Bioinformatics, 34(21), 2018, 3631–3637
doi: 10.1093/bioinformatics/bty398
Advance Access Publication Date: 18 May 2018
Original Paper

Sequence analysis

Combining probabilistic alignments with read pair information improves accuracy of split-alignments

Anish M. S. Shrestha[1,*], Naruki Yoshikawa[1] and Kiyoshi Asai[1,2]

[1]Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, University of Tokyo, 5-1-5, Kashiwanoha, Kashiwa-shi, Chiba, Japan and [2]Artificial Intelligence Research Center, AIST, 2-3-26 Aomi, Koto-ku, Tokyo, Japan

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on February 7, 2018; revised on April 19, 2018; editorial decision on May 7, 2018; accepted on May 13, 2018

Abstract

Motivation: Split-alignments provide base-pair-resolution evidence of genomic rearrangements. In practice, they are found by first computing high-scoring local alignments, parts of which are then combined into a split-alignment. This approach is challenging when aligning a short read to a large and repetitive reference, as it tends to produce many spurious local alignments leading to ambiguities in identifying the correct split-alignment. This problem is further exacerbated by the fact that rearrangements tend to occur in repeat-rich regions.

Results: We propose a split-alignment technique that combats the issue of ambiguous alignments by combining information from probabilistic alignment with positional information from paired-end reads. We demonstrate that our method finds accurate split-alignments, and that this translates into improved performance of variant-calling tools that rely on split-alignments.

Availability and implementation: An open-source implementation is freely available at: https://bitbucket.org/splitpairedend/last-split-pe.

Contact: anish@edu.k.u-tokyo.ac.jp

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

1 Introduction

A split-alignment is a pairwise sequence alignment in which different parts of the query align to disjoint regions in the reference. Split-alignments are important in comparative genomic studies as they provide direct evidence of rearrangements such as large deletions, inversions, or chromosomal translocations. Indeed, several methods that detect such rearrangements from DNA sequencing reads of a sampled genome, rely on split-alignments (e.g. Layer *et al.*, 2014; Rausch *et al.*, 2012; Zhao and Zhao, 2015). Given that sequencing based variant detection is becoming part of standard workflow in biological and medical studies, it is imperative that split-alignments are computed accurately.

Split-alignments cannot be found using conventional alignment methods. Scoring schemes based on affine gap penalty or those based on edit-distances are not suited, nor modeled, for events such as large deletions or translocations. Moreover, since split-alignments may be non-collinear (i.e. not left-to-right on both reference and query), they do not yield to standard Smith-Waterman-type algorithms and require specialized dynamic programing techniques (Frith and Kawaguchi, 2015). In practice, a split-alignment is obtained by first computing high-scoring local alignments and then ‘stitching’ parts of them together, to obtain a final split-alignment (Faust and Hall, 2012; Frith and Kawaguchi, 2015).

However, this approach becomes challenging when the query is a short read obtained from a high-throughput sequencer. This is largely because, for a short read, numerous spurious high-scoring local alignments may be found, confounding the true split-alignment. This could happen due to several factors: reads are error-prone, the reference is very large and highly repetitive, and rearrangements tend to accumulate in repeat-rich regions.

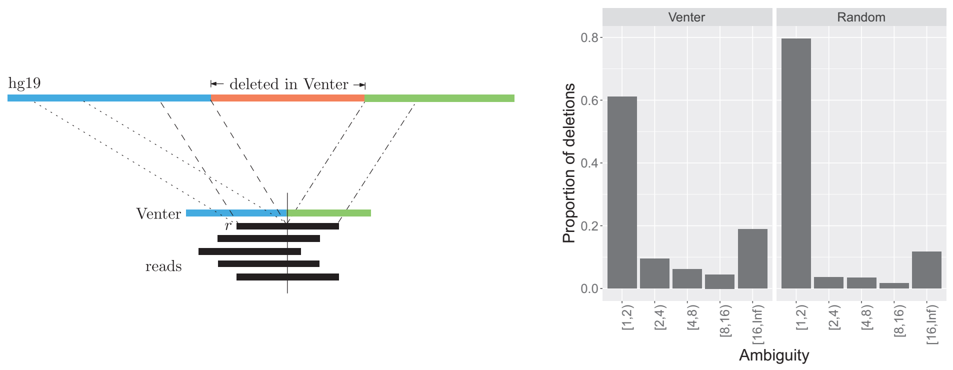

Fig. 1. (Left) From a deletion locus in the Venter genome, reads flanking the breakpoint are simulated, for all possible flank sizes. For one such read *r*, its left flank and right flank have two and one exact matches, respectively, in the reference; and therefore the split-alignment ambiguity for *r* is 2. The *ambiguity* for this deletion is the mininum split-alignment ambiguity among all its reads. (Right) Distribution of deletions based on their ambiguities, for deletions in the Venter genome and for artificial deletions placed randomly in hg19

