Bioinformatics, 33(15), 2017, 2281–2287
doi: 10.1093/bioinformatics/btx189
Advance Access Publication Date: 4 April 2017
Original Paper

OXFORD

Sequence analysis

Kart: a divide-and-conquer algorithm for NGS read alignment

Hsin-Nan Lin and Wen-Lian Hsu*

Institute of Information Science, Academia Sinica, Taipei, Taiwan

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on November 8, 2016; revised on March 7, 2017; editorial decision March 27, 2017; accepted on April 5, 2017

Abstract

Motivation: Next-generation sequencing (NGS) provides a great opportunity to investigate genome-wide variation at nucleotide resolution. Due to the huge amount of data, NGS applications require very fast and accurate alignment algorithms. Most existing algorithms for read mapping basically adopt seed-and-extend strategy, which is sequential in nature and takes much longer time on longer reads.

Results: We develop a divide-and-conquer algorithm, called Kart, which can process long reads as fast as short reads by dividing a read into small fragments that can be aligned independently. Our experiment result indicates that the average size of fragments requiring the more time-consuming gapped alignment is around 20 bp regardless of the original read length. Furthermore, it can tolerate much higher error rates. The experiments show that Kart spends much less time on longer reads than other aligners and still produce reliable alignments even when the error rate is as high as 15%.

Availability and Implementation: Kart is available at https://github.com/hsinnan75/Kart/.

Contact: hsu@iis.sinica.edu.tw

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Next-generation sequencing (NGS) allows biologists to investigate genome-wide variation at nucleotide resolution. It has contributed to numerous ground-breaking discoveries and become a very popular technique for sequencing DNA and characterizing genetic variations in populations. Since new sequencing technologies can produce reads on the order of million/billion base-pairs in a single day, many NGS applications require very fast alignment algorithms. The traditional sequence alignment approaches, like BLAST (Altschul *et al.*, 1990) or BLAT (Kent, 2002), are unable to deal with the huge amount of short reads efficiently. Consequently, many aligners for NGS short reads have been developed in recent years. They can be classified into two categories according to their indexing methods: hash tables and suffix array/BWT. A hash table based aligner uses all the subsequences of k-mers to obtain the occurrence locations. In contrast, a suffix array/BWT based aligner finds the maximal exact matches (MEM) between the read sequence and the reference genome. Each category of read aligners has its own merits and deficiencies. However, suffix array/BWT based aligners are more popular due to the efficiency of memory consumption.

Aligners based on hash tables include CloudBurst (Schatz, 2009), Eland (proprietary), MAQ (Li *et al.*, 2008a), RMAP (Smith *et al.*, 2008), SeqMap (Jiang and Wong, 2008), SHRiMP (Rumble *et al.*, 2009), ZOOM (Lin *et al.*, 2008), BFAST (Homer *et al.*, 2009), NovoAlign (proprietary), SSAHA (Ning *et al.*, 2001), and SOAPv1 (Li *et al.*, 2008b). Most hash table based aligners essentially follow the same seed-and-extend strategy (Li and Homer, 2010). A representative algorithm of this strategy is BLAST. BLAST keeps the occurrence locations of each k-mer of the database sequences in a hash table and then uses the given query's k-mers to scan and find exact matches by looking up the hash table. An exact match is used as a seed to extend the alignment using Smith-Waterman algorithm between the query and the reference sequence.

Aligners based on suffix array or using Burrorws-Wheeler transform (BWT) (Wheeler, 1994) include Bowtie (Langmead and Salzberg, 2012; Langmead *et al.*, 2009), BWA (Li and Durbin, 2009), BWA-SW (Li and Durbin, 2010), BWA-MEM (Heng Li), SOAPv2 (Li *et al.*, 2009), CUSHAW (Liu *et al.*, 2012), Subread (Liao *et al.*, 2013), HISAT/HISAT2 (Kim *et al.*, 2015), HPG-aligner (Tarraga *et al.*, 2014)and segemehl (Hoffmann *et al.*, 2009). Most aligners in this category rely on a suffix array to identify the maximal exact matches (called MEMs) and then build alignments based on the exact matches, which is also similar to the seed-and-extend methodology. One exception is the Subread aligner, which adopts a seed-and-vote step to determine the mapped genomic location with multiple seeds from a read sequence. The major advantage of using suffix arrays is that repetitive subsequences need to be aligned only once because they are collapsed onto a single path (Li and Homer, 2010).

Though current short read aligners provide solutions for mapping the massive amount of read sequences produced by NGS technologies, some are not fast enough and some are not accurate enough. Moreover, the third generation sequencing technologies raise further challenges for data analysis, namely, extremely long read sequences and much higher error rate. For example, the PacBio RS II system can generate reads in the length of 5500–8500 bp on average, but the single-read accuracy is only about 87%. Most short read aligners have difficulty in processing those read sequences.

