge(T_l, S_P)
        for each T_i ∈ T and T_i ≠ P do
            if cover(T_i, merge(T_l, P)) = True then
                T ← T − {T_i} /* Remove T_i from T */
            end if
        end for
        P ← T_l /* Set P as its left extension */
    until ext_l(P, T) = ()
    P ← T_1 /* Set P as the first short read in T */
    repeat
        T_r ← ext_r(P, T)
        S_P ← merge(S_P, T_r)
        for each T_i ∈ T and T_i ≠ P do
            if cover(T_i, merge(P, T_r)) = True then
                T ← T − {T_i} /* Remove T_i from T */
            end if
        end for
        P ← T_r /* Set P as its right extension */
    until ext_r(P, T) = ()
    S ← S ∪ {S_P} /* Add sequence S_P to set S */
end while
```

B. The Patching and Cutting Stage

1) Definitions:

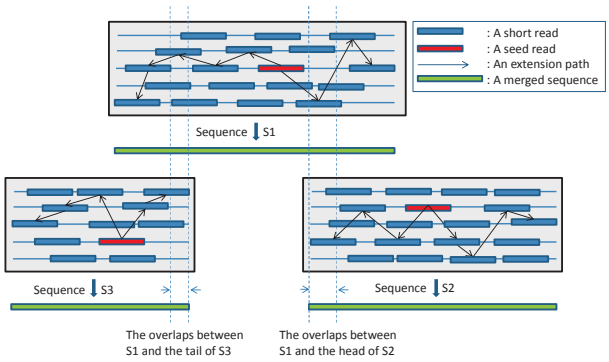


Fig. 2. The Seeding and Growing Stage

Definition 6 (Patch)**.** *Given two sequences $S = (s_1s_2...s_l)$ and $S' = (s'_1s'_2...s'_{l'})$, S' is called a left patch of S if we can find an i and a $p \geq k$ which make $sub(S', 1, p) = sub(S, i, p)$. Denote it as $patch_l(S', S) = (i, p)$. Similarly, S' is called a right patch of S if we can find an i and a $p \geq k$ which make $sub(S', l' - p + 1, p) = sub(S, i, p)$. Denote it as $patch_r(S', S) = (i, p)$.*

See Figure 3 for an illustration of the patching operation.

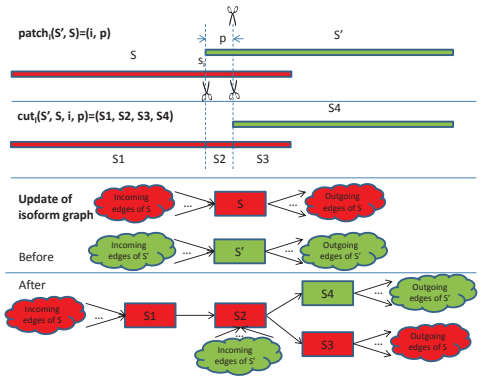
Definition 7 (Cut)**.** *Given two sequences $S = (s_1s_2...s_l)$ and $S' = (s'_1s'_2...s'_{l'})$, if $patch_l(S', S) = (i, p)$, a left cut between S and S' splits them to four subsequences $(s_1s_2...s_{i-1})$, $(s_is_{i+1}...s_{i+p-1})$, $(s_{i+p}s_{i+p+1}...s_l)$ and $(s'_{p+1}s'_{p+2}...s'_{l'})$. Denote this as $cut_l(S', S, i, p) = (S_1, S_2, S_3, S_4)$. Similarly, if $patch_r(S', S) = (i, p)$, a right cut between S and S' splits them to four subsequences $(s_1s_2...s_{i-1})$, $(s_is_{i+1}...s_{i+p-1})$, $(s_{i+p}s_{i+p+1}...s_l)$ and $(s'_1s'_2...s'_{l'-p})$. Denote this as $cut_r(S', S, i, p) = (S_1, S_2, S_3S_4)$.*

See Figure 3 for an illustration of the cutting operation.

