Article Navigation

rHAT: fast alignment of noisy long reads with regional hashing

Bo Liu, Dengfeng Guan, Mingxiang Teng, Yadong Wang Author Notes

Bioinformatics, Volume 32, Issue 11, 1 June 2016, Pages 1625–1631,
https://doi.org/10.1093/bioinformatics/btv662
Published: 14 November 2015 **Article history ▼**

Views ▼ Cite Permissions Share ▼

Abstract

Motivation: Single Molecule Real-Time (SMRT) sequencing has been widely applied in cutting-edge genomic studies. However, it is still an expensive task to align the noisy long SMRT reads to reference genome by state-of-the-art aligners, which is becoming a bottleneck in applications with SMRT sequencing. Novel approach is on demand for improving the efficiency and effectiveness of SMRT read alignment.

Results: We propose Regional Hashing-based Alignment Tool (rHAT), a seed-and-extension-based read alignment approach specifically designed for noisy long reads. rHAT indexes reference genome by regional hash table (RHT), a hash table-based index which describes the short tokens within local windows of reference genome. In the seeding phase, rHAT utilizes RHT for efficiently calculating the occurrences of short token matches between partial read and local genomic windows to find highly possible candidate sites. In the extension phase, a sparse dynamic programming-based heuristic approach is used for reducing the cost of aligning read to the candidate sites. By benchmarking on the real and simulated datasets from various prokaryote and eukaryote genomes, we demonstrated that rHAT can effectively align SMRT reads with outstanding throughput.

Availability and implementation: rHAT is implemented in C++; the source code is available at https://github.com/HIT-Bioinformatics/rHAT.

Contact:ydwang@hit.edu.cn

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

Issue Section: SEQUENCE ANALYSIS

1 Introduction

Single Molecule Real-Time (SMRT) sequencing has been widely applied since its emergence in 2009 (Eid *et al.*, 2009). With the superior read length and less systematic bias respect to next generation sequencing (NGS) (Roberts *et al.*, 2013), SMRT sequencing performed well in various cutting-edge genomics studies (Chaisson *et al.*, 2015; Huddleston *et al.*, 2014; Koren *et al.*, 2013). However, its higher sequencing error rate (Carneiro *et al.*, 2012; Chin *et al.*, 2013; Koren *et al.*, 2012) affects various critical steps of SMRT data analysis. One of them is read alignment.

Read alignment is to recover the likely genomic origins of reads by aligning them against reference genome. As one of the most fundamental and computing-intensive steps in genome re-sequencing studies (Langmead and Salzberg, 2012), efficient read alignment is on widely demand. However, due to the higher error rate, the speed of SMRT read alignment is not as fast as that of NGS reads (Chaisson and Tesler, 2012), which has become a bottleneck of SMRT data application.

Many efforts have been made to read alignment (Fonseca *et al.*, 2012), however, most of them are designed for NGS reads which are less adaptive to noisy SMRT reads. Previous studies (Chaisson and Tesler, 2012; Ono *et al.*, 2013) also indicated that long sequence alignment approaches (Kent, 2002; Kielbasa *et* al., 2012) can sensitively align SMRT reads, but at the expense of efficiency.

Three state-of-the-art SMRT read aligners are BWA-SW (Li and Durbin, 2010), BLASR (Chaisson and Tesler, 2012) and BWA-MEM (Li, 2013), with no exception on the basis of seed-and-extension. Mainly, they match partial reads (‘seeds’) to reference genome at first, and extend the matches (‘hits’) to compose full alignments. The major difference is in the way of seeding. BWA-SW aligns substrings of read to reference genome by suffix trie traversing, to find approximate matches with few positions as hits. BLASR and BWA-MEM use exact matches of short tokens between read and reference genome. BLASR formulated a probability model to depict the relationship between the error rate and the number of seeds. This model finds all the short token matches longer than a pre-defined threshold as hits. BWA-MEM finds all

Skip to Main Content

the maximal exact matches (MEMs) as hits, and also refers to the model of BLASR to tune its parameters for SMRT reads. Efforts have also been made to improve the efficiency of extension, i.e. BWA-SW and BWA-MEM utilized banded Smith-Waterman algorithm, and BLASR used a sparse dynamic programming (SDP)-based heuristic (Eppstein *et al.*, 1992).

One of the major bottlenecks of the state-of-the-art aligners is the efficiency of seeding. To be robust to the high error rate, all the three aligners densely employed the substrings starting from all the offsets of the read as seeds. However, for BLASR and BWA-MEM, due to the repetitiveness of genomes, there could be a great amount of hits generated by the dense short seeds along the whole read, which is expensive to merge and prioritize. For BWA-SW, the cost could be also huge to align many substrings of the noisy long read to reference genome. Moreover, as all the three aligners used Burrows-Wheeler Transformation (BWT) (Ferragina and Manzini, 2000) for indexing reference genome, there is additional cost to perform LF-mapping (Langmead *et al.*, 2009) to locate the precise genomic positions of the hits. Thus, for BWA-MEM and BLASR, the overhead of retrieving all the hits of short seeds is also non-neglectable.

Herein, we propose an efficient noisy long read alignment approach, Regional Hashing-based Alignment Tool (rHAT), for SMRT read alignment. rHAT aligns SMRT reads with specifically designed seed-and-extension strategy. In the seeding phase, rHAT selects a substring of the read as ‘seeding region’, and partitions reference genome into a series of windows, to find the windows having most k-mer matches with the seeding region as candidate sites for extension. This operation is implemented with small overhead through a hash table-based index, Regional Hash Table (RHT). In the extension phase, a SDP-based heuristic is designed for efficiently aligning the read to the candidate sites.