Keeping in mind all these challenges, we developed an alignment algorithm, called Kart, which uses both BWT array and hash table. Kart adopts a divide-and-conquer strategy, which separates a read into regions that are easy to align and regions that require gapped alignment, and align each region independently to compose the final alignment. In our experiments, the average size of fragments requiring gapped alignment is around 20 regardless of the original read length. The experiments on synthetic datasets show that Kart spends much less time on longer reads (150–7000 bp) than most aligners do, and still produces reliable alignments when the error rate is as high as 15%. The experiments on real datasets further demonstrate that Kart can handle reads with poor sequencing quality.

2 Materials and methods

2.1 Overview of our algorithms

Most suffix/BWT array based aligners, which follow the canonical seed-and-extend methodology, initiate an alignment with an MEM (seed) and extend the alignment with different dynamic programming strategies. Therefore, the performance of an aligner is greatly affected by the algorithms for seed exploration and the strategies for handling inexact matches. These aligners are sequential in nature. We adopted a divide-and-conquer strategy to reduce the time-consuming gapped alignment step using dynamic programming, which is suitable for mapping highly similar fragment sequences (each read is essentially a copy of a specific genome fragment except for a small percentage of sequencing errors).

2.2 Simple pairs and normal pairs

Since un-gapped (without indels) alignment is much faster than gapped alignment, for each mapped candidate region in the reference genome, we separate the given read sequence and their candidate regions into two groups: simple region pairs (abbreviated as *simple pairs*) and normal region pairs (*normal pairs*), where all simple pairs have perfect alignment (exact matches), and normal pairs require un-gapped/gapped alignment. Once the simple and normal pairs are identified, they can be processed and aligned independently and the final mapping result for a candidate region is simply the concatenation of the alignment of each simple and normal pair.

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 remainder of the paper. 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 a given BWT array of length l is defined as a maximal exact match between $R[i_1, i_2]$ (called the *read block*) and $G[j_1, j_2]$ (called the *genome block*), where $i_2 - i_1 = j_2 - j_1 = l - 1$ and is denoted by a 4-tuple (i_1, i_2, j_1, j_2). We use $\Delta\text{Pos} = (j_1 - i_1)$ to represent the position difference between the read and genome block.

2.3 Finding all LMEMs for the given read sequence

Since an LMEM represents an identical fragment pair between R and G, it is considered as a *simple* pair in this study. Kart finds all LMEMs via traversing a BWT array. An LMEM exploration starts from $R[i_1]$ and stops at $R[i_2]$ if the exact match extension reaches a mismatch at $R[i_2+1]$. In such cases, the next LMEM exploration will skip $R[i_2+1]$ and starts from $R[i_2+2]$ because $R[i_2+1]$ is very likely a sequencing error or sequence variation. Kart only considers an LMEM (i_1, i_2, j_1, j_2) as a qualified simple pair if its length is no less than a predefined threshold k and its occurrence is less than 50. The value k is typically between 10 and 16, and is determined based on the size of the reference genome. Intuitively, a short LMEM ($< k$ bp) might contain erroneous bases and would less likely include the true coordinate. A larger genome needs a larger k for a compromise between specificity and sensitivity.

Figure 1 shows the algorithm of the LMEM exploration procedure. The function *BWT_search* is a general BWT traversal method which takes a read sequence as the input and returns desirable LMEMs as well as their occurrences in the reference genome. If there are no sequencing errors or variations, there should be only one LMEM which covers the whole read sequence (i.e. $LMEM.len = |R|$). However, in reality, sequencing errors and variations happen a lot and they break a read into several LMEMs with variable lengths. Kart considers all qualified LMEMs as simple pairs, and identify normal pairs according to the distribution of simple pairs to create one or more candidate alignments.

```
Procedure of LMEM exploration:
    start ← 1
    stop ← |R|
    while (start ≤ stop)
        if R[start] ∉ {A,C,G,T} then
            start ← start+1
        else
            LMEM ← LMEM_Search(R, start, stop)
            if LMEM.len ≥ k then
                for each occurrence p of LMEM do
                    mem.i1 ← start
                    mem.i2 ← start + LMEM.len – 1
                    mem.j1 ← p
                    mem.j2 ← p+ LMEM.len – 1
                    mem.l ← LMEM.len
                    A ← A ∪ mem   // A is the set of all LMEMs
                start ← start + LMEM.len
            else
                start ← start + 1

    end
```

Fig. 1. The algorithm to explore all LMEMs with length ≥ k. *BWT_search* is the function to search for the occurrences of the maximal exact match for R[start, stop] on the given BWT array. It returns desirable LMEMs as well as their occurrences on the reference genome

