Sequence analysis

DART: a fast and accurate RNA-seq mapper with a partitioning strategy

Hsin-Nan Lin and Wen-Lian Hsu*

Institute of Information Science, Academia Sinica, Taipei, Taiwan 115

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on May 2, 2017; revised on August 29, 2017; editorial decision on August 31, 2017; accepted on September 3, 2017

Abstract

Motivation: In recent years, the massively parallel cDNA sequencing (RNA-Seq) technologies have become a powerful tool to provide high resolution measurement of expression and high sensitivity in detecting low abundance transcripts. However, RNA-seq data requires a huge amount of computational efforts. The very fundamental and critical step is to align each sequence fragment against the reference genome. Various de novo spliced RNA aligners have been developed in recent years. Though these aligners can handle spliced alignment and detect splice junctions, some challenges still remain to be solved. With the advances in sequencing technologies and the ongoing collection of sequencing data in the ENCODE project, more efficient alignment algorithms are highly demanded. Most read mappers follow the conventional seed-and-extend strategy to deal with inexact matches for sequence alignment. However, the extension is much more time consuming than the seeding step.

Results: We proposed a novel RNA-seq de novo mapping algorithm, call DART, which adopts a partitioning strategy to avoid the extension step. The experiment results on synthetic datasets and real NGS datasets showed that DART is a highly efficient aligner that yields the highest or comparable sensitivity and accuracy compared to most state-of-the-art aligners, and more importantly, it spends the least amount of time among the selected aligners.

Availability and implementation: https://github.com/hsinnan75/DART

Contact: hsu@iis.sinica.edu.tw

Supplementary information: Supplementary data are available at Bioinformatics online.

1 Introduction

Transcriptome analysis is the genome-wide study of gene structures and activities. It involves the identification of novel transcripts and the quantification of gene expression. DNA microarray technique is a very popular approach to study gene expression levels. However, the probe-target hybridization limits the accuracy of expression measurements, and it also limits to study only those genes for which probes are designed (Zhao et al., 2014). With the emergence of the next generation sequencing (NGS) platforms in recent years, the massively parallel cDNA sequencing (RNA-Seq) technologies have become another powerful tool to provide high resolution measurement of expression and high sensitivity in detecting low abundance transcripts. RNA-Seq does not require prior gene annotation and therefore is able to study unknown transcripts. Recent studies have showed that RNA-Seq demonstrates superior benefits over microarray in transcriptome profiling, though DNA microarrays are irreplaceable and they are still widely used when conducting transcriptional profiling experiments (Fu et al., 2009; Sirbu et al., 2012; Zhao et al., 2014). One of the major reasons is that RNA-Seq data analysis is much more complicated. It requires a huge amount of computational efforts. A typical human RNA-Sequencing experiment can generate up to billions of short sequence fragments. The very fundamental and critical step in the RNA-Seq analysis is to align each sequence fragment (or read) against a reference genome, followed by quantification of genes and identification of differentially expressed genes (Garber et al., 2011).
The alignment of a short read for DNA-resequencing or RNA-Seq against a reference genome is to identify the coordinate where the read originates. However, the task of RNA-Seq alignment is more challenging than that of conventional DNA-sequencing alignment. Although both types of alignments contain mismatches, insertions and deletions due to genomic variations and sequencing errors, the mature mRNA transcripts in eukaryotes are modified and they do not include intragenic regions (or introns) in the gene sequences. Thus a spliced RNA spans one or more introns and concatenates adjacent or distant exons, which may result in the alignment not contiguous and separated by large gaps. Though some DNA-Seq mappers, such as Erange (Mortazavi et al., 2008), OSA (Hu et al., 2012), SpliceSeq (Ryan et al., 2012), etc.) align reads against a transcriptome to avoid introns in the reference, the gene annotation is incomplete for most of the studied organisms. Alignments based on the transcriptome would be biased toward known transcripts. Therefore, de novo spliced RNA aligners are more preferable to analyze RNA-Seq reads for detecting novel splice junctions.