To illustrate how severe the issue of spurious alignments can get, we conducted a simple study in which we try to recover known large ($\geq$20 bp) deletions in the Venter genome (Levy *et al.*, 2007) from simulated error-free short (100 bp) reads from the loci of those deletions. To simulate high coverage, we generate for each deletion locus, reads that flank the deletion breakpoint with all possible flank sizes (see Fig. 1) . Finding the correct split-alignment of a read would require finding the correct pair of local alignments that contain its two flanks. However there might be many candidate pairs to choose from. A conservative estimate of the number of candidate pairs is given by the product of the number of exact matches in the reference of its left flank and that of its right flank—which we shall call the *split-alignment ambiguity* (see e.g. Fig. 1). For a deletion, we define its *ambiguity* as the minimum split-alignment ambiguity among all its reads. According to this definition, a deletion can be identified unambiguously (ambiguity = 1) if there is at least one read that is positioned such that each of its flanks matches exactly once to the reference. Higher ambiguity values indicate deletions located in regions that are hard-to-align due to repetitiveness. The distribution of Venter deletions based on their ambiguities is shown in Figure 1. When compared with the background distribution obtained from randomly placed artificial deletions with the same size spectrum, we can see that real deletions tend to be located in hard-to-align regions. What is more surprising is that a significant proportion (almost 40%) of Venter deletions cannot be identified unambiguously, even under our ideal conditions of error-free reads, stringent alignment scoring scheme that allows only for exact matches, highest-possible coverage, and best-case scenario definition of deletion ambiguity.

One source of information that can be used to resolve ambiguities in split-alignments and the associated rearrangement predictions is provided by paired-end reads. A paired-end read is obtained by sequencing a DNA fragment, typically a few hundred base-pairs, from both of its ends to obtain a pair of reads, also referred to as *mates*. The extra positional information that can be obtained from the mates can, in theory, be used in two ways. First, regions likely to contain large variants can be spotted by checking for discordant alignments, i.e. the case in which the distance between the alignments of the mates deviates significantly from what can be expected from the fragment length distribution. Second, we might be able to resolve the ambiguities in the split-alignment of a breakpoint-containing read by considering the alignments of its mate.

Current methods have largely focused only on the former, using discordant alignments in several ways. For instance, Delly

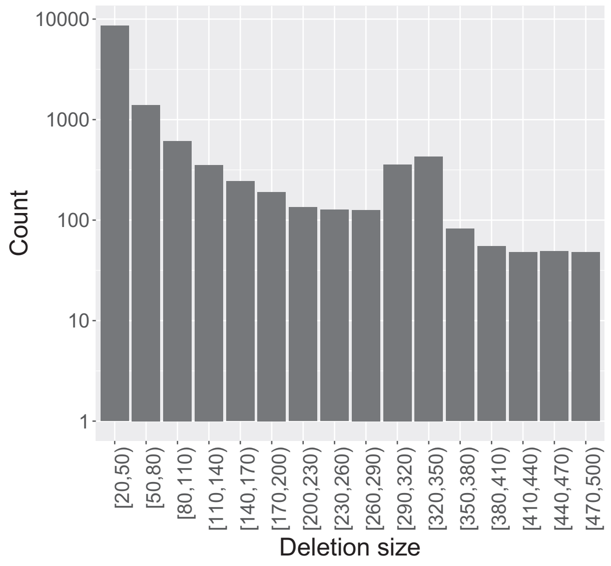

Fig. 2. Size distribution of deletions ($\geq$20 bp) present in the Venter genome