2.4 Identifying simple pairs and normal pairs

Since the same read can be mapped to multiple genome regions if it originates from a repetitive sequence, these simple pairs can also spread all over the genome. A candidate alignment is defined as an alignment between the read and a specific region on the reference genome. To identify all candidate alignments for the read, we cluster all adjacent simple pairs whose ΔPos differences are smaller than a predefined threshold g (the default value is 5). Therefore, neighboring simple pairs whose ΔPos differences $< g$ are clustered together to form a candidate alignment. Note that two simple pairs in a candidate alignment could overlap due to tandem repeats, sequence variations or overlapping LMEMs (when doing 8-LMEMs, as explained in Section 2.5). If two simple pairs overlap in a candidate alignment, say, in the genome portion (the same goes for the read portion), we chop off the overlapped portion in the genome and its corresponding read portion from the shorter pair to ensure that all simple pairs are non-overlapping. Figure 2 illustrates an example of two overlapped simple pairs, in which simple pair A overlaps with simple pair B due to sequence variation at that corresponding region. Kart removes the overlap by shrinking the shorter simple pair, say A. In this way, any two simple pairs in the same candidate alignment will not share any nucleotide.

We then create normal pairs filling the gaps between adjacent simple pairs to make a complete alignment as follows. Suppose two neighboring simple pairs are (i_{2q-1}, i_{2q}, j_{2q-1}, j_{2q}) and (i_{2q+1}, i_{2q+2}, j_{2q+1}, j_{2q+2}) respectively, and they must have $i_{2q+1} - i_{2q} > 1$ or $j_{2q+1} - j_{2q} > 1$, Kart inserts a normal pair to fill the gaps between the two regions. In such cases, a normal pair (i_r, i_{r+1}, j_r, j_{r+1}) is inserted where $i_r - i_{2q} = i_{2q+1} - i_{r+1} = 1$ if $i_{2q+1} - i_{2q} > 1$, otherwise let $i_r = i_{r+1} = -1$ (i.e., a null base). Likewise, $j_r - j_{2q} = j_{2q+1} - j_{r+1} = 1$ if $j_{2q+1} - j_{2q} > 1$, otherwise let $j_r = j_{r+1} = -1$. On the other hand, if the first (or the last) simple region in a candidate alignment does not cover the first (or the last) nucleotide of the read sequence, a normal pair would also be created to fill the gap. Figure 3 gives an example illustrating the concept of simple and normal pairs. In this example, the read sequence contains three substitution errors as well as an insertion error of size two. After the LMEM exploration, we can identify four simple pairs (labelled *A*, *B*, *C* and *D*). However, these four simple pairs do not cover the whole read sequence. Thus, we check every adjacent simple pairs and generate the corresponding normal pair according to the gaps in between: we generate the normal pair (11, 11, 321, 321) between simple pairs *A* and *B*; likewise, we generate the normal pair (23, 24, -1, -1) between simple pairs *B* and *C*, and the normal pair (49, 51, 357, 359) between simple pairs *C* and *D*. Thus, all of these simple pairs and normal pairs together form a complete alignment candidate.

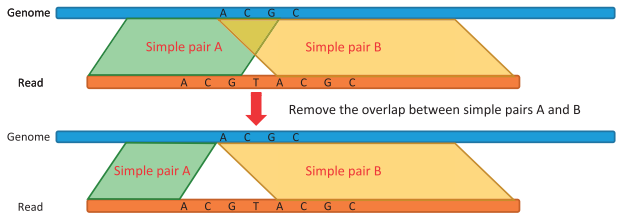

Fig. 2. Simple pair A overlaps with simple pair B. Kart removes the overlap by shrinking the size of the smaller simple pair

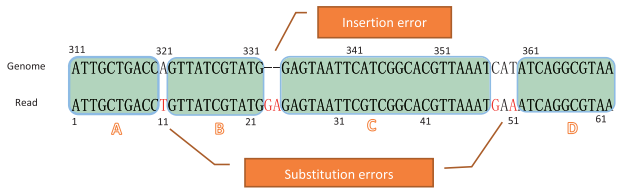

Fig. 3. Simple pairs and normal pairs. A read sequence can be decomposed into different parts according to the alignment with the genome sequence. A simple pair represents a pair of identical sequence fragments; a normal pair represents a pair of sequence fragments which contains some sequence variations in the alignment

2.5 Four types of normal pairs

Simple pairs are formed from LMEMs with perfect matches. They partition the read sequence into interlaced simple and normal region pairs, which can be independently aligned and the final alignment is simply the concatenation of the alignment of each simple and normal pair. A closer look at the normal pairs indicates that a substantial portion of normal pairs do not require gapped alignment either. When both the read block and the genome block of a normal pair are more than 30 bp long, we perform a second round of sequence partition operation to further divide it and reduce the portion that needs gapped alignment. This time, we look for LMEMs of size ≥ 8 bp within such normal pair. Since the size of such normal pair is much smaller than the whole genome, 8-mer index (from a hash table constructed on the fly) is used to identify matched 8-mer seeds between the read block and the genome block. These seeds were extended to LMEMs, referred to as 8-LMEMs. These 8-LMEMs could further separate a longer normal pair into shorter ones.