Various de novo spliced RNA aligners have been developed in recent years, which include QPALMA (De Boneta et al., 2008), TopHat (Trapnell et al., 2009), GSNAP (Wu and Nacu, 2010) (and GSTRUCT is its successive version), PALMapper (Jean et al., 2010), MapSplice (Wang et al., 2010), RUM(Grant et al., 2011), GEM (Marco-Sola et al., 2012), STAR(Dobin et al., 2013), TopHat2 (Kim et al., 2013), HISAT (Kim et al., 2015) and Subread (Liao et al., 2013). In summary, most of the short read aligners basically adopt the seed-and-extend strategy (Li and Homer, 2010), which is sequential in nature and takes much longer time on the extension step for dealing with mismatches. In the seeding step, aligners use either a hash table or a suffix array (SA)/Burrows Wheeler Transform (BWT) index (Wheeler, 1994) to perform seed exploration. A hash table based aligner uses all the subsequences of k-mers to obtain occurrence locations. In contrast, a SABWT based aligner finds the maximal exact matches (MEM) between the read sequence and the reference genome. An MEM is the maximal exact match between two sequences, which cannot be extended further without allowing mismatches. MEMs have been widely used as seeds for whole genome alignment and NGS read alignment (Choi et al., 2005; Li and Durbin, 2009; Liu and Schmidt, 2012). In the extension step, a dynamic programming algorithm or combined with a heuristic algorithm are often used to deal with mismatches or indels in the read alignments.

An exonic read is a read that can be aligned completely within the corresponding exon and its alignment against the reference genome is relatively straightforward. In contrast, the more challenging task in RNA-Seq alignment is to deal with the reads that span one or more introns. They are referred to as spanned reads in this article. Many above-mentioned RNA-Seq aligners adopt similar strategy to handle spanned reads. A spanned read is split into appropriate fragments and each fragment can be aligned continguously to a reference genome, and all the desirable sub-alignments within appropriate intron sizes are merged to form the complete alignment. However, these aligners differ in how they handle splice junction alignment.

QPALMA used Support Vector Machines (SVMs) to learn splice junctions from a transcriptome dataset. It then infers spliced alignments from seed regions based on the trained scoring functions. TopHat finds junctions in two phases. It uses Bowtie/Bowtie2 to map all reads to the reference and then assembles the mapped reads to generate contiguous consensus sequences (called islands). It then constructs all candidate splice junctions from neighboring islands that could form canonical (GT-AG) introns. GSNAP identifies splice junctions with probabilistic models which are derived from the frequencies of nucleotides neighboring splice sites. MapSplice partitions a read sequence into some consecutive segment and identifies exonic alignment for each segment. If a segment contains spliced alignment, the splice junction can be easily discovered using the double-anchored search method between the neighboring segments with exonic alignment. STAR finds the Maximal Mappable Prefix (MMP, similar to MEM) for exonic alignment. It repeatedly finds the MMPs for the unmapped portion of the read, therefore, the splice junctions can be discovered naturally. HISAT is the first aligner that employs a hierarchical indexing strategy (global and local FM indexing) for spliced alignment. It applies different strategies to handle different exonic and spliced alignment types. Subread detects splice junctions with a two-scan procedure. It identifies junction sites between the best two mapping locations for each read (first scan), and then validates junctions by examining all mapping possibilities (second scan).

Most of the above-mentioned aligners require considerable computing time to increase alignment sensitivity and accuracy. With the advances in sequencing technologies and the ongoing collection of sequencing data in the ENCODE project, more efficient algorithms are highly demanded to handle the huge amount of short reads as well as the associated sequence variations (Engstrom et al., 2013). Furthermore, some aligners might fail to detect splice junctions if sequencing errors happen at their neighboring locations. In this study, we proposed a novel RNA-Seq mapping algorithm, call DART (Division based Alignment for RNA-Seq Transcripts) to handle spliced alignment without any annotation guidance. It is derived from our DNA read mapper, Kart (Lin and Hsu, 2017). DART adopts a partitioning strategy to handle RNA-Seq transcript alignments. Unlike most of read aligners that try to extend a seed in both directions with a dynamic programming step, DART divides a read sequence into one or more segments to replace the seed extension step. The experiment results on synthetic datasets and real datasets show that DART is a highly efficient aligner that yields the highest sensitivity and accuracy and spends the least amount of time among the selected aligners.