There are aligners (Ahmadi *et al.*, 2012; David *et al.*, 2011; Weese *et al.*, 2012; Yanovsky, 2014) based on q-gram filter (Rasmussen *et al.*, 2006) which also utilizes the occurrences of short tokens matches within local genomic regions for seeding. However, despite of various implementations, they were mainly designed for shorter reads (from several tens to hundreds of bp) with lower error rates (around several percent), i.e. the NGS reads. This design may be not practical to SMRT read alignment. For rHAT, the characteristics of SMRT reads are fully considered in the design of genome indexing and read alignment modules for improving the performance.

Skip to Main Content

We benchmarked rHAT on real and simulated SMRT datasets from various prokaryote and eukaryote genomes. The experimental results demonstrated that rHAT can

effectively align SMRT reads with a considerable improvement on throughput. Moreover, it has superior ability in terms of end-to-end alignment which can produce consecutive alignments for most of the reads.

2 Methods

2.1 Overview of rHAT

rHAT organizes the reference genome by a series of windows, and builds RHT for indexing the *k*-mers within each of the windows in advance. In the seeding phase, a long substring of the read is employed as the 'seeding region'. All the matches of the *k*-mers within the seeding region are retrieved through RHT. These matches are subsequently used for calculating the occurrences of *k*-mer matches in various windows, and windows with most *k*-mer matches are employed as candidate sites for extension. This strategy is motivated by previous study (Rasmussen *et al.*, 2006), which suggested that, the occurrence of short token matches within local regions can be used as robust signal for seeding. It also refers to the study of BLASR (Chaisson and Tesler, 2012) which demonstrated that, for a long query string (e.g. >1000 bp) with moderate error rate, it has a high probability that many short token matches occur at its actual site. Considering these two issues and the superior length of SMRT reads, a long enough substring could be informative enough for seeding while it can directly reduce the number of short tokens to be handled. Moreover, rHAT utilizes the matches of short *k*-mers instead of variable length seeds, as *k*-mers can be retrieved more efficiently through hash-table-based index.

In the extension phase, rHAT utilizes a SDP-based heuristic from local short token matches for further reducing the cost. Moreover, a constraint on the distance between neighboring matches is introduced into this heuristic for preventing ill-defined alignments. Furthermore, if it fails in finding suitable end-to-end alignment, rHAT can also perform split alignment for the read, which is useful for handling structure variations as well as the potential errors in the seeding phase.

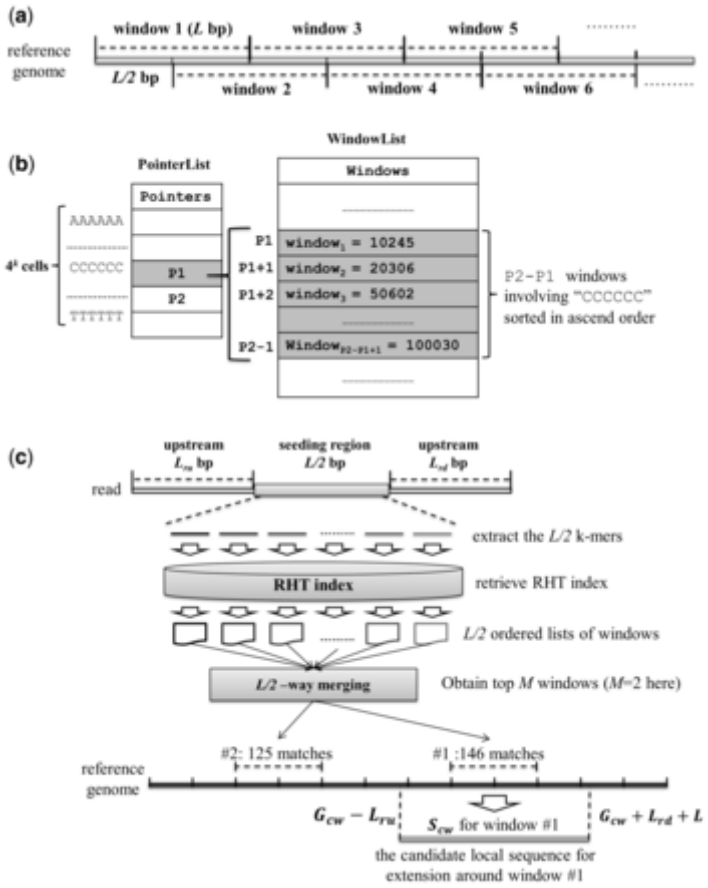
rHAT mainly consists of four modules, i.e. (i) index construction; (ii) seed generation and prioritization; (iii) extension at candidate sites; (iv) spilt alignment for chimeric reads. Some details are as following.

Skip to Main Content

2.2 Index construction

The reference genome is partitioned into a series of windows for describing the k-mers within the local genomic regions. The windows are L bp long, each overlapping with its neighbors by $L/2$ bp (Fig. 1a). The $L/2$ bp overlapping between neighboring windows is necessary, since it guarantees that a $L/2$ bp long substring of the read can be fully accommodated by a specific window for seeding (see more details in Section 2.3). Otherwise, all the windows could at most overlap with a part of the substring, which may affect the calculation of the occurrences of k-mer matches. L is selected as 2048 bp for guaranteeing high number of k-mer matches.

