Bioinformatics, 34(2), 2018, 190–197
doi: 10.1093/bioinformatics/btx558
Advance Access Publication Date: 5 September 2017
Original Paper

OXFORD

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 Author 2017. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/4.0/), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact journals.permissions@oup.com

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 concatenate adjacent or distant exons, which may result in the alignment not contiguous and separated by large gaps. Though some RNA-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 Bona *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 SA/BWT 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 contiguously 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, we assume G is the concatenation of G and G' in the following description. Let $R[i_1]$ be the i_1th 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_1th 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 $\Delta Pos = (j_1 - i_1)$ 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.

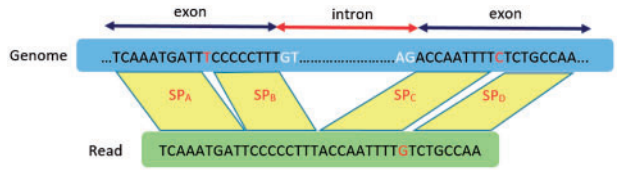

Fig. 1. The mapping idea of DART. The mapping can be divided into simple and normal pairs to deal with exact matches and mismatches separately

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 ΔPos 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 ΔPos 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

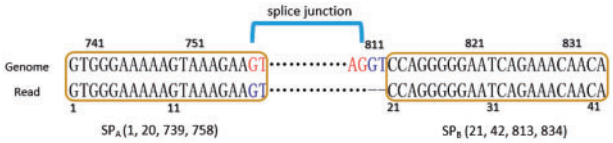

Fig. 3. Identification of the splice junction between two simple pairs A and B. The simple pair A should be shrunk by two nucleotides to match the splice site 'GT/AG' in the splice junction

exonic regions. Let two adjacent simple pairs be (i_1, i_2, j_1, j_2) and (i_3, i_4, j_3, j_4), 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 $G[j_2 + 1, j_3 - 1]$ represents the uncovered nucleotides in the genome portion. DART aligns the fragments $R[i_2 + 1, i_3\_1]$ and $G[j_2 + 1, j_2 + k]$ with the Needleman-Wunsch algorithm, and it also aligns the fragments $R[i_2 + 1, i_3 - 1]$ and $G[j_3 - k, j_3 - 1]$. Then DART find 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 $G[j_2 + 1, j_2 + p]$, and that of $R[i_2 + p + 1, i_3 - 1]$ and $G[j_3 - k + p, j_3 - 1]$ produce the highest alignment score. In the example of Figure 2B, the best choice of p is 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 SP_C and SP_D 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 R[19, 20] is mistakenly mapped to G[757, 758] which is the donor site of the corresponding splice junction. R[19, 20] should be mapped to G[811, 812] which is closely adjacent with the acceptor site. Though the alignment score stays the same whether R[19, 20] is mapped either to G[757, 758] or to G[811, 812], the incorrect mapping can result in wrong splice junction detection.

To avoid mapping to intronic regions, DART refines the splice junction sites by checking two adjacent fragment pairs if they are mapped to different exons (like SP_A and SP_B in Fig. 3). Given two adjacent fragment pairs, denoted as (i_1, i_2, j_1, j_2) and (i_3, i_4, j_3, j_4), respectively, DART checks if $G[j_2+1+shift, j_2+2+shift]$ is a donor site and $G[j_3-2+shift, j_3-1+shift]$ is an acceptor site where *shift* goes with 0, ±1, ±2, ... and ±9 sequentially until the splice site pair are checked. If the splice junction is found with $shift \neq 0$, the sizes of the corresponding fragment pairs are modified accordingly. DART checks the most four common splice sites: 'GT/AG', 'CT/AC', 'GC/AG' and 'CT/GC'. For example, the simple pairs SP_A and SP_B in Figure 3 will be modified as (1, 18, 739, 756) and (19, 42, 811, 834), respectively with $shift = -2$.

Finally, DART generates all sub-alignments based on the fragment pairs in the cluster. The sub-alignment for a simple pair is a perfect alignment without any mismatches, whereas the sub-alignment for a normal pair can be un-gapped alignment (with only mismatches) or gapped alignment (with indels). If the read portion and genome portion of a normal pair have equal size, then it is very likely the normal pair only contains substitution errors and the un-gapped alignment makes the best alignment; however, if a normal pair contains indel errors, the un-gapped alignment will result in low sequence identity. So, by checking the percentage of mismatches with a linear scan, we can determine whether a normal pair requires gapped alignment or not. DART performs the Needleman-Wunsch algorithm to generate gapped alignments. All the sub-alignments are concatenated together to form the final alignment. If there are more clusters to be considered for the same read, DART repeats the candidate alignment processing step to generate alternative alignments.