We describe the details of the alignment algorithm of DART in the Methods section. Then we analyze and compare its performance with some selected state-of-the-art aligners in the Results section. DART can be freely downloaded from https://github.com/hsinnan75/DART.

2 Materials and methods

2.1 Overview of algorithms

A unique feature of DART is that we adopt a partitioning strategy to handle the matches and mismatches separately between read sequences and the reference genome. DART divides a read alignment into two groups: simple region pairs (abbreviated as simple pairs) and normal region pairs (normal pairs), where all simple pairs have perfect alignment (exact match) and normal pairs require un-gapped or gapped alignment (due to mismatches or indels). Both simple pairs and normal pairs are referred to as fragment pairs. Once the fragment pairs are identified, they can be processed and aligned separately and the final mapping result is simply the concatenation of the alignment of each fragment pair.

The mapping algorithm of DART consists of two main steps: seed exploration and candidate alignment processing. Figure 1 illustrates the idea of the algorithm. Given a read sequence R, DART identifies all simple pairs with a BWT search algorithm in the seed exploration step. In the candidate alignment step, adjacent simple
pairs are clustered according to their coordinates. For example, the four simple pairs $SP_A$, $SP_B$, $SP_C$, and $SP_D$ in Figure 1 are clustered together because they are aligned in the neighboring regions. Then DART fills the gaps between simple pairs if they are in the same exonic region. The gap between $SP_A$ and $SP_B$ appears because a deletion happens in the read sequence and that between $SP_C$ and $SP_D$ appears because a sequencing error occurs in the read sequence. Thus, DART will generate the corresponding normal pairs for the two intra-exonic gaps. Adjacent simple pairs and the supplementary normal pairs form a complete alignment for the given read sequence. It is noteworthy that the gap between $SP_B$ and $SP_C$ is due to the splice junction and it is referred to as an intragenic gap. There is no need to add a normal pair for such gaps. DART discovers splice junctions according to the intragenic gaps between adjacent fragment pairs. The detailed implementation of the two main steps are described below.

### 2.2 Seed exploration

Consider a read sequence $R$, the reference genome $G$, and the BWT array constructed from $G$ and its reverse sequence $G'$. For simplicity and without losing generality, assume $G$ is the concatenation of $G$ and $G'$ in the following description. Let $R[i_1]$ be the $i_1$th nucleotide of $R$, and $R[i_1, i_2]$ be the subsequence between $R[i_1]$ and $R[i_2]$. Similarly, let $G[j_1]$ be the $j_1$th nucleotide of $G$, and $G[j_1, j_2]$ be the subsequence between $G[j_1]$ and $G[j_2]$. A locally maximal exact matches (LMEMs) on the BWT array of length $l$ is defined as a common substring between $R[i_1, i_2]$ and $G[j_1, j_2]$ (i.e. $R[i_1, i_2] = G[j_1, j_2]$) and it cannot be extended in either direction of $R[i_1, i_2]$ and $G[j_1, j_2]$ without allowing for a mismatch. This LMEM is denoted by a 4-tuple $(i_1, i_2, j_1, j_2)$ where $i_2 - i_1 = j_2 - j_1 = l - 1$. We use $\text{APos} = (l_1, l_2)$ to represent the position difference of an LMEM.

Dart finds all LMEMs by traversing a BWT array. The traversal algorithm is identical to that described in (Li and Durbin, 2009). Readers who are interested in the search algorithm are referred to the article. The traversal starts from $R[i_1]$ and stops at $R[i_2]$ if the exact matching meets a mismatch at $R[i_2+1]$, i.e. $R[i_1, i_2]$ is a substring of the reference sequence, whereas $R[i_1, i_2+1]$ is not. The next LMEM exploration will start from $R[i_2+1]$ until it reaches the end of the read sequence. DART only keeps those LMEMs whose sizes are no less than a predefined threshold $k$ and whose occurrences are less than 50. The value $k$ is determined based on the size of the reference genome. Each qualified LMEM is then converted into one or more simple pairs according to the occurrences. If $R[i_1, i_2]$ has multiple copies in $G$, then each copy is denoted by a 4-tuple respectively. For example, $R[1, 20]$ is the longest substring that matches two substrings of $G$, say $G[501, 520]$ and $G[1001, 1020]$, thus the LMEM will be converted into two simple pairs, which are $(1, 20, 501, 520)$ and $(1, 20, 1001, 1020)$, respectively. The simple pairs $SP_A$, $SP_B$, $SP_C$ and $SP_D$ in Figure 1 are examples of LMEMs found by the array traversal. In the toy example, the array traversal breaks due to an indel error, a splice junction and a mismatch, respectively, and finally it reaches the end of the read sequence.