Fig. 1.



View large Download slide

A schematic illustration of the seeding operation. **(a)** The reference genome is partitioned into a series of *L* bp windows, each overlapping its neighbors by *L/2* bp. **(b)** The main data structure of RHT. RHT mainly consists of two lists, i.e. PointerList and WindowList. PointerList is a table having 4^k cells. Each cell records a pointer corresponding to the list of the windows for a certain *k*-mer. WindowList records all the window lists, each occupies a set of consecutive cells. Moreover, in each of the list, the involved windows are recorded in ascending order. In this case, P1 points to the window list of 'CCCCCC', and the length of the list is represented by the difference between neighboring pointers, i.e. P2–P1. **(c)** The seeding of rHAT. The window lists of the *L/2 k*-mers of the seeding region are retrieved through RHT index. An *L/2*–way merging is performed for finding the top *M* windows (*M* = 2 in this case) with most occurrences of *k*-mer matches as candidates. The candidates are further extended for extension

Skip to Main Content

For each of the *k*-mers of reference genome, rHAT records all the windows it occurred to build RHT index. RHT mainly consists of two main data structures (Fig. 1b):

1. 'PointerList' with 4^k cells directly records the pointers to the corresponding lists of windows for all the possible *k*-mers. The parameter *k* is typically configured as 11–15 bp, considering the error rate of SMRT reads.
2. 'WindowList' records all the window lists. For each *k*-mer, all the windows it occurred are pre-sorted and recorded in ascending order in its own list, which occupies a set of continuous cells of WindowList.

With the two data structures, an ordered list of all the hits of a certain *k*-mer can be directly retrieved with small overhead. It is also worthnoting that, due to the overlapping between neighboring windows, a certain position one *k*-mer occurs can be involved in two windows. In practice, RHT only records the downstream one, and recovers the upstream one on-the-fly.

2.3 Seed generation and prioritization

rHAT finds the windows with most *k*-mer matches in four steps as following (Fig. 1c):

1. rHAT extracts the $L/2$ bp substring centered at the read as seeding region. For reads shorter than $L/2$ bp, the whole read will be employed.
2. rHAT retrieves the window lists of the *k*-mers within the seeding region through RHT.
3. rHAT performs a multi-way merging on the pre-sorted window lists to calculate the occurrences of *k*-mer matches of various windows.
4. rHAT subsequently performs a heapsort to prioritize the windows, and selects the top *M* windows as candidates.

rHAT further extends the *M* candidate windows. For a candidate window whose offset is G_{cw}, assuming the lengths of the regions flanking the seeding region on the read are respectively L_{ru} (upstream) and L_{rd} (downstream), the window is extended to a local region, $[G_{\mathrm{cw}} - L_{\mathrm{ru}},\ G_{\mathrm{cw}} + L_{\mathrm{rd}} + L]$, and the corresponding genomic sequence within the region (denoted as S_{cw}) is used for extension.

Skip to Main Content

As the extension for noisy long read could be expensive, the choice of the parameter *M* is critical to the throughput. Since the windows are long and sparsely placed along the reference genome, most of the local sequences within various windows are non-

repetitive. Thus, for most reads, it is enough to use only a few top windows to cover its actual site, which is beneficial for reducing the cost of extension. Various settings on the *M* parameter were considered in the benchmarking for investigating its effect (shown in the Section 3).

If the seeding region of the read comes from a repetitive region longer than $L/2$ bp, the actual site could be not involved in the top *M* candidates, and an error may occur. However, due to the large read length, it is also possible that the whole read spans the repeat and rHAT can find that the read is very divergent from the candidate site. In such cases, rHAT will fail in end-to-end alignment, and the chimeric read alignment strategy will be triggered to realign the read, which may rescue the errors. But if the read is from a very long repetitive region which it cannot span, it is still possible to be end-to-end aligned to a false site. This problem is hard to solve in theory (Treangen and Salzberg, 2011), however, the benchmarks on the datasets from various genomes indicated that, for most reads analogous to the P5/C3 or above release of SMRT sequencing, their lengths are large enough to span the repeats in practice, and such errors are relatively rare.

2.4 Extension at candidate sites

rHAT aligns the read to each of the local sequences with the SDP-based heuristic approach, and prioritizes the alignments by their scores. Given a certain S_{cw}, it is mainly implemented in four steps as following.

1. A lookup table is built for indexing all the *l*-mers of the read (*l* = 11 bp in default), and rHAT scans S_{cw} from upstream to downstream to capture all the $\geqslant l$ bp matches between S_{cw} and the read through the lookup table. Each of the matches can be denoted as a triplet (VR_i, VS_i, VL_i), where VR_i and VS_i are respectively the offsets of V_i on the read and S_{cw}, and VL_i is the length of the match.

2. rHAT builds a direct acyclic graph (DAG) to compose the skeleton of the alignment (Supplementary Fig. S1a). The vertices of the DAG consist of all the recorded matches, V_i, plus two auxiliary vertices, $V_{\text{start}}(0,0,0)$ and $V_{\text{end}}\ (L_R + 1,\ L_S + 1, 0)$, where L_R and L_S are respectively the lengths of the read and S_{cw}. The two auxiliary vertices represent the start and the end of the alignment. The edges of the DAG consist of the links between the pairs of vertices $V_i \rightarrow V_j$ which meets the following conditions, $VR_i + VL_i \leq VR_j \leq VR_i + VL_i + T_{\text{wait}}$ and $VS_i + VL_i \leq VS_j$, where T_{wait} is a constraint parameter for modeling the error rate of the read.