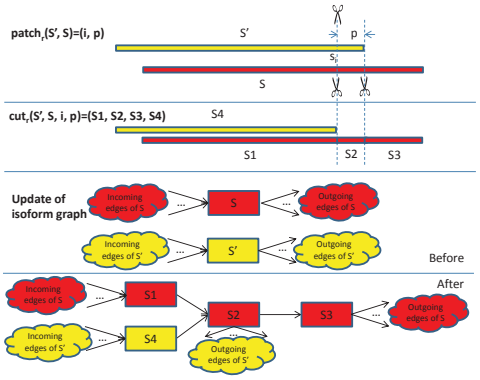
Definition 8 (Joint)**.** *Given two sequences $S = (s_1s_2...s_l)$ and $S' = (s'_1s'_2...s'_{l'})$. A joint sequence between them is $(s_1s_2...s_ls'_1s'_2...s'_{l'})$. Denote it as $joint(S, S')$. Also define the joint between n sequences, $S_1, S_2, ..., S_n$, as $joint(S_1, S_2, ..., S_n) = joint(S_1, joint(S_2, joint(S_3, ..., joint(S_{n-1}, S_n))))$.*

Definition 9 (Isoform graph, isoform structure and contig)**.** *An isoform graph $G = (V, E)$ is a directed graph where each vertex is a sequence of letters from $\{A, T, C, G\}$ and each edge from vertex V_i to vertex V_j indicates that there exists a short read R which is a subsequence of the joint sequence of V_i and V_j i.e. $cover(R, joint(V_i, V_j)) = True$. Given an isoform graph $G = (V, E)$, an isoform structure is a linear path in G starting at a vertex with no incoming edges and ending at a vertex with no outgoing edges. Representing an isoform structure with n vertices as $V_1 \rightarrow V_2 \rightarrow ... \rightarrow V_n$, the corresponding contig is $joint(V_1, V_2, ..., V_n)$.*

2) A patching and cutting algorithm: Given a set of sequences outputted by the seeding and growing algorithm, the



(a) Leftward patching, cutting and graph updating



(b) Rightward patching, cutting and graph updating

Fig. 3. The Patching and Cutting Stage

following algorithm dynamically constructs an isoform graph. Initially, vertices in the graph are set as sequences from the input and the connection relationship (connected or not connected) between them is unknown. For every pair of vertices, the algorithm checks their connection relationship flag. If it is "unknown", the two vertices are tested for patching. If they can patch to each other, then their sequences are cut into segments, the vertices are split, directed edges are added, and the relationship flags between all involved vertices are updated accordingly. Sometimes this patching and cutting process may cause sequences that are unnecessarily cut i.e. in the graph there may be pairs of vertices where each pair is connected only by one edge. If this happens, the two vertices are merged into one. After the isoform graph has been constructed, every linear path starting from a vertex with no incoming edges and ending at a vertex with no outgoing edges is put into an isoform structure set, and the joint sequence based on that path is put into a contigs set.

Algorithm Patching and Cutting

Input: A set of sequences S and the minimum overlap length k

Output: An isoform graph $G = (V, E)$, a set of isoform structures I, and a set of contigs C

$V \leftarrow S$ /* Initialize V, a set of vertices in the graph */
$E \leftarrow \emptyset$ /* Initialize E, a set of edges in the graph */
for each pair $V_i, V_j \in V$ **do**
  $F(V_i, V_j), F(V_j, V_i) \leftarrow$ "$unknown$" /* Initialize F, the relationship flags between vertices */
end for
for each pair $V_i, V_j \in V$ **and** $F(V_i, V_j) =$ "$unknown$" **do**
  if $patch_l(V_j, V_i) = (v, p)$ **then**
    $Update\_Graph(G, V_i, V_j, v, p,$ "$left$"$)$
  else if $patch_r(V_j, V_i) = (v, p)$ **then**
    $Update\_Graph(G, V_i, V_j, v, p,$ "$right$"$)$
  end if
end for
$G \leftarrow Merge\_Vertices(G)$
$I \leftarrow Get\_Isoform\_Structures(G)$
$C \leftarrow \emptyset$ /* Initialize C, a set of contigs */
for each isoform structure $I_i = V_1 \rightarrow V_2 \rightarrow ... \rightarrow V_n \in I$ **do**
  $C = C \cup \{joint(V_1, V_2, ..., V_n)\}$
end for

Procedure Update_Graph$(G, F, V_i, V_j, v, p, dir)$