![Fig. 1. The mapping idea of DART. The mapping can be divided into simple and normal pairs to deal with exact matches and mismatches separately.](https://academic.oup.com/bioinformatics/article-abstract/34/2/190/4104410)

### 2.3 Candidate alignment processing

All the simple pairs identified by the seed exploration step are sorted by their genomic locations. Adjacent simple pairs are clustered together if their genomic locations are within a user-defined distance threshold. This threshold can be defined as the maximal intron size. Simple pairs that are within the maximal intron size are considered in the same transcript and therefore should be put in the same cluster. The clustering procedure starts from the first simple pair on the list and checks if the next simple pair is within the distance threshold with the previous one. If they are truly neighboring, we put them in the same cluster and check the next simple pair with the newly added one. Otherwise, the current cluster is no longer expanded and a new cluster is created for the simple pair. The clustering procedure continues until all the simple pairs are clustered. Each cluster is then evaluated by the total length of its simple pairs. To avoid delay in processing clusters that are unlikely to be true alignment, DART only keeps clusters which produce the longest length (denoted as $L$) or above $L - 20$. However, if the input data consists of paired-end reads, DART will first compare the clusters of two paired-end reads and keep all the clusters that meet the paired-end conditions.

A read alignment will be built from a simple pair cluster. Two simple pairs in the same cluster could overlap due to tandem repeats or sequence variations. In such cases, the overlapped portion in the genome and read portion will be chopped off from the shorter simple pair to ensure that all simple pairs are non-overlapping. Sometimes simple pairs could intersect to each other. In such cases, we remove the simple pairs that cause this dislocation. If the simple pairs in a cluster exhibit gaps either in the read sequence or genome sequence, DART will identify the corresponding normal pairs to fill all the gaps. Note that the intragenic gaps are ignored since their presence is due to splice junctions. The gaps of the read sequence could be either aligned with the same exonic region of the respective simple pairs, or split into two segments to form a spliced alignment if the adjacent simple pairs are mapped to different exons.

Figure 2 gives examples to illustrate the two cases. In Figure 2A, there are some uncovered nucleotides between $SP_A$ and $SP_B$, three in the read portion and two in the genome portion. Since the APoS of $SP_A$ and $SP_B$ are $310 - 311 - 1$ and $309 - 333 - 24$, respectively, it suggests that $SP_A$ and $SP_B$ are in the same exon. In such cases, DART simply inserts a normal pair $(21, 23, 331, 332)$ in between $SP_A$ and $SP_B$ to fill the gaps. It can be also observed that gaps could span one or more introns. As shown in Figure 2B, there are five uncovered nucleotides in the read sequence and simple pairs $SP_C$ and $SP_D$ are in different exons since their APoS difference $(100 vs. 487)$ is above the predefined minimal intron size (the default value is 5). In such cases, the gaps could be split into two parts if they cover two