(Rausch *et al.*, 2012) uses them to filter genomic intervals that might contain large rearrangements, which are then searched for split-alignments. Lumpy (Layer *et al.*, 2014) has separate modules to search for evidence from paired-end and split-alignments, processing the independently obtained call-sets in the final stage. However, a serious limitation of using discordant-alignment signals is posed by the fact that fragment lengths are intrinsically variable due to the steps involved in library preparation. A quick analysis of 10 arbitrarily chosen NCBI SRA datasets of human-genome-based experiments showed that the standard deviation (SD) of fragment lengths can be as large as 128 bp (median of 80 bp). To accommodate for this variability, it is typical to not classify as discordant, cases in which the fragment length inferred from alignments of mates is within a certain threshold. The implication of any choice of threshold can be understood in the light of the deletion size distribution shown in Figure 2. The distribution is exponential (except for a peak that can be attributed to Alu elements), and thus deletions shorter than 110–140 bp account for most of the deletions. In this context, even with a stringent threshold (e.g. $\pm 3\times$ SD), we will miss a significant fraction of the deletions.

In this work, we take the latter approach, utilizing pairing information *during* split alignment computation itself. Our method builds on the work by Frith and Kawaguchi (2015), which employs a probabilistic sequence alignment technique to capture information about (split-) alignment uncertainty in the form of alignment column probabilities. Our main idea is to use a Bayesian updating procedure to revise these column probabilities based on information from paired-end reads, and to choose appropriate columns to construct a split-alignment. We do not know of any other split-aligner or variant caller that uses paired-end information in such a manner. We compare our method against several existing split-aligners and show that we find split-alignments more accurately. By pairing the aligners with downstream variant callers, we show that the gain in alignment accuracy translates directly into higher sensitivity and specificity of variant predictions, for simulated as well as real datasets.

2 Materials and methods

Let x be a read obtained from paired-end sequencing of a DNA fragment f, and let y be its mate. Let us denote the reference sequence by g. An overview of our method is as follows. First, we find high-scoring local alignments of x and y, separately, to g. Next we compute the probability of each column appearing in the alignments of x. We update the column probabilities using information coming from the alignments of y. Finally, we choose high-probability columns to form an alignment (possibly split) of x. We repeat this process for y with x as information source. These steps are described in detail below.

2.1 Step 1. Computing local alignments and column probabilities

We first compute local alignments of x and y to g, ignoring the pairing. Suppose we obtain sets $X = \{X_1, X_2, \ldots, X_{|X|}\}$ and $Y = \{Y_1, Y_2, \ldots, Y_{|Y|}\}$ of high-scoring local alignments of x and y, respectively, to g. If X contains only a single alignment or is empty, there is no split-alignment of x to find. If there are multiple local alignments of x, parts of them might constitute a split-alignment. However, if multiple alignments overlap on (part of) the read, it might not be clear where along the read the split occurs, if at all. Therefore, in the following steps, we consider each read position separately.

We digress slightly to review the notion of *column probability*. Let s, t be two DNA sequences. Consider bases $s[i]$ and $t[j]$. Let $\mathcal{A}$ be the set of all possible alignments between s and t, and let us assume that we have a probability distribution P over $\mathcal{A}$. Let $\mathcal{A}_{ij}$ be the subset of $\mathcal{A}$ comprising those alignments containing the ($s[i]$, $t[j]$) column, i.e. in which $s[i]$ and $t[j]$ are aligned to each other. The probability P_{st}^{ij} that $s[i]$ is aligned to $t[j]$, often referred to as 'column probability' can be computed as:

$$P_{st}^{ij} = \frac{\sum_{A \in \mathcal{A}_{ij}} P(A)}{\sum_{A \in \mathcal{A}} P(A)}. \qquad (1)$$

For the case where $\mathcal{A}$ consists only of non-split (i.e. conventional colinear) alignments, it is well known how to efficiently compute the column probabilities for all pairs of bases (Durbin *et al.*, 1998). In fact, this can also be done for the case where $\mathcal{A}$ allows split-alignments (Frith and Kawaguchi, 2015). We refer the readers to the corresponding references for details regarding the underlying probabilistic model and the algorithms for computation of column probabilities, for each case.

Returning to our problem setting, consider base $x[i]$, and suppose that out of the alignments in X, it appears in m_i alignments $X_{i_1}, X_{i_2}, \ldots, X_{i_{m_i}}$. To simplify the presentation, we assume for now that these alignments are gapless. Then, for each $j, 1 \le j \le m_i$, let $g[i_j]$ be the position in g to which $x[i]$ aligns in X_{i_j}. Let P_{i_j} be the probability that the alignment column $(x[i], g[i_j])$ is correct, which is given by the column probability P_{xg}^{i,i_j}. We assume here that the aligner used to find the local alignments also provides us with the column probabilities. This is in fact the case with LAST (Frith and Kawaguchi, 2015), and our method relies on it for this step.