Input: Isoform graph $G = (V, E)$, flags variable F, vertices V_i and V_j, cutting location variables v and p, and direction variable dir

Output: Updated graph G

if $dir =$ "$left$" **then**
  $(V_1, V_2, V_3, V_4) \leftarrow cut_l(V_j, V_i, v, p)$
else
  $(V_1, V_2, V_3, V_4) \leftarrow cut_r(V_j, V_i, v, p)$
end if
$V \leftarrow (V - \{V_i, V_j\}) \cup \{V_1, V_2, V_3, V_4\}$ /* Remove two and add four vertices */
for each edge $V_m \rightarrow V_n \in E$ **do**
  if $V_n = V_i$ **then**
    replace V_n with V_1
  end if
  if $V_m = V_i$ **then**
    replace V_m with V_3
  end if
  if $V_n = V_j$ **then**
    if $dir =$ "$left$" **then**
      replace V_n with V_2
    else
      replace V_n with V_4
    end if
  end if
  if $V_m = V_j$ **then**
    if $dir =$ "$left$" **then**
      replace V_m with V_4
    else
      replace V_m with V_2
    end if

end if
end for
if $dir =$ "$left$" **then**
$E \leftarrow E \cup \{V_1 \rightarrow V_2, V_2 \rightarrow V_3, V_2 \rightarrow V_4\}$ /* Add three edges */
else
$E \leftarrow E \cup \{V_1 \rightarrow V_2, V_2 \rightarrow V_3, V_4 \rightarrow V_2\}$ /* Add three edges */
end if
$Update\_Flags(F, V_i, V_j, V_1, V_2, V_3, V_4)$

See Figure 3 for an illustration of the graph updating procedure.

Procedure Update_Flags($F, V_i, V_j, V_1, V_2, V_3, V_4$)
Input: Flags variable F and vertices $V_i, V_j, V_1, V_2, V_3, V_4$
Output: Updated flags variable F
delete flags $F(V_i, V_i)$ and $F(V_j, V_j)$
for each relationship flag $F(V_m, V_n)$ **do**
if $V_m = V_i$ **then**
$F(V_1, V_n), F(V_2, V_n), F(V_3, V_n) \leftarrow F(V_m, V_n)$
$F(V_n, V_1), F(V_n, V_2), F(V_n, V_3) \leftarrow F(V_m, V_n)$
delete flags $F(V_m, V_n)$ and $F(V_n, V_m)$
end if
if $V_m = V_j$ **then**
$F(V_2, V_n), F(V_4, V_n) \leftarrow F(V_m, V_n)$
$F(V_n, V_2), F(V_n, V_4) \leftarrow F(V_m, V_n)$
delete flags $F(V_m, V_n)$ and $F(V_n, V_m)$
end if
end for
for $1 \leq m, n \leq 4$ **do**
$F(V_m, V_n) \leftarrow$ "$known$"
end for
$F(V_3, V_4), F(V_4, V_3) \leftarrow$ "$unknown$"

Procedure Merge_Vertices(G)
Input: An isoform graph G
Output: Graph G with merged vertices
for each pair $V_i, V_j \in V$ **do**
if V_i has only one outgoing edge **and** V_j has only one incoming edge **and** $V_i \rightarrow V_j \in E$ **then**
$V_m \leftarrow joint(V_i, V_j)$ /* Merge two vertices */
$V \leftarrow (V - \{V_i, V_j\}) \cup \{V_m\}$ /* Remove two vertices and add the merged vertex */
for each $V_n \rightarrow V_i \in E$ **do**
Replace V_i with V_m /* Update incoming edges of V_i */
end for
for each $V_j \rightarrow V_n \in E$ **do**
Replace V_j with V_m /* Update outgoing edges of V_j */
end for
end if
end for

Procedure Get_Isoform_Structures(G)
Input: An isoform graph G
Output: An isoform structure set I
$I \leftarrow \emptyset$ /* Initialize I */
for each vertex $V_i \in V$ that has no incoming edges **do**
$I \leftarrow \{V_i\}$
end for
for each path $I_i = V_1 \rightarrow V_2 \rightarrow ... \rightarrow V_n \in I$ **do**
if vertex V_n has outgoing edges **then**
for each edge $V_n \rightarrow V_m \in E$ **do**
$I \leftarrow I \cup \{V_1 \rightarrow V_2 \rightarrow ... \rightarrow V_n \rightarrow V_m\}$
end for
$I \leftarrow I - \{I_i\}$ /* Remove I_i from I */
end if
end for

C. Error Control

The above algorithm does not take into account the sequencing errors in short reads. In reality, due to those errors, the short reads do not always perfectly overlap with each other. Therefore the algorithm must have error tolerance capability in order to process real sequencing data. For this purpose, we slightly modify the following definitions:

Definition 10 (Overlap, updated)**.** *Short read $P = (p_1 p_2 ... p_l)$ overlaps with $Q = (q_1 q_2 ... q_l)$ if $(p_{l-l_o+1} p_{l-l_o+2} ... p_l) = (q_1 q_2 ... q_{l_o})$ for $l_o \geq k$ with at most $e_1 l_o$ differences i.e. errors and no more than e_2 errors are next to each other, where l_o is the length of overlapped letters, k is the minimum overlap cut-off, e_1 is the maximum error rate in any short read i.e. there are at most $e_1 l$ sequencing errors in a short read of length l, and e_2 is a small constant, for instance, 2. Without loss of generality, we redefine the overlap between any two sequences in the same manner.*

Definition 11 (Cover and Subsequence, updated)**.** *Given two sequences $S = (s_1 s_2 ... s_l)$ and $S' = (s'_1 s'_2 ... s'_{l'})$, S' is covered by S if $l \geq l'$ and $(s_{i+1} s_{i+2} ... s_{i+l'}) = (s'_1 s'_2 ... s'_{l'})$ for some $1 \leq i \leq l$ with at most $e_1 l'$ differences i. e. errors and no more than e_2 errors are next to each other. We also say that S' is a subsequence of S.*

With these modified definitions, the proposed algorithms can now be applied with no change to process data with sequencing errors. The algorithm needs to take two parameters, e_1 and e_2, where e_1 can be determined by the user and is usually less than 2%. e_2 can be a fixed constant such as 2 or 3.

IV. Complexity Analysis

A. Time Complexity of the Localization Stage

As the localization stage aligns whole or split short reads to the reference genome, given an aligner of running time $T(m, l)$ where m is the average number of short reads localized into one annotated gene or un-annotated transcriptional unit and l is the short read length, the localization stage runs in $O(T(m, l))$ time for each gene or transcriptional unit. Suppose the total number of genes and un-annotated transcriptional units in a whole transcriptome is n, the time complexity of the localization stage is $O(T(m, l)n)$.

Suppose the average length of a grown sequence is s, the *de novo* reconstruction algorithm of transcript structures runs in $O(smn)$ time as shown below.

B. Time Complexity of the Seeding and Growing Algorithm

There are three major operations in this algorithm: $ext()$, $merge()$, and $cover()$. Their complexities all depend on the complexity of a basic operation: $overlap()$. Suppose the short read length is l, its complexity is $O(l)$. In each extension, because there are at most m pairs of short reads to test for overlaps , operation $ext()$ runs in $O(ml)$ time. The complexity of $merge()$ is $O(l)$ because the merging locations are already known from the preceding $ext()$ operation. The complexity of $cover()$ is also $O(l)$. Because the typical length of a grown sequence is s and usually an $ext()$ operation extends a growing sequence by $O(l)$ letters, when seeding and growing a sequence, the number of times $ext()$ and $merge()$ operations are called is $O(\frac{s}{l})$. The number of $cover()$ operations, when its input is limited to those overlapped short reads discovered by the $ext()$ operation, is only $O(\frac{s}{l}o)$ where o is the number of overlapped short reads discovered when extending a read. Therefore, the total time for growing one sequence is $O(\frac{s}{l}ml) + O(\frac{s}{l}l) + O(\frac{s}{l}ol)$=$O(sm)$ as $m > o$. Also, there are only a small number of such sequences to grow. Typically this number is $O(1)$. Thus the total time for growing all the sequences in one gene or transcript unit is $O(sm)$. The total time for growing all the sequences in n genes or transcript units in a whole transcriptome is $O(smn)$. Typical values of s, m and n are $O(10^3)$, $O(10^6) \sim O(10^7)$ and $O(10^4)$, respectively.