When processing PacBio reads, if a normal pair (without 8-LMEMs) at the end of a read has length more than 5000, then its corresponding read block is simply clipped from further consideration, and we only perform a local alignment for that read to avoid making an alignment in a poor quality region. When processing Illumina reads, we found that if the read block and genome block of a normal pair have equal size, then it is very likely the normal pair only contains substitution errors and the un-gapped alignment gives rise to the best alignment for such pair; 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. Moreover, Illumina reads could contain adaptor sequences or become chimera with a tiny probability. We check the sequence identity of the normal block at both ends of a read and remove the one whose sequence identity is $< 50\%$. In such cases, Kart clips the corresponding read block and report a local alignment instead. Summarizing the above discussion, we divided the normal pairs into the following four types:

1. A normal pair is a *NP-clip* if (1) it is at the ends of a PacBio read and its length is more than 5000 or (2) it is at the ends of an Illumina read and its sequence identity is $< 50\%$.
2. A normal pair is a *NP-gap free* if its read block and genome block have equal size, and its number of mismatches in a linear scan is less than 20% of the block size.
3. A normal pair is a *NP-indel* if one of the fragment (either its read block or genome block) is an empty string and the other contains at least one nucleotide.
4. The remaining normal pairs are referred to as *NP-NW*s, which require Needleman-Wunsch algorithm for gapped alignment.

Disregarding those NP-clips, one can see from Table 3 that those NP-NWs requiring gapped alignment were sufficiently separated so

that their average lengths are around 20 for different datasets, which can be processed much faster than the original read.

2.6 Paired-end reads mapping

Paired-end reads are two reads that are physically associated within a certain distance from each other. They can help reduce mapping ambiguity and increase mapping reliability. Kart supports paired-end reads mapping as follows. To align paired-end reads, Kart finds the candidate alignments for each read separately and then compare the two groups of mapping results to see if there is a pair of alignments one from each group that satisfies the paired-end distance constraints. If there is no such pair, it implies that the paired-end reads contain more sequencing errors such that at least one of them is not mapped properly. In such cases, Kart will initiate a rescue procedure trying to find a proper pair of alignments based on the candidate alignments of the opposite read. The implementation of the rescue procedure is described in detail below.

Suppose G_1 and G_2 are the two groups of candidate alignments corresponding to paired-end reads R_1 and R_2. Let $G_1 = \{m_1, m_2, \ldots, m_p\}$ and $G_2 = \{n_1, n_2, \ldots, n_q\}$ where $m_1, m_2, \ldots$, and m_p represent candidate alignments of R_1; n_1, n_2, … and n_q represent candidate alignments of R_2. For each alignment m in G_1 with the coordinate c, Kart tries to map R_2 onto the downstream region of c according to the mapping procedure described above, but the LMEM length threshold is set to 10 to increase the sensitivity of the mapping. In doing so, Kart is able to compromise the noises of R_2 and identify the best alignment n' in the downstream region of R_1's alignment. Kart repeats the same procedure using each alignment n in G_2 to find the best alignment m' for R_1 in the upstream region of n. At this moment, both the pair of m_i and n' and m' and n_j meet the paired-end constraint, and we select the pair with the highest sequence identity of the alignments.

2.7 Mapping quality score

MAQ (Li *et al.*, 2008a,b) 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 optimal and sub-optimal alignments. We adopted a scoring scheme to estimate the mapping quality.

The mapping quality score MAPQ is estimated with the formula: $\mathrm{MAPQ} = 30 \times [1 - (S_1 - S_2)\ /S_1] * \log(S_1)$, where S_1 is the optimal alignment score and S_2 is the sub-optimal alignment score. MAPQ is limited to be between 0 and 60. If there are multiple alignments with the optimal score (i.e., repeats), MAPQ is 0 and Kart reports the first best alignment.

2.8 Summary of our algorithms

Given a read sequence R, Kart identifies all LMEMs with variable lengths along the read sequence by the BWT search against the reference sequences. Each LMEM is turned into a simple pair or more if it appears multiple times in the reference. Adjacent simple pairs are then clustered according to their ΔPos. After removing overlaps between adjacent simple pairs, Kart fills in the gaps between simple pairs with normal pairs to make each cluster a complete candidate alignment. When both the read block and the genome block of a normal pair are more than 30 bp long, we perform a second round of sequence partitioning to further divide it and reduce the portion that needs gapped alignment. The final read alignment is the concatenation of simple/normal pairs in the same candidate alignment. Finally, Kart reports the alignment with the highest alignment score or paired alignments for paired-end reads.