2.2 Step 2. Updating the column probabilities based on pairing information

Next we update the column probabilities associated with each position i in x based on the information coming from its mate y in the form of the set of alignments Y. We will frame this as a Bayesian hypothesis testing scenario with the set $\{H_j^i | 1 \le j \le m_i\}$ of hypotheses, where each H_j^i states that $x[i]$ has been sequenced from position $g[i_j]$. Mate y will function as data in our setting. We set $P\left(H_j^i\right)$, the prior probability of H_j^i as the column probability P_{i_j} obtained in Step 1.

In order to compute the posterior probability $P\left(H_j^i|y\right)$, we need to define the likelihood $P\left(y|H_j^i\right)$. We distinguish two cases: (i) conjoint, when read y is informative about the alignment of $x[i]$, and (ii) disjoint when it's not. The latter accounts for cases such as chromosomal translocation or inversion in which $x[i]$ and y correspond to disjoint regions in g. A binary indicator variable I represents the two cases. Then the likelihood of having observed an alignment Y_k, $1 \le k \le |Y|$, is:

$$P\left(Y_k, I|H_j^i\right) \propto \begin{cases} P\left(l_f\right) \times e^{S(Y_k)} \times P(I=0) & \text{if conjoint} \\ \dfrac{1}{2l_g} \times e^{S(Y_k)} \times P(I=1) & \text{if disjoint,} \end{cases} \qquad (2)$$

where $S(Y_k)$ is the alignment score of Y_k, l_g is the length of the reference g, and l_f is the length of the fragment length that can be inferred from $g[i_j]$ and the first reference position in Y_k. Since an alignment score under traditional alignment scoring schemes are log-likelihood ratios, $e^{S(Y_k)}$ is proportional to the probability of Y_k (if the scoring scheme uses a scaling factor, say T, then we must first rescale the score to $S(Y_k)/T$). The term $2 \times l_g$ accounts for the number of bases in the two strands of a haploid genome. To compute the probability $P\left(l_f\right)$, we assume that fragment length follows a normal distribution, the parameters of which can be learnt from a sample of read pairs for which we can obtain confident unique alignments on the reference. We use a default value of $P(I=1) = 0.01$, but can be adjusted to suit the study—for instance higher values might be suitable for reads obtained from, say cancer genomes, where it is more likely that read pairs are disjoint. One way to estimate $P(I=1)$ is to align a sufficiently large sample of paired reads, and from confidently aligning pairs in the sample, to observe the fraction of read pairs for which their alignments are at a distance that is highly unlikely under the estimated fragment size distribution.

We can now express the probability of having observed read y as:

$$P\left(y|H_j^i\right) = \sum_k \sum_I P\left(Y_k, I|H_j^i\right), \qquad (3)$$

which leads to the computation of the posterior as:

$$P\left(H_j^i|y\right) = \frac{P\left(y|H_j^i\right) \times P\left(H_j^i\right)}{\sum_l P\left(y|H_l^i\right) \times P\left(H_l^i\right)}. \qquad (4)$$

We perform this column probability-updating procedure separately for each position in x.

To keep the description simple, we have deliberately omitted a few details: it might be the case that the prior probabilities of H_j^i do not add up to 1, or that $x[i]$ is aligned to a gap character in one of the local alignments. To handle such cases, we need to make slight adjustments to our scheme, which we describe in the Supplementary Material.

2.3 Step 3. Computing a final alignment

We wish to report a final alignment for x, in which each base of x appears at most once. Therefore we need to make a choice of which columns will appear in the reported alignment. For each base of x, we simply choose the column that has the highest posterior probability, and report the collection of all such columns as the final alignment. Although this ensures that a base appears at most once, it could in theory, result in alignments with unrealistic splits—for instance, i and $i+2$ in x align to position j and $j+2$ in g, respectively, but $i+1$ aligns to a distant position. For the 75 million pairs of reads that we simulated for performance testing described in the next section, we encountered no such cases; and therefore we assume that such pathological cases are not much of a concerning issue in practice.

Finally, we repeat the procedure for y, this time with the alignments in X treated as data.