![Fig. 2. (A) Gaps between simple pairs in an exonic read. (B) Gaps between simple pairs in a spanned read.](https://academic.oup.com/bioinformatics/article-abstract/34/2/190/4104410)
exonic regions. Let two adjacent simple pairs be \( (i_1, i_2, i_3, i_4) \) and \( (i_3, i_4, i_5, i_6) \), respectively, and suppose \( i_3 - i_2 = k > 1 \). Thus, \( R[i_2 + 1, i_3 - 1] \) represents the uncovered nucleotides of length \( k \) in the read portion, and \( G[i_2 + 1, i_3 - 1] \) represents the uncovered nucleotides in the genome portion. DART aligns the fragments \( R[i_2 + 1, i_3 - 1] \) and \( G[i_2 + 1, i_3 + k] \) with the Needleman-Wunsch algorithm, and it also aligns the fragments \( R[i_2 + 1, i_3 - 1] \) and \( G[i_3 - k, i_3 - 1] \). Then DART finds the cut point \( p \) to maximize the identical pairs of the two alignments, so that the alignment of \( R[i_2 + 1, i_2 + p] \) and \( G[i_2 + 1, i_2 + p] \), and that of \( R[i_2 + p + 1, i_3 - 1] \) and \( G[i_3 - k + p, i_3 - 1] \) produce the highest alignment score. In the example of Figure 2B, the best choice of \( p = 2 \), so that the two fragment pairs \((19, 20, 119, 120)\) and \((21, 23, 508, 510)\) produce the highest alignment score. Thus the two fragment pairs are inserted as normal pairs between SPC and SPD to cover the entire read sequence. Though the Needleman-Wunsch algorithm is used to identify the cut point, the gap sizes are normally very small and it does not take much time on the alignments.

It is also noteworthy that a fragment pair might cover intronic region accidentally. A canonical splice junction starts with the dinucleotide GT (donor site) and ends with the dinucleotide AG (acceptor site). If another dinucleotide GT is closely adjacent the acceptor site, it would be mapped to the donor site by accident. Figure 3 gives an example of such cases. It can be seen that the dinucleotide GT at \( 3 \) gives an example of such cases. It can be seen that the dinucleotide GT (donor site) and ends with the dinucleotide AG (acceptor site), it would be mapped to the donor site by accident. Figure 3 – 1 represents the uncovered nucleotides of length \( j \) in the read portion, and \( j_1, j_2, j_3 \), respectively, and suppose \( j_3 = j_1 + j_2 \). Then DART checks if \( j_3 = j_1 + j_2 \). If another dinucleotide GT is closely adjacent the acceptor site, it would be mapped to the donor site by accident. Figure 3 will be modified as \((1, 18, 739, 756)\) and \((19, 42, 811, 834)\), respectively with \( shift = 834 \), respectively with \( shift = 834 \). Thus, the two fragment pairs are inserted as normal pairs between SPC and SPD to cover the entire read sequence.

3 Results

3.1 Implementation and experiment design

DART was developed under Linux environment and implemented with standard C/C++. It supports multi-thread to take advantage of multi-core computers. DART reads a BWT-based index file and takes a read library (single-end or paired-end reads) in FASTA/FASTQ format as input. DART reports read alignments in the SAM (Sequence Alignment/Map) format (Li et al., 2009). It is difficult to estimate the correctness of read alignments using real datasets since the true coordinate of each read sequence is unknown. Therefore, we created simulated read libraries to estimate the performance of read aligners. Here, we simulated read libraries of the human genome (Hg38, size: 3 G bp) using the Flux simulator (Griebel et al., 2012), a popular software to simulate RNA-Seq experiments in silico. Flux simulates the RNA sequencing protocols and produces the read distributions observed in practice fairly well. Simulated RNA-seq reads were generated from the known transcripts (GENCODE release 25) (Harrow et al., 2012) on the entire human genome using the 76 bp error model by Flux simulator. To test the capabilities of RNA-seq aligners, we generated four Illumina-like paired-end read datasets with different read lengths: 76, 101, 151 and 251 bp. They are labeled as SimRead_76, SimRead_101, SimRead_151 and SimRead_251, respectively. Each dataset contains around 40 million paired-end reads.

Most of RNA-Seq aligners are evaluated by the alignment sensitivity, mapping accuracy, true-/false-positive rates of splice junction detection, and mapping speed (Conesa et al., 2016; Dobin et al., 2013; Engstrom et al., 2013; Li and Homer, 2010). We follow the same benchmarking metrics to evaluate performance of DART and compare it with other RNA-Seq aligners. Since Flux simulator only provides the transcript level coordinate, we estimate the average sequence identity (called SeqIdy) of read alignments to reveal the accuracy of base-to-base alignment. A read alignment is considered a true one if its mapping coordinate is within the original transcript. A predicted splice junction is considered as a true one if it meets the boundary of a true splice junction with a maximal difference of 5 bp (due to the consideration of genome alteration); otherwise it is considered as a false splice junction. Suppose an RNA-Seq aligner handles a library of \( N \) reads and reports at least one alignment for \( N' \)
reads and it predicts $M$ distinct splice junctions. The resulting alignments are then compared with the original transcript level coordinates of the simulated reads, and the predicted splice junctions are verified with the gene annotation (GENCODE release 25). The alignment accuracy is estimated only on the read alignment with $\text{MAPQ} \geq 0$. Suppose there are $n$ reads with $\text{MAPQ} \geq 0$ and $n'$ reads are mapped to the correct transcript and $m$ splice junctions agree with the gene annotations. Thus the benchmarking metrics are defined as follows:

$$\text{sensitivity} = \frac{n'}{n};$$
$$\text{accuracy} = \frac{n'}{n};$$
$$\text{recall} = \frac{n'}{n};$$
$$\text{SeqIdy} = \frac{\text{identical base pairs}}{\text{alignment length}};$$
$$\text{splice junction accuracy} = \frac{m}{M}.$$

Note that TopHat2 and Subread do not output the splice junctions directly, we used bed_to_juncs (a program in the TopHat2 package) to generate the predicted splice junctions from the output file of junctions.bed.

We also downloaded four recently released read libraries to measure the performance in practice on the SRA database (Leinonen et al., 2011). They are SRR3351428, ERR1518881, SRR3439468 and SRR34394988. These data comprise 278, 706, 960 paired-end reads. The read length ranges from 100 to 151 bp. Since the true genomic origins of the real datasets are unknown, we estimate the performance of aligners with the following objective criteria: sensitivity, SeqIdy, reported splice junctions, true splice junctions, splice junction accuracy and runtime. To avoid estimation bias due to multiple hits (i.e. ambiguous mapping), we only evaluated the first alignment for each read.

All reads in the test data were processed on a Linux 64-bit system with 4 Intel Xeon E7-4830 2.13 GHz CPUs and 2TB physical memory. DART was compared with the following existing RNA-seq read aligners: STAR, TopHat2, Subread, MapSplice2 and HISAT2. The comparison was conducted with the default parameters of each aligner since the above-mentioned aligners were developed and optimized for human genomes and recent RNA-seq data. It is a reasonable and commonly accepted practice (Dobin et al., 2013). Each aligner was asked to only report the best alignment or a random best if there were multiple hits. All aligners were run in the de novo mode (i.e. without using transcript annotations) with 16 threads to speed up the whole procedure. The mapping process is forced to terminate if the whole dataset cannot be finished within 24h. We mark NA in the measurement for such cases. The arguments as well as version number of each aligner are summarized in the Supplementary data (Supplementary Table S1).

### 3.2 Evaluation on simulated datasets

Table 1 summarizes the performance evaluation of selected RNA-Seq aligner on the simulated datasets. It is observed that DART produced the highest or comparable sensitivity, accuracy, recall and SeqIdy. Its performance exhibited consistency among datasets with various read lengths. It is also noteworthy that DART spent 576 s on the whole simulated datasets. STAR also produced the high accuracy and SeqIdy, but its sensitivity and recall decreased when read length became 251 bp long. The sensitivity and recall of STAR on the SimRead_251 were 0.939 and 0.921, respectively, which were much lower than those of DART (0.997 and 0.971). STAR also spent much more time on the SimRead_251. STAR spent 850 s on the whole simulated datasets. HISAT2 ran faster than TopHat2, Subread and MapSplice2; however, its sensitivity and recall were much lower than those of DART and STAR. It left many reads unaligned. Subread produced better alignments than Tophat2 and HISAT2. Its sensitivity and recall were higher than the two aligners. However, its alignment speed was not fast enough. MapSplice2 produced comparable alignments. Its sensitivity, recall and SeqIdy were comparable to those of DART and STAR, but it spent much more time to yield good quality alignments. Tophat2 was the slowest aligner among the selected methods. It spent 63107 s on the whole simulated datasets. Moreover, Tophat2 produced the worst sensitivity and recall. It is also observed that all the selected aligners except DART and MapSplice2 produced relatively low sensitivity when the read length became longer.