3 Results

3.1 Implementation and experiment design

Kart was developed under Linux 64-bit environment and implemented with standard C/C++. It supports multi-thread to take advantage of multi-core computers. Kart reads a BWT-based indexing file and a read library (single-end or paired-end reads) in FASTA/FASTQ format as input and it reports read alignments in the SAM (Sequence Alignment/Map) format (Li *et al.*, 2009). For the sake of user-friendliness, we have minimized parameter settings and make Kart work on all kinds of read libraries.

It is difficult to estimate the correctness of read alignments using real datasets since the actual chromosomal coordinate of each read sequence is unknown. Therefore, we simulated read libraries to estimate the performance of read aligners. Here, we simulated read libraries of the human genome (Hg19) using the wgsim program (https://github.com/lh3/wgsim) to produce the synthetic datasets. All the Illumina-like simulated read libraries were generated with default setting of the wgsim. It altered the genome sequences with 0.1% of mutation rate (15% of which are INDELs and 85% are SNPs) to simulate polymorphisms. In turn, wgsim generated synthetic reads with 2% of substitution sequencing errors.

We also simulated PacBio-like reads using the wgsim program with 13% of mutation rate (all INDEL polymorphisms) and an additional 2% of substitution sequencing errors for Hg19. It can be expected that reads from new sequencing technologies are getting longer (Li and Durbin, 2010). For example, the latest Illumina MiSeq System can generate reads of 300 bp long. Therefore, in this study, we simulated ten million pair-end reads of 100 bp, 150 bp and 300 bp long for Illumina-like data and one million single-end reads of 7000 bp long for PacBio-like data. For the convenience of describing these synthetic libraries, *Hg19_L150_E02* represents the Illumina-like library with 150 bp read length and 2% of error rate of Hg19. *Hg19_L7000_E15* represents the PacBio-like library with 7000 bp read length and 15% of error rate (13% of indels and 2% of substitutions). To assess the performance on real data, we randomly downloaded four datasets from NCBI SRA, Illumina and PacBio web sites, which are *SRR622458*, *SRR826460*, *SRR826471* and *M130929* (http://datasets.pacb.com/2014/Human54x/fasta.html). The first three datasets were produced by Illumina sequencers, and the last one was by PacBio sequencer. All the datasets were from human genome samples.

We used precision, recall (also referred to as accuracy in this study) and running time to estimate the performance of read alignments on simulated data. A read is considered a true positive (TP) if its mapping coordinate is within a distance of 30 bp to the original coordinate. Given a library with N reads and n out of N are mapped with at least one alignment, the precision and recall are defined as the following percentages:

$$precision = \#\ \text{of}\ TPs/n\ \times 100\%;$$

$$recall = \#\ of\ TPs/N\ \times 100\%.$$

Therefore, if every input read is mapped, precision is equal to recall. To avoid estimation bias due to multiple hits (i.e., ambiguous mapping), we only evaluated the first alignment for each read. For real data, we measure the sensitivity (also referred to as mapping rate)

and runtime, where sensitivity is the percentage of reads with at least one alignment relative to all input reads, namely,

$$sensitivity = n/N \times 100\%.$$

Since the actual coordinates of real data are unknown, we evaluate the mapping quality by calculating the number of identical base pairs in an alignment. The reason is that the best alignment will likely have the maximal number of identical base pairs.

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. We compared Kart with several existing read aligners: BWA-MEM, Bowtie2, Cushaw3, HISAT2, HPG-aligner, Subread, LAST (Frith *et al.*, 2010), Minimap (Li, 2016)and BLASR (Chaisson and Tesler, 2012) (the last three are for PacBio data only). The other aligners which were not considered in this study are those which do not support multi-threading or do not accept the format of test data (such as Gassst, Ssaha2 and NovoAlign), plus those which could not be run properly in our system or took much more time to process the test data (such as GEM, hobbes and razers3).

The selected aligners represent state-of-the-art NGS short read mappers and are widely used in NGS data analysis. We tried to use the default parameter settings to test each aligner unless the performance was not satisfactory. We also forced each aligner to only report the best alignment or a random best if there were multiple hits by setting the parameters. All aligners were run with 16 threads to speed up the alignment procedure. The arguments as well as version number of each aligner are summarized in the Supplementary Material.