3 Results

We compared the performance of our method (LAST-SPLIT-PE) with other aligners capable of split-alignment, using both simulated and real datasets. Of the methods compared, BWA can take paired reads, but we do not know of any document describing how it incorporates this information into split-alignments. YAHA does not consider pairing, but it employs a specialized algorithm for split-alignment. LAST-SPLIT also does not consider pairing, but since our method relies on it for finding initial local alignment and column probabilities, including it in the comparison allows us to investigate the gain from leveraging pairing information. Parameters used for each aligner is detailed in the Supplementary Material.

We performed three kinds of tests. First, using reads simulated from a reconstruction of the Venter genome, we examined how well each aligner can split-align reads that come from breakpoint regions. Next, we investigated the effect of split-alignment accuracy on the end goal of variant detection by feeding the output of each aligner to a common variant-calling tool (LUMPY). Finally, we applied the aligner-and-variant-caller pairings to a real read dataset obtained from the CHM1 cell line, for which a set of structural variants have been determined using an independent sequencing technology. In the following sections, we describe the specifics of each test and the corresponding results we obtained.

All data and scripts used for evaluation are available at the software website.

3.1 Evaluation of split-alignment accuracy

We first evaluated split-alignment accuracy using DNA reads simulated from the diploid genome of a human individual (C. Venter), which was reconstructed from the reference genome hg19 based on the variants present in the Venter genome (Levy *et al.*, 2007). The Venter variants make for a good benchmarking set as they are reportedly of high quality, having been obtained by aligning Sanger reads which are known for being accurate and long. As opposed to the (common) practice of simulating large variants by placing them in random locations of the reference, our simulation technique captures the difficulties of split-alignment (see Fig. 1).

Paired-end reads (length = 101 bp, mean fragment length = 300 bp and SD = 30 bp, total no. = 75 million pairs) were randomly drawn from the Venter genome. Sequencing errors were simulated by introducing errors in the reads, based on the per-base error probability encoded in the fastq file of the ERR1813601 dataset.

We examined the alignments of reads that come from sites of large deletions $\geq$ 20 bp in the reconstructed Venter genome. Specifically, we looked at those reads that cross a deletion breakpoint such that their flanks on either side are at least 20 bp. Of the total reads, there were about 42, 000 such reads. We define a read flank to be *correctly aligned* if at least one of bases in the flank is correctly aligned, and a read flank to be *wrongly aligned* if none of the bases is correctly aligned. A read is said to be *correctly split-aligned* if both the flanks are correctly aligned, and it is said to be *wrongly split-aligned* if either of the flanks is wrongly aligned. Note that we exclude the case in which an aligner only reports a correct local alignment for only one of the flanks. Although such alignments are not strictly wrong, they do not help the end goal of variant detection.

Figure 3 shows the results. Comparing LAST-SPLIT-PE to LAST-SPLIT, we can clearly see that there is a definite gain in utilizing pairing information in split-alignments. Our method also outperforms both BWA and YAHA over the whole range of mapping quality/error probability threshold values. It is also worth noting that even in the least stringent setting, we manage to correctly split-align only roughly a third of the 42 000 reads, while wrongly aligning many more reads. This is not surprising, given our observation about alignment challenges in deletion loci in Figure 1.

3.2 Base-level precision

Since our definition of a correctly split-aligned read seems somewhat generous, it is useful to check the precision of correct split-alignments at the base-level. For each aligner and for each mapping quality threshold used in Figure 3, we counted the total number of correctly aligned bases in those reads that were considered to be correctly split-aligned. This number expressed as a fraction of the total number of bases in the correctly split-aligned reads, does not vary much over the different mapping quality thresholds (Fig. 4). Averaged over all operating points, LAST-SPLIT-PE has a 97.38% precision, which is comparable to LAST-SPLIT (97.48) and slightly better than BWA (96.93) and YAHA (94.2). This result implies that when aligners align correctly, they are also precise at the base-level.