2.4 Mapping quality score

MAQ (Li *et al.*, 2008) introduced the idea of mapping quality to estimate the reliability of a read alignment. It can be converted into the probability of a query sequence being aligned incorrectly. The mapping quality is estimated based on the uniqueness of optimal alignment. An alignment generated by Dart is assigned with a MAPQ based on the following rules:

50 = unique mapping;
3 = maps to 2 locations;
2 = maps to 3 locations;
1 = maps to 4–9 locations;
0 = maps to 10 or more locations.

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 coordinate 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 MAPQ > 0. Suppose there are n reads with MAPQ > 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:

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

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 *SRR3439488*. 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 24 h. 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 theSimRead_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 63 107 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 96 700, 102 162, 108 771 and 111 487, 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 24 h. 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

Table 1. Performance comparison of DART and other selected aligners on the simulated datasets

Synthetic datasets	Aligner	Sensitivity	Accuracy	Recall	SeqIdy	Reported SJ	True SJ	SJ accuracy	Runtime
SimRead_76	DART	0.991	0.989	0.957	0.999	99761	96700	0.969	71
	STAR	0.978	0.981	0.958	0.996	108202	101163	0.935	129
	TopHat2	0.852	0.961	0.853	0.998	102230	93850	0.918	6172
	Subread	0.965	0.988	0.929	0.998	99033	95469	0.964	2610
	MapSplice2	0.962	0.976	0.940	0.997	101230	97895	0.967	3602
	HISAT2	0.911	0.977	0.889	0.999	100589	96922	0.964	353
SimRead_101	DART	0.992	0.988	0.965	0.997	105584	102162	0.968	95
	STAR	0.977	0.982	0.958	0.996	112674	105459	0.936	154
	TopHat2	0.809	0.967	0.809	0.999	109153	99501	0.912	10357
	Subread	0.955	0.987	0.925	0.998	105269	101136	0.961	2346
	MapSplice2	0.979	0.980	0.960	0.998	110219	104434	0.948	4736
	HISAT2	0.898	0.979	0.879	0.998	104309	100633	0.965	384
SimRead_151	DART	0.996	0.989	0.971	0.994	112614	108771	0.966	146
	STAR	0.969	0.984	0.953	0.995	117793	110832	0.941	208
	TopHat2	0.720	0.974	0.718	0.999	114134	104970	0.920	20055
	Subread	0.928	0.987	0.901	0.998	111156	106542	0.958	2394
	MapSplice2	0.994	0.979	0.973	0.997	110676	106594	0.963	6032
	HISAT2	0.871	0.981	0.854	0.997	107932	104315	0.966	464
SimRead_251	DART	0.997	0.988	0.971	0.989	115680	111487	0.964	264
	STAR	0.939	0.982	0.921	0.995	118922	112132	0.943	359
	TopHat2	0.606	0.973	0.601	0.999	117547	107358	0.913	26523
	Subread	0.893	0.983	0.863	0.997	114634	109503	0.955	4170
	MapSplice2	0.998	0.967	0.964	0.997	111967	107395	0.959	7920
	HISAT2	0.829	0.978	0.811	0.995	108670	105086	0.967	635

Note: Each dataset contains around 40 million reads with different lengths. The simulation was based on known transcripts from the entire human genome.

Table 2. Performance comparison of DART and other selected aligners on the real datasets

Real datasets	Aligner	Sensitivity	SeqIdy	Reported SJ	True SJ	SJ accuracy	Runtime
SRR3351428 (100bp)	DART	0.975	0.999	236920	150260	0.634	244
	STAR	0.922	0.996	270788	152192	0.562	270
	TopHat2	0.844	0.998	217011	146077	0.673	22464
	Subread	0.858	0.998	221700	146518	0.661	3312
	MapSplice2	0.966	0.996	240918	149255	0.620	67446
	HISAT2	0.883	0.998	149592	129379	0.865	404
ERR1518881 (101bp)	DART	0.874	0.997	243515	154851	0.636	369
	STAR	0.841	0.987	259194	157026	0.606	371
	TopHat2	0.640	0.995	220662	150126	0.680	21185
	Subread	0.759	0.992	229369	151325	0.660	4008
	MapSplice2	0.893	0.988	221275	150496	0.680	15021
	HISAT2	0.756	0.993	162639	135460	0.833	480
SRR3439468 (151bp)	DART	0.930	0.996	197235	129148	0.655	481
	STAR	0.841	0.992	206157	129008	0.626	594
	TopHat2	NA	NA	NA	NA	NA	NA
	Subread	NA	NA	NA	NA	NA	NA
	MapSplice2	0.930	0.990	158581	113879	0.718	49320
	HISAT2	0.482	0.994	131892	105180	0.797	1306
SRR3439488 (151bp)	DART	0.899	0.995	142410	112562	0.790	427
	STAR	0.775	0.990	148672	113102	0.761	813
	TopHat2	NA	NA	NA	NA	NA	NA
	Subread	NA	NA	NA	NA	NA	NA
	MapSplice2	0.851	0.989	151771	107025	0.705	36240
	HISAT2	0.657	0.994	120311	100198	0.833	703