Skip to Main Content

3. rHAT finds the optimal path connecting V_{start} and V_{end} which maximizes the total number of matched bases as the skeleton of alignment. This is implemented by scoring the vertices with the following recurrence equation.

$$ML(V_j) = \max\{ML(V_i) + VL_j\},\ V_i \in \text{precursors}(V_j)$$

where V_j denotes a vertex except V_{start}, and V_i is a precursor of V_j.

4. The skeleton partitions the read and S_{cw} into a series of paired segments (Supplementary Fig. S1b). rHAT separately aligns each of the unaligned pairs of segments with banded Smith-Waterman algorithm (Li, 2013), and combines the alignments of all the pairs to compose the alignment for the whole read. The mapping quality is estimated by the formula, $60 \times (AS_1 - AS_2)/AS_1$, where AS_1 and AS_2 are respectively the scores of the best and second best alignments.

The parameter T_{wait} describes the maximum allowed distance on the read between two vertices connected by an edge. This setting is motivated by that, given a certain error rate, the waiting length, i.e. the distance between two error-free tokens on the read can be modeled by geometric distribution (Chaisson and Tesler, 2012). In rHAT, this property is utilized for considering the error rate of the read and preventing ill-defined alignment. If the read is aligned to a false positive candidate site, or there is large variation event (e.g. structure variation) within the read, there could be much fewer local matches for some parts of the read. Under this circumstance, the threshold T_{wait} may trigger a failure for building the skeleton due to that some vertices are lack of precursors. The failed candidate will be discarded to avoid potential errors. For SMRT reads, we assume the error rate is 15%, and set T_{wait} with large enough values according to the minimum length of local matches. With the setting of T_{wait}, it has very low probability (about 10^{-8}) that the distance between two neighboring true positive $\geqslant l$ bp matches can be over T_{wait}. In practice, to prevent failures caused by extraordinary sequencing errors and improve the consecutiveness of the alignment, for a read which all its candidates failed in end-to-end alignment, rHAT can also dynamically double the threshold for composing more sensitive alignment. However, such alignments would be assigned lower mapping qualities, as it could introduce more uncertainty.

2.5 Chimeric read alignment

Skip to Main Content

rHAT treats a read failed in end-to-end alignment as chimeric read, and deals with it in a split alignment strategy. The read would be split into a series of non-overlapping

segments at first. Each of the segments is $L/2$ bp long only except for the last one which could be at most L bp. rHAT treats each of the segments as an individual read, and separately finds candidate sites for them with the seeding strategy mentioned above. rHAT further chains the neighboring segments whose candidate sites are also consecutively placed in the reference genome. After the chaining, a set of merged segments with specific candidate sites is generated, and rHAT separately aligns each of them to the corresponding candidate sites with the SDP-based approach.

This strategy can provide split alignments for the reads with large scale events, such as structure variations. Moreover, as the failure of the end-to-end alignment may be also caused by the false positive candidate sites, this strategy is helpful for correcting such errors as well, i.e. the re-seeding of the segments could re-align the segments to their actual sites.

3 Results

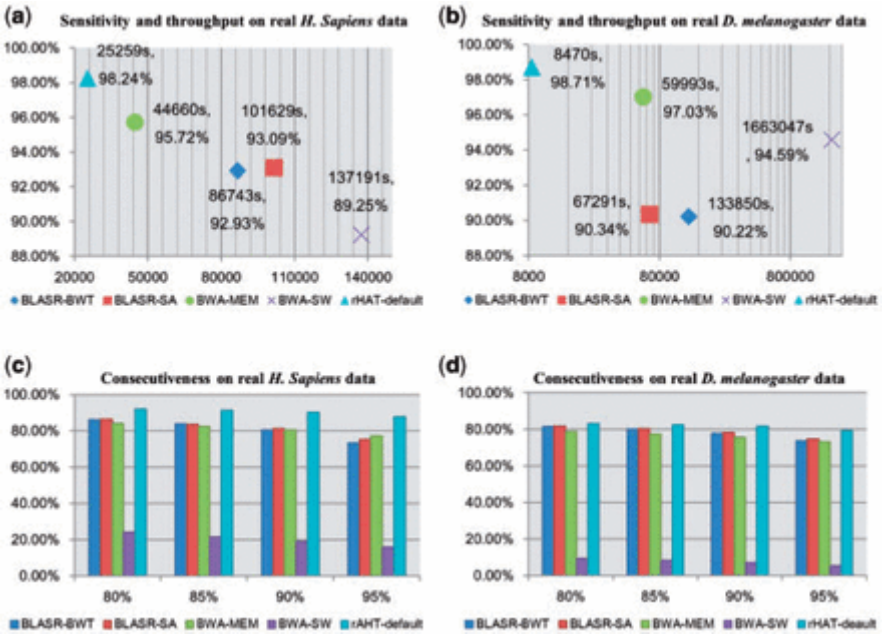
Seven real and simulated SMRT datasets from various genomes (Supplementary Table S1) were used for benchmarking rHAT and three state-of-the-art aligners, BLASR (version 1.3.1), BWA-MEM (version 0.7.12) and BWA-SW (version 0.7.12). rHAT were run with a variety of configurations on the parameters; BLASR was run in its default setting with two kinds of indices, i.e. BWT and suffix array (SA), respectively; BWA-SW was run with parameters (*-b*5 *-q*2 *-r*1 *-z*20) referring to previous study (Carneiro *et al.*, 2012); and BWA-MEM was run with the SMRT optimized parameters (*-x* pacbio). More details of the implementation of the benchmarking are in Supplementary Protocol.