3.2 Evaluation on Illumina-like simulated datasets

Table 1 shows the evaluation result for the selected aligners on the Illumina-like libraries of Hg19. From Table 1, we can see that most selected aligners produced comparable alignments on read libraries with different read lengths, where their precisions and recalls ranged from 97 to 99%. In fact, most of the incorrect alignments were due to the ambiguity of repetitive regions. In most cases, the alignment accuracies were improved when the read length became longer. For example, the alignment accuracies of Kart on *Hg19_L100_E02, Hg19_L150_E02* and *Hg19_L300_E02* were 97.8%, 98.5% and 99.1%, respectively; and those of BWA-MEM were 98.6%, 98.9% and 99.2%, respectively. Bowtie and HISAT2 were less sensitive when aligning longer reads. In particular, HISAT2's recall on *Hg19_L300_E02* was 53.6%.

Table 1. Illumina and PacBio-like simulated data. Ten million paired-end reads of 100 bp, 150 bp and 300 bp and one million single-end reads of 7000 bp were simulated from human genome (Hg19) with wgsim simulator

Synthetic datasets	Aligner	Precision	Recall	Runtime
Hg19_L100_E02	Kart	97.8	97.8	53
	Bowtie2	96.3	95.8	149
	BWA-MEM	98.6	98.6	403
	Cushaw3	98.2	98.2	1412
	HPG-Aligner	97.7	97.5	146
	HISAT2	95.3	92.7	78
	Subread	98.5	93.4	353
Hg19_L150_E02	Kart	98.4	98.4	66
	Bowtie2	96.2	96.2	266
	BWA-MEM	98.9	98.9	581
	Cushaw3	98.6	98.6	1278
	HPG-Aligner	98.5	98.5	315
	HISAT2	92.3	89.4	91
	Subread	98.0	96.9	474
Hg19_L300_E02	Kart	99.0	99.0	113
	Bowtie2	96.1	96.1	718
	BWA-MEM	99.2	99.2	1096
	Cushaw3	99.1	99.1	3085
	HPG-Aligner	99.1	99.1	317
	HISAT2	70.5	54.6	155
	Subread	98.8	98.8	774
Hg19_L7000_E15	Kart	99.6	99.6	733
	BWA-MEM	99.8	99.8	4614
	LAST	99.9	99.4	78432
	Minimap	83.4	83.4	288
	BLASR	99.8	99.8	9185

In terms of runtime, it can be seen that Kart was the fastest aligner among all considered algorithms. In this analysis, the runtime of Kart on the three simulated datasets was 53, 66 and 113 seconds, respectively. Thus, our divide-and-conquer strategy provides a much faster solution for NGS read mapping, especially for longer reads.

3.3 Evaluation on PacBio-like simulated datasets

Table 1 also shows the evaluation result on one million PacBio-like reads. Because PacBio-like reads contain much more insertion/deletion errors, a read is considered to be correctly aligned if the mapping coordinate is within a distance of 100 bp to the original coordinate. In this evaluation only Kart, BWA-MEM, LAST, Mimimap and BLASR were compared because the others were not designed for aligning long reads in PacBio data. In Table 1, Kart, BWA-MEM, LAST and BLASR yielded similar recall on the PacBio simulated data. It shows that these aligners were capable of dealing with very long reads with high error rate. However, Kart was much faster: the runtime of Kart was 733 seconds, and that of BWA-MEM, LAST and BLASR was 4614, 78 432 and 9185, respectively. Though Minimap took the least amount of time (288 seconds), its mapping accuracy was only 83.4%. It is noteworthy that Minimap is not a regular read aligner since it does not generate alignments. It can only identify long approximate matches quickly. Therefore, we did not evaluate Minimap on real PacBio dataset.

3. 4 Evaluation on real datasets

In additional to synthetic datasets, we also downloaded four real datasets of whole human genome from NCBI SRA, Illumina and PacBio web sites, which are

1. *SRR622458* (40 million 101 bp paired-end Illumina reads).
2. *SRR826460* (40 million150 bp paired-end Illumina reads).
3. *SRR826471* (34 million 250 bp paired-end Illumina reads).
4. *M130929* (1.2 million 7118 bp single-end PacBio reads).

Table 2 summarizes the evaluation result on these datasets. In this evaluation, we use the sensitivity and the average number of identical bases to estimate the quality of alignments. It can be seen that Kart spent the least amount of time on mapping the reads in these real datasets. Kart was several times faster than alternative aligners. Kart also produced the largest number of identical base pairs on most of the datasets (sensitivity × average identical bases).Take *SRR622458* for example. Kart produced alignments with 98.6% of sensitivity and 99 of identical bases on average. BWA-MEM, Bowtie2 and Cushaw3 produced comparable alignments with Kart, but they spent more time on the alignments. Note that HPG-aligner failed to finish all reads in *SRR622458* due to unknown reasons. Some aligners left more reads unmapped. HISAT2 only produced 86.0%, 91.9% and 43.9% of sensitivity on the three Illumina datasets.