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

Table 3. Comparison of the peak RAM usage

Aligner	Memory usage (GB)
DART	12.0
STAR	30.0
TopHat2	4.5
Subread	10.0
MapSplice2	6.3
HISAT2	5.6

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

Dataset	Threads	runtime
SimRead_76	1	937
	2	464
	4	240
	8	126
	16	71

the threads are fully utilized all the time during the mapping process.

4 Conclusions

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.

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 *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.

Conflict of Interest: none declared.

References

Choi,J.H. *et al.* (2005) GAME: A simple and efficient whole genome alignment method using maximal exact match filtering. *Comput. Biol. Chem.*, **29**, 244–253.

Conesa,A. *et al.* (2016) A survey of best practices for RNA-seq data analysis. *Genome Biol.*, **17**, 13.

De Bona,F. *et al.* (2008) Optimal spliced alignments of short sequence reads. *Bioinformatics*, **24**, I174–I180.

Dobin,A. *et al.* (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**, 15–21.

Engstrom,P.G. *et al.* (2013) Systematic evaluation of spliced alignment programs for RNA-seq data. *Nat. Methods*, **10**, 1185+.

Fu,X. *et al.* (2009) Estimating accuracy of RNA-Seq and microarrays with proteomics. *BMC Genomics*, **10**, 161.

Garber,M. *et al.* (2011) Computational methods for transcriptome annotation and quantification using RNA-seq. *Nat. Methods*, **8**, 469–477.

Grant,G.R. *et al.* (2011) Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM). *Bioinformatics*, **27**, 2518–2528.

Griebel,T. *et al.* (2012) Modelling and simulating generic RNA-Seq experiments with the flux simulator. *Nucleic Acids Res.*, **40**, 10073–10083.

Harrow,J. *et al.* (2012) GENCODE: The reference human genome annotation for The ENCODE Project. *Genome Res.*, **22**, 1760–1774.

Hu,J. *et al.* (2012) OSA: a fast and accurate alignment tool for RNA-Seq. *Bioinformatics*, **28**, 1933–1934.

Jean,G. *et al.* (2010) RNA-Seq read alignments with PALMapper. *Curr. Protoc. Bioinformatics*, **32**, 11.6:11.6.1–11.6.37.

Kim,D. *et al.* (2015) HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods*, **12**, 357–U121.

Kim,D. *et al.* (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol.*, **14**, R36.

Leinonen,R. *et al.* (2011) The sequence read archive. *Nucleic Acids Res.*, **39**, D19–D21.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Li,H. *et al.* (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.

Li,H. and Homer,N. (2010) A survey of sequence alignment algorithms for next-generation sequencing. *Brief. Bioinf.*, **11**, 473–483.

Li,H. *et al.* (2008) Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res.*, **18**, 1851–1858.

Liao,Y. *et al.* (2013) The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Res.*, **41**, e108.

Lin,H.N. and Hsu,W.L. (2017) Kart: a divide-and-conquer algorithm for NGS read alignment. *Bioinformatics*, **33**, 2281–2287.

Liu,Y.C. and Schmidt,B. (2012) Long read alignment based on maximal exact match seeds. *Bioinformatics*, **28**, I318–I324.

Marco-Sola,S. *et al*. (2012) The GEM mapper: fast, accurate and versatile alignment by filtration. *Nat. Methods*, **9**, 1185–U1176.

Mortazavi,A. *et al*. (2008) Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nat. Methods*, **5**, 621–628.

Ryan,M.C. *et al*. (2012) SpliceSeq: a resource for analysis and visualization of RNA-Seq data on alternative splicing and its functional impacts. *Bioinformatics*, **28**, 2385–2387.

Sirbu,A. *et al*. (2012) RNA-seq vs dual- and single-channel microarray data: sensitivity analysis for differential expression and clustering. *Plos One*, **7**, e50986.

Trapnell,C. *et al*. (2009) TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, **25**, 1105–1111.

Wang,K. *et al*. (2010) MapSplice: Accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Res*., **38**, e178.

Wheeler,M.B. (1994) A block-sorting lossless data compression algorithm. *SRC Res. Rep*.

Wu,T.D. and Nacu,S. (2010) Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, **26**, 873–881.

Zhao,S. *et al*. (2014) Comparison of RNA-Seq and microarray in transcriptome profiling of activated T cells. *Plos One*, **9**, e78644.