3.1 Benchmarking on real data

Two SMRT P5/C3 release datasets respectively from *H. sapiens* and *D. melanogaster* genomes were used for evaluating rHAT on real SMRT data. The reads were respectively aligned to the reference genomes of *H. sapiens* (build GRCh37/hg19) and *D. melanogaster* (build DM5). Mainly, three issues were observed from the results (Fig. 2 and Supplementary Table S2).

Fig. 2.

Skip to Main Content



View large Download slide

Benchmarking on real datasets. **(a–b)** The throughputs and sensitivities of the aligners on the *H. sapiens* (a) and *D. melanogaster* (b) datasets. The horizontal and vertical axes are respectively the wall time of the alignment (in seconds) and the percentage of aligned bases. **(c–d)** The consecutiveness of the aligners on the *H. sapiens* (c) and *D. melanogaster* (d) datasets. The percentages marked on the horizontal axis indicate the threshold of the proportion of covered bases, i.e. $p_i^{\text{aln}}\,(i = 1, 2, 3, 4)$, for considering if the alignment is consecutive. The bars indicate the proportions of reads consecutively aligned by the aligners. Each point or bar in (a–d) also corresponds to a line of Supplementary Table S2

Firstly, rHAT can provide outstanding throughput (Fig. 2a, b). With the default setting (*-l* 11, *-w* 1000, *-m* 5, *-k* 13), rHAT is several folds faster than other aligners, only except for BWA-MEM on the *H. Sapiens* dataset where rHAT still outperforms but they are comparable. With other settings (Supplementary Table S3), rHAT could be even faster with similar sensitivity. For example, it is much faster than BWA-MEM on the *H. Sapiens* dataset with (*-l* 11, *-w* 1000, *-m* 5, *-k* 15). This throughput is beneficial for improving the overall efficiency of SMRT data analysis.

Secondly, rHAT can provide sensitive alignments (Fig. 2a, b). For these two datasets, rHAT had the least unaligned reads, while the other three aligners are also comparable to that of rHAT, indicating that all of them can be tolerant to the sequencing errors (Supplementary Table S2). We further assessed the proportion of aligned bases as a metric of base-level sensitivity. This metric is also important to long read alignment, since it could still be lack of information for downstream analysis if reads are only partially aligned. rHAT aligned most bases, indicating that it also achieved the best base-level sensitivity. We further inspected the details of the alignments, and found that, more reads had a large proportion of bases being clipped by the other three

Skip to Main Content

aligners. However, rHAT aligned fewer reads with large clippings, which is the main reason of the higher base-level sensitivity of rHAT.

Thirdly, rHAT can provide consecutive alignments (Fig. 2c, d). Since structure variation is relatively rare in genome, most reads are SV-free and they should be consecutively aligned in essence. Moreover, as some of SV detection approaches use soft-clippings as signal (English *et al.*, 2014), if too many SV-free reads were split aligned, the downstream analysis could be also affected in practice. For assessing the consecutiveness of the alignments, we set four thresholds, i.e. $p_1^{\mathrm{aln}} = 80\,\%$, $p_2^{\mathrm{aln}} = 85\,\%$, $p_3^{\mathrm{aln}} = 90\,\%$, and $p_4^{\mathrm{aln}} = 95\,\%$, and investigated the proportions of the reads which have at least one alignment covering at least p_i^{aln} $(i = 1, 2, 3, 4)$ of its bases. The result indicates that rHAT consecutively aligned more reads than those of the other three aligners. Moreover, it is worthnoting that BWA-SW generated consecutive alignments for only a small portion of reads.

3.2 Benchmarking on simulated data

For further assessing the sensitivity, accuracy, throughput of rHAT, as well as its scalability to genomes in various sizes, we benchmarked rHAT with the simulated datasets from five various genomes, i.e. *E. coli* (the 536 strain), *S. cerevisiae* (build sacCer3), *D. melanogaster* (build DM3), *A. thaliana* (build TAIR10) and *H. sapiens* (build GRCh37/hg19). For each genome, a 1x coverage dataset was generated by PBSim (Verison 1.0.3) (Ono *et al.*, 2013). The average read length was configured as 8000 bp for mimicking the P5/C3 or above release of SMRT sequencing, and 15% sequencing errors (12% insertions, 2% deletions and 1% substitutions) were introduced referring to previous study (Carneiro *et al.*, 2012). Mainly, the following four issues were observed from the results (Table 1 and Supplementary Table S3).

Table 1.

Results of the aligners on the simulated *H. sapiens* dataset

Aligner	Aligned bases %	Correctly aligned reads %	Correctly aligned bases %	Running time (s)
BLASR-BWT	99.64	96.65	96.33	79 041
BLASR-SA	99.77	96.24	96.04	111 169

Skip to Main Content

Aligner	Aligned bases %	Correctly aligned reads %	Correctly aligned bases %	Running time (s)
BWA-MEM	99.71	98.01	98.21	111 451
BWA-SW	97.97	0.00	95.44	297 301
rHAT	99.97	99.25	99.35	17 172

'Aligned bases%' indicates the proportion of bases which are covered by at least one alignment of the corresponding reads. 'Correctly aligned reads%' indicates the proportion of reads which are correctly aligned. It is also worthnoting that the BWA-SW has a very low ratio of correctly aligned read mainly due to that it cannot align most of the reads consecutively. 'Correctly aligned bases%' indicates the proportion of bases which are covered by position-correct alignments. 'Running time' indicates the wall time of the alignment, measured in seconds. Each line also corresponds to a line of the '*H. sapiens*-sim dataset' part of Supplementary Table S3.