C. Time Complexity of the Patching and Cutting Algorithm

The time complexity of the patching and cutting algorithm is determined by the number of vertices and the number of edges in the isoform graph. Because there are only $O(1)$ sequences generated in the Seeding and Growing Stage, typically the number of vertices in the final graph is $O(1) \sim O(10)$ and the number of edges is $O(1) \sim O(10^2)$ at most. As these numbers are much smaller than s, m, and n, the time complexity of the patching and cutting algorithm is neglectable.

The total time complexity of the algorithms for short reads localization and transcripts reconstruction is thus $O(T(m,l)n + smn)$.

V. Simulation Studies

We simulated RNA-seq experiments using FluxSimulator ([http://flux.sammeth.net/simulator.html]), an open source software package simulating whole transcriptome sequencing experiments with Illumina Genome Analyzer. The software works by first randomly generating integer copies of each splicing isoform according to the annotation file provided by the user, followed by constructing an amplified, size-selected library, and sequencing it *in silico*. The resulting cDNA fragments are then sampled randomly for simulated sequencing, where the initial and terminal of each selected fragment are reported as reads.

The real-world deep sequencing data is often either single-end longer reads or paired-end shorter reads. Our algorithm processes both types of reads with no differences. When necessary, we obtain the reverse complements of certain short reads so that all the reads are of the same orientation. We tested the performance of our algorithm using eight simulated single-end data sets shown in Table I. The short read length is chosen at 100. For the human data, we used the human Ensembl annotation file version 57 along with the human reference genome GRCh37/hg19 in the simulations. For the mouse data, we used the mouse Ensembl annotation file version 61 and the mouse reference genome NCBI37/mm9. For the data with errors, we first used SAMStat ([20]) to profile and calculate the mismatch frequency at each position within the reads from NCBI Short Read Archive, Accession Number SRA011001. Then we applied the error model based on the result from SAMStat to FluxSimulator for generating 100-mer short reads with errors. The error model is shown in Figure 4. For each error position, we assigned an equal probability for A, T, C and G. The simulated RNA-seq datasets are available for download from [http://sammate.sourceforge.net/].

TABLE I
DATASETS FOR THE SIMULATION STUDIES

Dataset	Organism	Number of reads (million)	Error-free
1	Human	15	Yes
2	Human	30	Yes
3	Human	15	No
4	Human	30	No
5	Mouse	15	Yes
6	Mouse	30	Yes
7	Mouse	15	No
8	Mouse	30	No

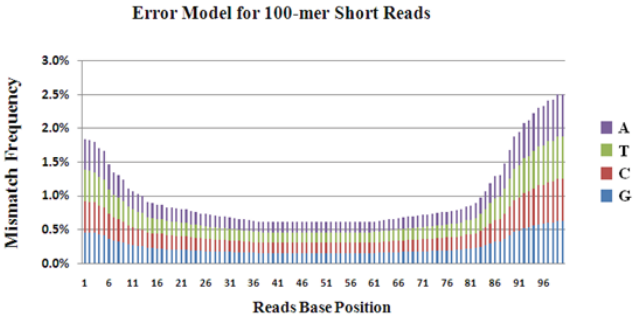


Fig. 4. Error Model for the Simulated Datasets

A. Performance of Localization

For the four human data sets, over 92% split reads are uniquely aligned. The percentage for the mouse data sets is over 91%. For those $7\%-8\%$ of split reads aligned to different chromosomes or different genes in the same chromosome, we localize the reads to the locations where most of the split reads are mapped to. See Figure 5 for a report of the performance of our localization approach on the simulated data sets.

Fig. 5. Performance of Localization.

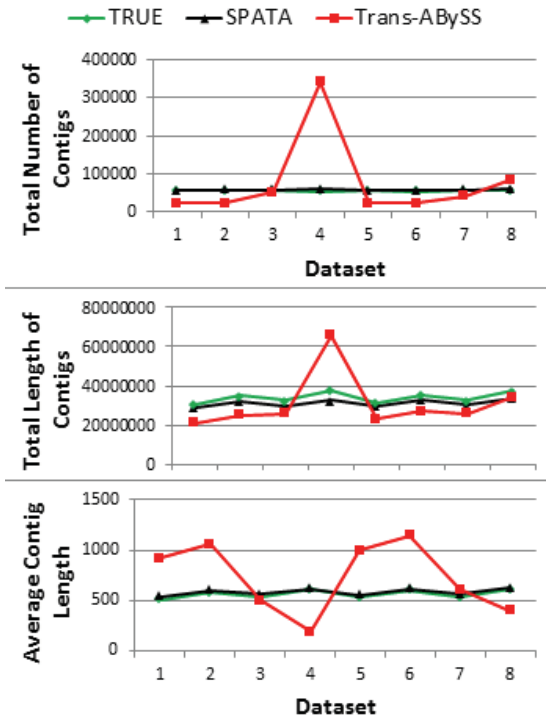


Fig. 6. Comparison of the statistics of contigs.