Table 2. Experiment result on the four real datasets with different read lengths

Real datasets	Aligner	Sensitivity	Identical base pairs	MEM (Gb)	Runtime
SRR622458	Kart	98.6	99	12	158
Illumina-101 bp	Bowtie2	97.4	99	4.5	458
	BWA-MEM	98.8	97	8.5	1157
	Cushaw3	99.1	98	4.8	9063
	HPG-Aligner	NA	NA	31.2	NA
	HISAT2	86.0	99	5.5	298
	Subread	91.2	97	18.4	1362
SRR826460	Kart	99.3	149	12	186
Illumina-150 bp	Bowtie2	98.4	149	4.5	769
	BWA-MEM	99.3	147	8.5	1374
	Cushaw3	99.3	148	4.8	10736
	HPG-Aligner	98.3	147	31.2	1204
	HISAT2	91.9	149	5.5	371
	Subread	97.5	147	18.4	2694
SRR826471	Kart	98.6	237	12	395
Illumina-250 bp	Bowtie2	94.7	237	4.5	1729
	BWA-MEM	98.6	220	8.5	3027
	Cushaw3	98.4	232	4.6	37689
	HPG-Aligner	NA	NA	27.9	NA
	HISAT2	43.9	247	5.5	461
	Subread	NA	NA	18.4	NA
M130929	Kart	100.0	5152	13	1811
PacBio-7118 bp	BWA-MEM	90.7	2953	9	7338
	LAST	97.2	5022	15	31295
	BLASR	97.8	5389	28.9	18682

For the experiment result on the PacBio dataset *M130929*, Kart and BLASR produced comparable alignments on the 1.2 million PacBio reads. However, BLASR spent more time than Kart. BWA-MEM ran faster than BLASR, but its sensitivity and average number of identical pairs are not as good as those on Illumina datasets. LAST's speed ranked last, but it produced comparable alignments with Kart and BLASR.

We further compared memory usage of each selected aligner. Though some aligners allow users to set the maximum memory usage for read mapping, we did not set any limit and let the aligner take as much memory as it needs to speed up the mapping process. In Table 2, we found that each aligner consumed similar amount of memory on different datasets. BWA-MEM, Bowtie2, Cushaw3 and HISAT2 required less memory (<10 GB). Kart and Subread required 12 GB and 18 GB, respectively, and HPG-aligner and BLASR required around 30 GB.

3.5 Efficiency of Kart's divide-and-conquer strategy

The performance analysis on simulated datasets shows that Kart is an efficient algorithm for NGS read mapping. We adopt a divide-and-conquer strategy to separate a read sequence into simple pairs and normal pairs, and align each pair independently. A simple pair has a perfect match, but a normal pair requires more time to find its best alignment. Hence, if the fraction of the normal pairs is smaller, and the fragment sizes are shorter, the read can be mapped faster.

To demonstrate the efficiency of Kart's divide-and-conquer strategy with different read lengths, we analyzed the average sizes of simple and normal pairs on the four real datasets. Table 3 shows the average sizes of all fragment pairs after two rounds of sequence partition, namely, LMEM-seed, 8-LMEM-seed, NP-gap free, NP-indels and NP-NW. Note that, fragment pairs in the first four groups do not require gapped alignment, only those in the last group do. Take SRR622458 as an example, the average size of LMEM-seed is 73 bp, and 96.5% nucleotides belong to this group. When applying the normal pair partitioning with 8-mers, we could identity 8-LMEM-seeds with 11.4 bp on average. The most time consuming fragment pairs, the NP-NWs, are 17.5 bp on average, and only 1.9% nucleotides fall into this group. SRR826460 shows similar result with SRR622458, though SRR826471 has much higher ratio in the group of NP-NW, which suggests that Illumina yields higher error rates on 250 bp long reads. For the real PacBio dataset, it is observed that the average size of LMEM-seed is 21.3 bp, and only 13.7% of nucleotides fall into this group; however, the second round of seeding could identify 8-LMEM-seeds with 12.4 bp on average and 39.7% of nucleotides go to this group; It is noteworthy that the most time consuming group, the NP-NW, only needs to perform DP on 21.3 bp long fragment pairs on average, with 44.3% of all nucleotides.

Table 3. The average sizes of simple and normal pairs after two rounds of sequence partition on Illumina datasets (those NP-clips were not included in the percentage calculation)

Dataset	LMEM-seed	8-LMEM-seed	NP-gap free	NP-indels	NP-NW
SRR622458	73.0 (96.5%)	11.4 (0.9%)	3.9 (0.7%)	1.8 (0%)	17.5 (1.9%)
SRR826460	112.7 (97.9%)	13.7 (0.5%)	4.5 (0.7%)	1.9 (0%)	19.5 (0.9%)
SRR826471	104.2 (84.9%)	12.4 (3.8%)	7.5 (2.5%)	1.9 (0%)	22.8 (8.8%)
M130929	21.3 (13.7%)	12.4 (39.7%)	10.8 (0.1%)	1.4 (2.6%)	21.3 (44%)

4 Conclusions

In this article, we present Kart, a new sequence aligner for sensitive, rapid and accurate mapping from NGS reads to a reference genome. We use a BWT array to produce an alignment consisting of simple pairs and normal pairs. Each simple pair represents an identical fragment pairs between the query sequence and the reference genome, and each normal pair represents a highly similar fragment pairs. We show that Kart's divide-and-conquer strategy can reduce the required dynamic programming process and save considerable amount of time, especially for mapping long read sequences. In our evaluation analysis on simulated reads and real data, Kart yields the best or comparable alignments and spends the least amount of time.