View Large

Firstly, similar to that of real data, rHAT significantly outperformed other aligners in speed for all the datasets, indicating its good efficiency as well as scalability to various sizes of reference genomes.

Secondly, all the aligners generated sensitive alignments for almost all the reads, while rHAT marginally aligned more bases than the other three aligners.

Thirdly, rHAT correctly aligned the reads. As it is important to simultaneously recover the correct positions and produce consecutive alignments for long reads, we considered a read is correctly aligned only if there is an alignment whose leftmost position is within 50 bp of the grand truth deducting the clipped parts, meanwhile, it can cover at least 80% of the bases of the read. With default setting, rHAT correctly aligned more reads than other competitors for the *S. cerevisiae*, *A. thaliana* and *H.sapiens* datasets. For the *E. coli* and *D. melanogaster* datasets, BWA-MEM was most accurate and rHAT was the best runner-up. Overall, rHAT achieved the best average correctness for the five datasets (99.15%). It is also worthnoting that BWA-SW cannot generate consecutive alignment for almost all the reads, so that its correctness is hard to assess by this metric.

Fourthly, we investigated the proportion of bases covered by the position-correct alignments for further assessing the correctness of the alignments. Here, an alignment is considered as position-correct only if its leftmost position is within 50 bp of the grand truth deducting the clipped parts, regardless of the number of bases

Skip to Main Content

being covered. Although it does not consider the consecutiveness of the alignments, this metric describes the usability of the alignments without the bias against tools using clipping, and it can also fully consider all the split alignments for a given read. On this metric, the trend was similar. rHAT is still the winner on the *S. cerevisiae*, *A. thaliana* and *H. Sapiens* datasets, and achieves highest average on the five datasets (99.31%), while BWA-MEM is more accurate on the *E. coli* and *D. melanogaster* datasets. Moreover, BWA-SW also correctly aligned comparable proportions of bases to those of other aligners, indicating that the alignment of BWA-SW is also accurate, although it is usually not consecutive.

Because of the wide application of human genome re-sequencing, we further inspected the reads of the *H. sapiens* dataset which were incorrectly aligned by rHAT, and found that the errors were mainly caused by three issues as following.

i. False positive local matches could mislead the skeleton of the alignment. Due to the repetitiveness of local genomic sequence, some of the vertices of the SDP graph could be false positive matches. These matches may be mistakenly involved in the skeleton, since it could increase the total number of matched bases. The mistakenly involved local matches were usually the matches of the short tokens near the ends of the reads, since the SDP heuristic is more effective to prune the false positive matches of the short tokens within the inner part of the read by considering the total number of matched bases (Supplementary Fig. S2). Moreover, as the error usually happens near the ends of the read, the bases of the inner part of the read can still be appropriately aligned.

We tuned the threshold on the length of local matches and found that longer local matches worked better for reducing such errors since they are less false positive. Considering both of the accuracy and the sensitivity of the alignment, we choose $l = 11$ bp as default value. With this setting, 1776 reads (0.50%) of the simulated *H. sapiens* dataset were consecutively aligned to incorrect positions due to this issue. The results of other settings on this parameter are in Section 3.3.

We also investigated its influence on real data, as this is the issue mostly affected the accuracy. The influence is difficult to assess due to lack of grand truth. However, in most cases, the false positive local matches could cause large indels around the beginning part of the alignment, which misleads the read to a wrong position. Thus, for each of the reads, we consider that it is potentially affected if there is a >50 bp indel in the primary alignment of its leftmost 300 bp (for a read aligned to the reverse strand, we checked the primary alignment of its rightmost 300 bp). 9861 reads (3.39%) of real *H. sapiens* dataset meet this condition. This ratio is higher than that of

the simulated dataset. However, it is also worthnoting that this could be an overestimation of the influence, i.e. the actual number of the reads being affected should be lower, since such indels may also be caused by real SVs. An example is shown in Supplementary Figure S3. In this case, there is a 144 bp deletion near leftmost part of the alignments, which coincides with a recent study of SV on the same human sample (Chaisson *et al.*, 2015). BLASR also produced a similar alignment for this read. However, both of BWA-MEM and BWA-SW clipped the corresponding part of the read, which may be less informative to downstream SV analysis. Except the 9861 reads, there are 258 070 reads consecutively aligned by rHAT with more confidence that they were not influenced by the false positive matches. This number is still higher than those of the reads consecutively aligned by the other three aligners.

ii. The repetitiveness of reference genome could also cause incorrect alignment. Since rHAT only aligns a given read to the top M candidates considering the tradeoff between the effectiveness and efficiency, the true site may be missed if the read is from a long repetitive region. In such cases, the read could be end-to-end aligned to incorrect positions if the whole read is not long enough to span the repetitive region. However, due to the large read length, only 100 reads (0.03%) of the simulated *H. sapiens* dataset were consecutively aligned to incorrect positions due to this issue. Moreover, when the read was aligned to long repetitive regions, rHAT could still find it hard to confidently align the read, and assign the read a lower mapping quality score. Furthermore, the results of the various settings on the number of candidates (see in Section 3.3) indicated that employing more candidates could reduce such errors, but at the expense of speed.

iii. 804 reads (0.23%) of the simulated *H. sapiens* dataset were resulted as incorrectly aligned due to that rHAT failed in aligning them consecutively. Most of them were considered as chimeric read and handled by the split alignment strategy. Similar to those of consecutively aligned reads, some bases of these split aligned reads were also mistakenly handled, also mainly due to the two issues mentioned above. To measure the errors, we assessed the number of bases misaligned by split alignment. In total, 2 442 179 bases (0.09%) of simulated *H. sapiens* dataset were incorrectly aligned by split alignment.