B. *Performance of* de novo *Assembly*

We compared the output of our assembly program with the ground truth data as well as the output of the Trans-ABySS assembly program. We set the minimum overlap length of our program as $k = 20$. We used the default parameter values for Trans-ABySS if there were any. For each dataset, the ground truth is a set of contigs (a contig is a continuous sequence of bases) directly extracted from the short reads whose locations are known in a simulation by referring to its annotation file. We evaluate the performance of an assembly program according to the following metrics: total number of contigs, total length of contigs, average contig length, % fully covered contigs (a contig is fully covered by another one if it can be completely aligned to the latter), % covered contigs (a contig is covered by others if each part of it is fully covered by one of those contigs), % covered bases ($\frac{Total\ length\ of\ covered\ contigs}{Total\ length\ of\ contigs}$), and the contigs length distribution. We used SSAHA2 ([21]), a sequence alignment by hashing tool, to align two sets of contig sequences stored in the FASTA format. See Figures 6, 7 and 8 for a visual comparison between the ground truth and the two assembly algorithms, SPATA and Trans-ABySS (TA).

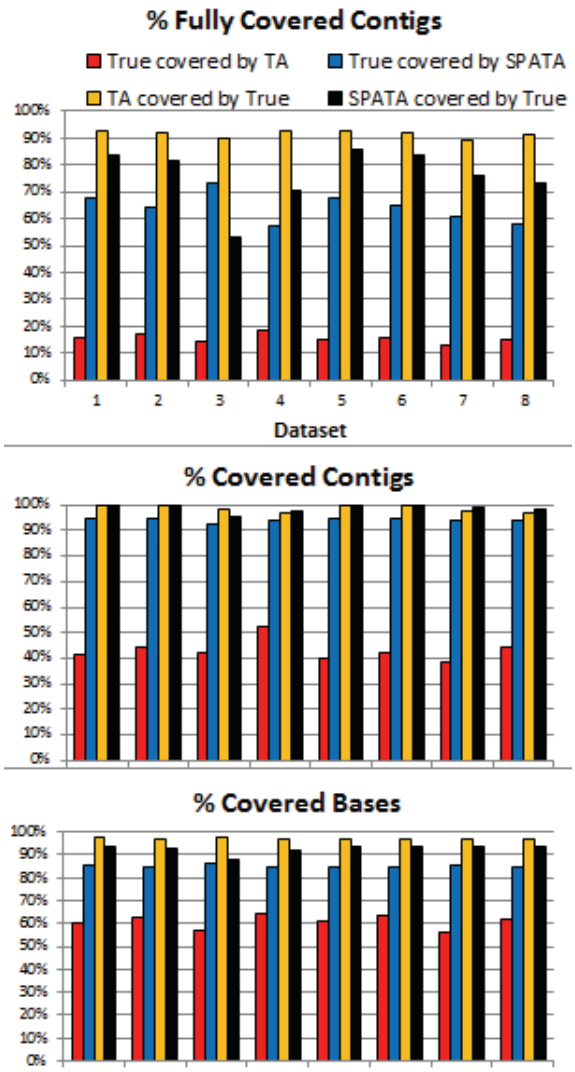


Fig. 7. Comparison of the statistics of contigs.

As shown in Figure 6, the result of SPATA is more consistent with the ground truth than the result of TA. Table II shows the average percentages of contigs and bases coverage based on Figure 7. Similarly, the contig length distribution of SPATA is more consistent with the ground truth than the result of TA (Figure 8), which has a larger variance in the length ranges of ≤ 600 and ≥ 5000.