PacBio reads are usually difficult to map efficiently because of their extremely long sequences and high sequencing error rate. From the analysis results on simulated and real PacBio datasets, Kart not only can generate accurate alignments, but also spends much less time than others. With the improved sequencing technology, new high-throughput sequencers are likely to generate much longer reads with varied quality. Experiment results show that Kart is an appropriate aligner that can produce efficient and accurate alignments for reads with various lengths and quality.

In this study, we only considered sequencing errors, SNPs and small indels in short read sequences. Studies have shown that structural variations (such as inversions and translocations) are more difficult to detect than the above variations, and it is estimated that 13% of the human genome are defined as structural variants in the normal population (Sudmant *et al.*, 2015). We believe the divide-

and-conquer strategy of Kart could be used to develop a more advanced aligner for handling structural variations.

Acknowledgements

The authors would like to thank Dr Yen-Hua Huang, Ms Hannah Lin and Ms Sharon Chiu at National Yang-Ming University, Taiwan for discussion and advice at the development stage of this study. We appreciate the advice and information on NGS aligners from Dr Einar Andreas Rødland. In particular, we are very grateful for the read simulator wgsim and BWA developed by Dr Heng Li. The indexing and bwt search procedure of Kart were modified from BWA.

Funding

Bioinformatics Core Facility for Translational Medicine and Biotechnology Development, MOST105-2319-B-400-002.

Conflict of Interest: none declared.

References

Altschul,S.F. *et al*. (1990) Basic local alignment search tool. *J. Mol. Biol*., **215**, 403–410.

Chaisson,M.J. and Tesler,G. (2012) Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics*, **13**, 238.

Frith,M.C. *et al*. (2010) Incorporating sequence quality data into alignment improves DNA read mapping. *Nucleic Acids Res*., **38**, e100.

Hoffmann,S. *et al*. (2009) Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *Plos Comput. Biol*., **5**, e1000502.

Homer,N. *et al*. (2009) BFAST: an alignment tool for large scale genome resequencing. *Plos One*, **4**, A95–A106.

Jiang,H. and Wong,W.H. (2008) SeqMap: mapping massive amount of oligonucleotides to the genome. *Bioinformatics*, **24**, 2395–2396.

Kent,W.J. (2002) BLAT–the BLAST-like alignment tool. *Genome Res*., **12**, 656–664.

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

Langmead,B. and Salzberg,S.L. (2012) Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, **9**, 357–359.

Langmead,B. *et al*. (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*., **10**, R25.

Li,H. (2016) Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, **32**, 2103–2110.

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

Li,H. and Durbin,R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics*, **26**, 589–595.

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

Li,R.Q. *et al*. (2008b) SOAP: short oligonucleotide alignment program. *Bioinformatics*, **24**, 713–714.

Li,R.Q. *et al*. (2009) SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics*, **25**, 1966–1967.

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. *et al*. (2008) ZOOM! Zillions of oligos mapped. *Bioinformatics*, **24**, 2431–2437.

Liu,Y. *et al*. (2012) CUSHAW: a CUDA compatible short read aligner to large genomes based on the Burrows-Wheeler transform. *Bioinformatics*, **28**, 1830–1837.

Ning,Z. *et al*. (2001) SSAHA: a fast search method for large DNA databases. *Genome Res*., **11**, 1725–1729.

Rumble,S.M. *et al*. (2009) SHRiMP: accurate mapping of short color-space reads. *Plos Comput. Biol*., **5**, e1000386.

Schatz,M.C. (2009) CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics*, **25**, 1363–1369.

Smith,A.D. *et al*. (2008) Using quality scores and longer reads improves accuracy of Solexa read mapping. *BMC Bioinformatics*, **9**, 128.

Sudmant,P.H. *et al*. (2015) An integrated map of structural variation in 2,504 human genomes. *Nature*, **526**, 75+.

Tarraga,J. *et al*. (2014) Acceleration of short and long DNA read mapping without loss of accuracy using suffix array. *Bioinformatics*, **30**, 3396–3398.

Wheeler,M.B. *et al*. (1994) A block-sorting lossless data compression algorithm. *SRC Res. Rep*., **124**.