3.3 Results of rHAT with various parameter settings

Skip to Main Content

Other than the default setting, we also tuned five parameters most related to the performance of rHAT to investigate their effects, i.e. the minimum length of the local matches used for SDP (`-l`), the maximum allowed window hits per k-mer (`-w`), the

number of candidates for extension (-*m*), the size of the *k*-mer (-*k*), and the number of threads (-*t*).

The minimum length of the local matches used for SDP (-*l*) could influence the sensitivity and the correctness of the alignments (Supplementary Tables S4 and Supplementary Data). Smaller local matches (e.g. $l = 8$ bp) may have slightly better sensitivity, however, there could also be more false positive matches which may mislead the skeleton of the alignment and affect the accuracy. Thus, it is better to use higher threshold, e.g. $l = 11$ bp, to simultaneously achieve good sensitivity and correctness.

The maximum allowed window hits per *k*-mer (-*w*) marginally affected the alignment (Supplementary Tables S4 and Supplementary Data). With various configurations, similar sensitivities and correctness were obtained. Moreover, higher allowed number (e.g. $w = 2000$) could be at the expense of throughput, mainly due to the increasing cost of merging window lists.

The number of candidates for extension (-*m*) also influences the throughput (Supplementary Tables S4 and Supplementary Data) since it is expensive to align the noisy long read to many candidate sites. However, employing too few candidates, e.g. $m = 1$ or 2, could speed up at the expense of correctness, since for some reads, the actual sites may not be prioritized as the top candidates due to the repetitiveness of the genome. Employing more candidates could enable rHAT to miss less actual sites and improve the sensitivity and correctness. Considering the tradeoff between effectiveness and efficiency, $m = 5$ was used as the default setting. Moreover, it is also good to tune this parameter for various application scenarios.

The *k*-mer size (-k) also has effect on the throughput (Supplementary Tables S4 and Supplementary Data), mainly due to that, smaller *k*-mer size may have lower selectivity, i.e. the *k*-mers may hit more windows, which may also increase the cost of merging. Moreover, too small *k*-mer size may make many *k*-mers be ignored by rHAT due to that they hit too many windows, which exceeds the limitation on the number of windows per seed. An example is setting $k = 10$ for the simulated *H. sapiens* dataset (Supplementary Table S8). In this case, rHAT ignored many k-mers as they hit more than 1000 windows, resulted in poor sensitivity and accuracy. However, too large *k*-mer size may also affect the sensitivity since there would be fewer *k*-mer matches due to the high error rate.

Skip to Main Content

Furthermore, *k*-mer size is also critical to the memory footprint, since it partially determines the size of RHT. There are 4^k and $|G|-k+1$ cells for the PointerList and

WindowList, respectively, where |G| is the size of the reference genome. Each of the cells occupies 32 bits to support genome upto 4 Tbp, as it needs log(2|G|/L) bits to record a window ID. With this implementation, the RAM usage may be not practical for too large k-mer size. In practice, with some additional data structures, the peak memory footprints of rHAT were 13.70 ($k = 13$) and 17.40 gigabytes ($k = 15$) for the real *H. Sapiens* dataset, and 1.01 ($k = 13$) and 4.15 ($k = 15$) gigabytes for the real *D. melanogaster* dataset. This requirement can fit the configurations of most modern servers and workstations, meanwhile, $k = 13–15$ can also produce good results.

Since parallelization is also very important to read alignment tasks, we also benchmarked the throughput of rHAT in multiple threads. The three relatively large simulated datasets, *D. melanoganster*, *A. thaliana* and *H. sapiens*, were employed. rHAT and the other three aligners were run on a server with 8 cores (Supplementary Protocol). Each of the aligners was run in 2, 4 and 8 threads (Supplementary Table S9). The results suggested that rHAT was also faster than the competitors with the same number of threads.

4 Discussion

We propose rHAT, a novel seed-and-extension based approach for noisy long read alignment. Mainly, there are three key features of rHAT. (i) rHAT utilizes the occurrences of k-mer matches between a substring of the read and the reference genome for seeding, which directly reduces the short tokens that need to be handled without loss of effectiveness. (ii) With the help of RHT, the seeding can be efficiently implemented by multi-way merging. Furthermore, it also has a small overhead for retrieving k-mer matches. (iii) The SDP-based heuristic approach also reduces the cost of aligning the noisy long read to local genomic sites. Moreover, this heuristic approach also helps in preventing ill-defined alignment by the constraint on the distance between local matches.

rHAT was benchmarked on real and simulated datasets from various genomes. The experimental results demonstrated that rHAT can be robust to the high error rate and provide effective alignments for SMRT reads. Comparing to other state-of-the-art aligners, there are two other major contributions made by rHAT. First, rHAT has an outstanding throughput which could be beneficial for breaking the bottleneck of the application on SMRT data. Second, rHAT can more consecutively align SMRT reads, which could also facilitate downstream analysis.