VI. Conclusion

In this paper, we have presented an accurate yet fast approach for *de novo* transcriptome assembly at the base-level

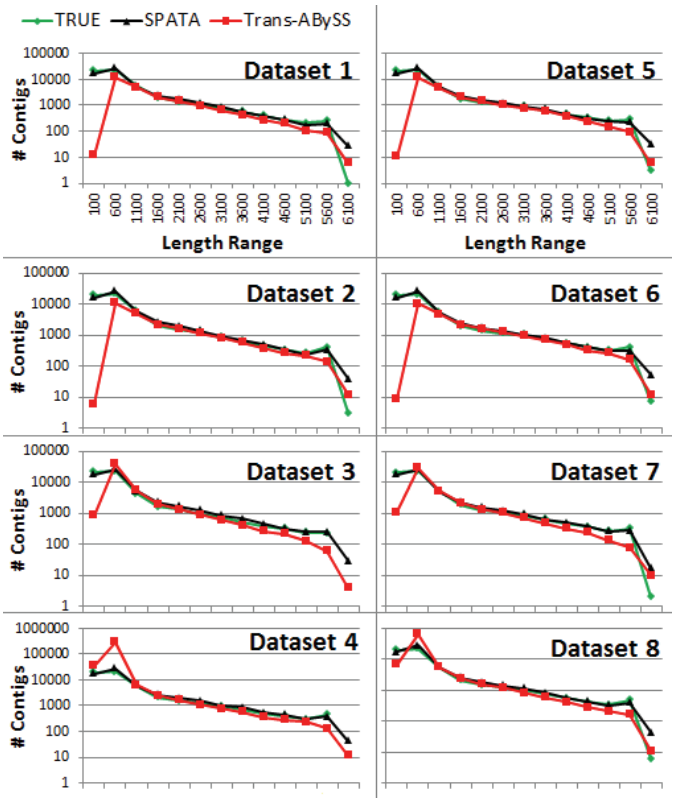


Fig. 8. Comparison of the statistics of contigs.

TABLE II
AVERAGE PERCENTAGES OF COVERAGE

Coverage	Contigs (fully)	Contigs	Bases
True by TA	15.6%	43.2%	60.9%
True by SPATA	64.3%	94.2%	85.2%
TA by True	91.4%	98.7%	97.1%
SPATA by True	75.9%	98.8%	92.6%

resolution using RNA-seq reads. Two original contributions were claimed: the first is read localization that effectively localizes splicing reads and reads with structural variations to the local regions of the human reference genome. It dramatically reduces the computational burden to a very manageable level. The second is a new algorithm to *de novo* assembly all the expressed mRNA transcript structures at the base-level resolution within each local region. Our future effort is to make this approach available to users with a user-friendly interface.

REFERENCES

[1] A. J. Matlin, F. Clark, and C. W. J. Smith, "Understanding alternative splicing: towards a cellular code." *Nature Reviews Molecular Cell Biology*, vol. 6, no. 5, pp. 386 – 398, 2005.

[2] C. A. Maher, C. Kumar-Sinha, X. Cao, S. Kalyana-Sundaram, B. Han, X. Jing, L. Sam, T. Barrette, N. Palanisamy, and A. M. Chinnaiyan, "Transcriptome sequencing to detect gene fusions in cancer." *Nature*, vol. 458, no. 7234, pp. 97 – 101, 2009.

[3] P. Medvedev, M. Stanciu, and M. Brudno, "Computational methods for discovering structural variation with next-generation sequencing," *Nature Methods*, vol. 6, no. 11 Suppl, pp. S13–20, 2009.

[4] X.-S. Wang, J. R. Prensner, G. Chen, Q. Cao, B. Han, S. M. Dhanasekaran, R. Ponnala, X. Cao, S. Varambally, D. G. Thomas, T. J. Giordano, D. G. Beer, N. Palanisamy, M. A. Sartor, G. S. Omenn, and A. M. Chinnaiyan, "An integrative approach to reveal driver gene fusions from paired-end sequencing data in cancer," *Nature Biotechnology*, vol. 27, no. 11, pp. 1005–1011, 2009.

[5] Y. Barash, J. A. Calarco, G. Weijun, P. Qun, W. Xinchen, O. Shai, B. J. Blencowe, and B. J. Frey, "Deciphering the splicing code." *Nature*, vol. 465, no. 7294, pp. 53 – 59, 2010.