For the identification of splice junctions, Table 1 also shows that DART, Subread, MapSplice2 and HISAT2 produced similar accuracy on the splice junction detection. Their splice junction accuracies were around 0.95–0.96. The number of true splice junctions of each aligner increased when the read length became longer. It suggests that longer reads provide better splice junction detection for RNA-Seq data analysis. For example, the numbers of true splice junctions identified by DART on the four simulated datasets were 96700, 102162, 108771 and 111487, respectively. Among all the selected aligners, TopHat2 produced less accurate and fewer numbers of splice junctions. With the simulated RNA-Seq datasets, we demonstrated that DART is a highly accurate and fast aligner. It is not only the fastest RNA-Seq aligner, but it also produces the most accurate or comparable alignments among the state-of-the-art aligners. DART is also less sensitive to the read length. It yields consistent alignment sensitivities on reads with different read lengths.

### 3.3 Evaluation on real datasets

Table 2 summarizes the performance of the selected aligners on the real datasets. In this benchmark, it can be observed that DART produced the highest or comparable sensitivity and SeqIdy on all the datasets and it was also the fastest. DART and MapSplice2 yielded similar sensitivity; however, DART produced higher SeqIdy than MapSplice2, and MapSplice2 spent much more time to generate similar alignments. STAR was also the second fastest aligner on real datasets, however, its sensitivity and SeqIdy were not as good as those of DART and MapSplice2. We also noticed that MapSplice2 spent much more time on the dataset of SRR3351428. It seemed like MapSplice2 had difficulty in dealing with some particular reads in that dataset. HISAT2 yielded lower sensitivity though it run faster than MapSplice2. Subread was faster than MapSplice2 on the datasets of SRR3351428 and ERR1518881, but it produced less sensitivity. Moreover, it could not finish the mapping process on the remaining two real datasets within 24h. TopHat2 did not perform very well on the real datasets. Its sensitivity was the lowest among the tested methods and it also failed to finish the alignments on the last two real datasets within the time limit.

For the splice junction detection on real datasets (shown in Table 2), DART, TopHat2, Subread and MapSplice2 produced similar results. It suggests that the splice junction detections of those methods were very consistent on the real datasets. Though HISAT2 achieved higher accuracy on the splice junction detection, it produced less number of true splice junctions. STAR produced lower splice junction accuracy on the real datasets, though the numbers of observed splice junctions were similar to other methods.

We also compared the memory usage of each aligner. Though some aligners allow users to set the maximal memory usage, we did not give any limitations and let each aligner take as much memory as it needs. In Table 3, we found that TopHat2, HISAT2 and
MapSplice2 required less physical memory, followed by Subread and DART which required 10 and 12 GB, respectively. STAR required around 30 GB of physical memory. Note that the memory requirement was measured on the simulated and real datasets running with 16 threads.

All the selected aligners were evaluated with 16 threads, we further analyzed the efficiency of DART with different number of threads. Table 4 shows the runtime of DART on the dataset of SimRead_76. It can be observed that DART is highly efficient to utilize multi-threads. The runtime is nearly half as the number of threads doubles. However, the efficiency still degenerates as the number of threads increase due to the disk overhead. Since all the threads need to access the same files for input and output, they must wait to gain the exclusive file accesses. Thus, not all
In this article, we present DART, a new de novo RNA-seq aligner for sensitive, rapid and accurate mapping to reference sequences. DART is a BWT-based aligner and it adopts a partitioning strategy to divide a read into simple and normal pairs. Each simple pair is a perfect alignment and each normal pair is a gapped/un-gapped alignment.

By benchmarking on the simulated and real datasets, we demonstrate that the proposed partitioning mapping strategy can replace the extension step in the conventional seed-and-extend strategy and reduce the time of alignment. We show that DART was able to align 160 million simulated paired-end reads with various lengths in 576 s, while the second fastest aligner, i.e. STAR took 850 s. For the real datasets, DART aligned around 278 million paired-end reads in 1521 s, whereas STAR took 2048 s on the same datasets. Though DART is faster, it can still generate accurate alignments and yield high sensitivity regardless of read length. It also produces high sequence identity of alignments. In addition, it detects comparable or larger number of splice junctions than other aligners. The accuracy of predicted splice junctions on the simulated datasets is between 0.96 and 0.97, and that on the real datasets is between 0.63 and 0.79. Both of which are comparable to or better than the selected aligners.