Skip to Main Content

A major drawback of rHAT is that the SDP-based heuristic could be affected by the false positive local matches. This issue could make rHAT misalign a proportion of bases and map the read to an incorrect position nearby the correct one. In downstream analysis, two methods could be useful for reducing this affection. One is to directly clip the alignments which have large indels within their beginning parts. The other one is to correct the potential errors by local re-alignment which is based on the multiple sequence alignment (MSA) of the reads aligned to the same local genomic region.

Considering its sensitivity, correctness, consecutiveness and throughput, rHAT is overall an effective and efficient tool for SMRT read alignment. With the explosive increase of sequencing data, the improvement made by rHAT is profitable to various applications of SMRT sequencing. It could be a good choice to incorporate rHAT into the developing computational biology pipelines to leverage cutting-edge genomic studies. Future works will focus on two aspects. Firstly, the sequencing quality scores are still not taken into account. In the future, we will develop novel method to incorporate the quality scores into extension phase for further improving the quality of alignment. For example, the quality scores may be useful for more effectively recognizing false positive local matches which is beneficial for mitigating their affection on the building of skeleton. Secondly, the memory footprint of rHAT is relatively high, and the k-mer size is also limited by the data structures of RHT. Considering larger k-mer size could be also helpful for aligning noisy long reads (Chaisson and Tesler, 2012; Li, 2013), we will also focus on developing novel succinct data structures which can support larger k-mer size for seeding and further reduce the memory footprint.

Acknowledgements

We are grateful to Dr. Michael Brudno from University of Toronto for helpful discussions and suggestions.

Funding

This work has been partially supported by the National Nature Science Foundation of China (grant numbers: 61173085 and 61301204), the National High-Tech Research and Development Program (863) of China (grant numbers: 2012AA020404, 2012AA02A604, 2014AA021505 and 2015AA020101).

Conflict of Interest: none declared.

References

Ahmadi A. et al . (2012) Hobbes: optimized gram-based methods for efficient read alignment. *Nucleic Acids Res.* , 40, e41.

Google Scholar CrossRef PubMed

Carneiro M.O. et al . (2012) Pacific biosciences sequencing technology for genotyping and variation discovery in human data. *BMC Genomics* , 13, 375.

Google Scholar CrossRef PubMed

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

Google Scholar CrossRef PubMed

Chaisson M.J. et al . (2015) Resolving the complexity of the human genome using single-molecule sequencing. *Nature* , 517, 608–611.

Google Scholar CrossRef PubMed

Chin C.S. et al . (2013) Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat. Methods* , 10, 563–569.

Google Scholar CrossRef PubMed

David M. et al . (2011) SHRiMP2: sensitive yet practical short read mapping. *Bioinformatics* , 27, 1011–1012.

Google Scholar CrossRef PubMed

Eid J. et al . (2009) Real-time DNA sequencing from single polymerase molecules. *Science* , 323, 133–138.

Google Scholar CrossRef PubMed

English A.C. et al . (2014) PBHoney: identifying genomic variants via long-read discordance and interrupted mapping. *BMC Bioinformatics* , 15, 180.

Google Scholar CrossRef PubMed

Skip to Main Content

Eppstein D. et al . (1992) Sparse dynamic programming I: linear cost functions. *J. Assoc. Comput. Machinery* , 39, 519–545.

Google Scholar CrossRef

Fonseca N.A. et al . (2012) Tools for mapping high-throughput sequencing data. *Bioinformatics* , 28, 3169–3177.

Google Scholar CrossRef PubMed

Ferragina P. Manzini G. (2000) Opportunistic data structures with applications. Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp. 390–398.

Huddleston J. et al . (2015) Reconstructing complex regions of genomes using long-read sequencing technology. *Genome Res.* , 24, 688–696.

Google Scholar CrossRef

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

Google Scholar CrossRef PubMed

Kiełbasa S.M. et al . (2011) Adaptive seeds tame genomic sequence comparison. *Genome Res.* , 21, 487–493.

Google Scholar CrossRef PubMed

Koren S. et al . (2012) Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nat. Biotechnol.* , 30, 693–700.

Google Scholar CrossRef PubMed

Koren S. et al . (2013) Reducing assembly complexity of microbial genomes with single-molecule sequencing. *Genome Biol.* , 14, R101.

Google Scholar CrossRef PubMed

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

Google Scholar CrossRef PubMed

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

Google Scholar CrossRef PubMed

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

Skip to Main Content

Google Scholar CrossRef PubMed

Li H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997.

Ono Y. et al . (2013) PBSIM: PacBio reads simulator—toward accurate genome assembly. *Bioinformatics* , 29, 119–121.

Google Scholar CrossRef PubMed

Rasmussen K.R. et al . (2006) Efcient q-gram filters for finding all epsilon-matches over a given length. *J. Comput. Biol.* , 13, 296–308.

Google Scholar CrossRef PubMed

Roberts R.J. et al . (2013) The advantages of SMRT sequencing. *Genome Biol.* , 14, 405.

Google Scholar CrossRef PubMed

Treangen T.J. Salzberg S.L. (2011) Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nat. Rev. Genet.* ,13, 36–46.

Google Scholar PubMed

Weese D. et al . (2012) RazerS 3: faster, fully sensitive read mapping. *Bioinformatics* , 28, 2592–2599.

Google Scholar CrossRef PubMed

Yanovsky V. (2014) Various algorithms for high throughput sequencing. Ph.D thesis of University of Toronto.

Supplementary data

Supplementary Data
– zip file

Comments

0 Comments
Skip to Main Content