[6] K. Wang, D. Singh, Z. Zeng, S. J. Coleman, Y. Huang, G. L. Savich, X. He, P. Mieczkowski, S. A. Grimm, C. M. Perou, J. N. MacLeod, D. Y. Chiang, J. F. Prins, and J. Liu, "Mapsplice: Accurate mapping of rna-seq reads for splice junction discovery," *Nucleic Acids Research*, 2010.

[7] K. F. Au, H. Jiang, L. Lin, Y. Xing, and W. H. Wong, "Detection of splice junctions from paired-end rna-seq data by splicemap," *Nucleic Acids Research*, 2010.

[8] C. Trapnell, L. Pachter, and S. L. Salzberg, "Tophat: discovering splice junctions with rna-seq," *Bioinformatics*, vol. 25, no. 9, pp. 1105–1111, 2009.

[9] M. T. Dimon, K. Sorber, and J. L. DeRisi, "Hmmsplicer: A tool for efficient and sensitive discovery of known and novel splice junctions in rna-seq data," *PLoS ONE*, vol. 5, no. 11, p. e13875, 2010.

[10] R. Bohnert and G. Ratsch, "rquant.web: a tool for rna-seq-based transcript quantitation," *Nucleic Acids Research*, vol. 38, no. suppl 2, pp. W348–W351, 2010.

[11] N. Deng, A. Puetter, K. Zhang, K. Johnson, Z. Zhao, C. Taylor, E. K. Flemington, and D. Zhu, "Isoform-level microrna-155 target prediction using rna-seq," *Nucleic Acids Research*, 2011.

[12] J. M. Kidd, G. M. Cooper, W. F. Donahue, H. S. Hayden, N. Sampas, T. Graves, N. Hansen, B. Teague, C. Alkan, F. Antonacci, E. Haugen, T. Zerr, N. A. Yamada, P. Tsang, T. L. Newman, E. Tzn, C. Ze, H. M. Ebling, N. Tusneem, and R. David, "Mapping and sequencing of structural variation from eight human genomes." *Nature*, vol. 453, no. 7191, pp. 56 – 64, 2008.

[13] 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, B. E. Taillon, Z. Chen, A. Tanzer, A. C. E. Saunders, J. Chi, F. Yang, N. P. Carter, M. E. Hurles, S. M. Weissman, T. T. Harkins, M. B. Gerstein, M. Egholm, and M. Snyder, "Paired-end mapping reveals extensive structural variation in the human genome," *Science*, vol. 318, no. 5849, pp. 420–426, 2007.

[14] S. Lee, E. Cheran, and M. Brudno, "A robust framework for detecting structural variations in a genome," *Bioinformatics*, vol. 24, no. 13, pp. i59–i67, 2008.

[15] 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, D. E. Horsman, J. M. Connors, R. D. Gascoyne, M. A. Marra, and S. J. M. Jones, "De novo transcriptome assembly with abyss," *Bioinformatics*, vol. 25, no. 21, pp. 2872–2877, 2009.

[16] G. Robertson, J. Schein, R. Chiu, R. Corbett, M. Field, S. D. Jackman, K. Mungall, S. Lee, H. M. M. Okada, J. Q. Qian, M. Griffith, A. Raymond, N. Thiessen, T. Cezard, Y. S. Butterfield, R. Newsome, S. K. Chan, R. She, R. Varhol, B. Kamoh, A.-L. L. Prabhu, A. Tam, Y. Zhao, R. A. Moore, M. Hirst, M. A. Marra, S. J. Jones, P. A. Hoodless, and I. Birol, "De novo assembly and analysis of RNA-seq data." *Nature methods*, vol. 7, no. 11, pp. 909–912, 2010.

[17] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, and n. Birol, "Abyss: A parallel assembler for short read sequence data," *Genome Research*, vol. 19, no. 6, pp. 1117–1123, 2009.

[18] 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*, vol. 28, no. 5, pp. 511 – 515, 2010.

[19] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology*, vol. 10, no. 3, pp. R25+, 2009.

[20] T. Lassmann, Y. Hayashizaki, and C. O. Daub, "Samstat: monitoring biases in next generation sequencing data," *Bioinformatics*, 2010.

[21] Z. Ning, A. J. Cox, and J. C. Mullikin, "SSAHA: a fast search method for large DNA databases." *Genome research*, vol. 11, no. 10, pp. 1725–1729, 2001.