The sensitivity of read mapping is one of the important factors for further analysis, such as gene expression level measurement or structural variants detection. Specifically, orphan reads and one-end anchored (OEA) are often used to identify sequence variants. Orphan reads refer to those paired-end reads where neither read sequences can be aligned with high sequence identity to the reference genome, and OEA refer to those paired-end reads in which one of the paired reads is aligned to the reference genome. Thus, mapping sensitivity and the number of paired alignments are crucial for further read analysis. We have shown that DART is capable of producing high-sensitivity mapping and generating more paired alignments with the least amount of time. With the advances in sequencing technologies and the ongoing collection of sequencing data in the ENCODE project, we believe DART is a better aligner to handle the huge amount of short reads as well as the associated sequence variants detection.

### 4 Conclusions

In this article, we present DART, a new \textit{de novo} RNA-seq aligner for sensitive, rapid and accurate mapping to reference sequences. DART is a BWT-based aligner and it adopts a partitioning strategy to divide a read into simple and normal pairs. Each simple pair is a perfect alignment and each normal pair is a gapped/un-gapped alignment.

By benchmarking on the simulated and real datasets, we demonstrate that the proposed partitioning mapping strategy can replace the extension step in the conventional seed-and-extend strategy and reduce the time of alignment. We show that DART was able to align 160 million simulated paired-end reads with various lengths in 576 s, while the second fastest aligner, i.e. STAR took 850 s. For the real datasets, DART aligned around 278 million paired-end reads in 1521 s, whereas STAR took 2048 s on the same datasets. Though DART is faster, it can still generate accurate alignments and yield high sensitivity regardless of read length. It also produces high sequence identity of alignments. In addition, it detects comparable or larger number of splice junctions than other aligners. The accuracy of predicted splice junctions on the simulated datasets is between 0.96 and 0.97, and that on the real datasets is between 0.63 and 0.79. Both of which are comparable to or better than the selected aligners.

The sensitivity of read mapping is one of the important factors for further analysis, such as gene expression level measurement or structural variants detection. Specifically, orphan reads and one-end anchored (OEA) are often used to identify sequence variants. Orphan reads refer to those paired-end reads where neither read sequences can be aligned with high sequence identity to the reference genome, and OEA refer to those paired-end reads in which one of the paired reads is aligned to the reference genome. Thus, mapping sensitivity and the number of paired alignments are crucial for further read analysis. We have shown that DART is capable of producing high-sensitivity mapping and generating more paired alignments with the least amount of time. With the advances in sequencing technologies and the ongoing collection of sequencing data in the ENCODE project, we believe DART is a better aligner to handle the huge amount of short reads as well as the associated sequence variants detection.

### Acknowledgements

We appreciate the advice and discussion from Dr. Ei-Wen Yang. The BWT indexing and BWT array search procedure of DART were modified from BWA. We would also like to thank Thasso Griebel \textit{et al}. for developing the RNA-seq simulator.

### Funding

Bioinformatics Core Facility for Translational Medicine and Biotechnology Development/Ministry of Science and Technology (Taiwan) 105-2319-B-400-002.

\textbf{Conflict of Interest}: none declared.

### References


### Table 3. Comparison of the peak RAM usage

<table>
<thead>
<tr>
<th>Aligner</th>
<th>Memory usage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DART</td>
<td>12.0</td>
</tr>
<tr>
<td>STAR</td>
<td>30.0</td>
</tr>
<tr>
<td>TopHat2</td>
<td>4.5</td>
</tr>
<tr>
<td>Subread</td>
<td>10.0</td>
</tr>
<tr>
<td>MapSplice2</td>
<td>6.3</td>
</tr>
<tr>
<td>HISAT2</td>
<td>5.6</td>
</tr>
</tbody>
</table>

### Table 4. The scaling of runtime of DART on the dataset SimRead_76

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Threads</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimRead_76</td>
<td>1</td>
<td>937</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>464</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>240</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>126</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>71</td>
</tr>
</tbody>
</table>