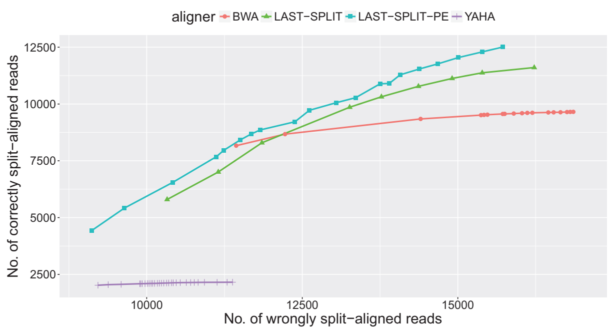

Fig. 3. Split-alignment accuracy. For each aligner, the curve is obtained by evaluating it at different error-probability/mapping quality thresholds. For LAST-SPLIT-PE, the mapping quality of the alignment of a flank is the highest column probability among the columns in that alignment

This also affords justification for our earlier definition of correct alignment.

So far in this section, we have not considered false positives, i.e. cases in which a read is mistakenly split-aligned when it should not be. We remedy this situation in the following evaluation.

3.3 Evaluation of the effect on variant calling

Using the same read set as before, we fed the alignments of *all* reads produced by each aligner to LUMPY, a method that makes structural-variant calls based on signals from discordant alignments of mates and split-alignment of single reads. We leave out LAST-SPLIT as its SAM-format output is not compatible with LUMPY. To contrast the strategy of LUMPY to that of DELLY, which first filters genomic intervals likely to contain structural variants based on discordant alignments of mates, we fed the same alignments from BWA to DELLY. The parameters used are documented in the Supplementary Material.

We evaluated the performance of the aligner-caller pairs based on their deletion calls. The set of Venter deletions $\geq$20 bp were set as positive examples (total of 13 495 instances). A true positive call is one that correctly identifies the start and end coordinates of a true deletion, allowing for some shifts in the predicted positions, to account for the micro-homologies that are present at deletion breakpoints. The amount of shift allowed is specific to each deletion and is set as follows. Let l (similarly, r) be the length of the longest common suffix (prefix) between the deleted sequence and the upstream (downstream) flank in the reference. For each true deletion with start and end coordinates s and e, we allow the predicted start and end coordinates to be in the interval $[s - l, s + r)$ and $(e - l, e + r]$, respectively.

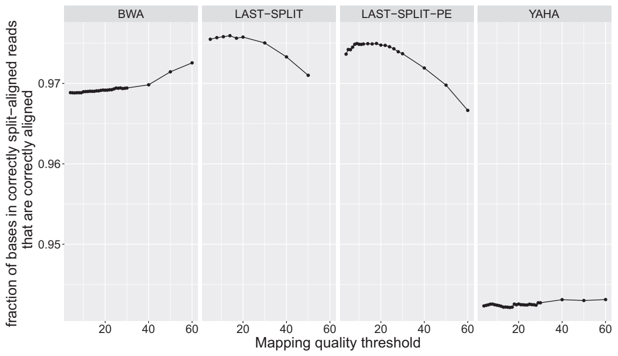

Fig. 4. Base-level precision of the alignments of reads classified as correctly split-aligned

The results are shown in Figure 5. Since variant calling is impacted by the fragment length characteristics, it is informative to divide the deletion calls into three groups based on their lengths: 20 bp to 1 × SD, 1 × SD to 3 × SD and larger than 3 × SD, where SD is the standard deviation of the fragment length distribution.

For the first and second groups, our method paired with LUMPY overwhelmingly outperforms the other methods, showing remarkable sensitivity while maintaining high precision. However, in the third group, the precision of our method degrades for less stringest operating points. The operating point shown as a larger dot corresponds to mapping quality threshold of 20 (i.e. error probability of 0.01); and judging from the three panels, a slightly stricter threshold (e.g. 30), seems to be a good default choice when running LAST-SPLIT-PE.

Since DELLY relies heavily on discordant paired-end reads, most of the BWA-DELLY calls are present in the last group. There it shows good specificity, but does not match the sensitivity of BWA-LUMPY and LAST-SPLIT-PE-LUMPY. It is intriguing how, given the same set of BWA alignments, LUMPY and DELLY behave differently in this group. For the BWA-LUMPY combination, using our split-alignment extraction script instead of the default, greatly improves the sensitivity of LUMPY for smaller variants (first and second groups). For the third group, the two BWA-LUMPY curves coincide, as expected.

A key observation—and a rather disappointing one—is that, of the >13 000 Venter deletions $\geq$ 20 bp, the LAST-SPLIT-PE-and-LUMPY combination can only find around 3300 of them, even at a setting that prioritizes sensitivity (mapping quality threshold of 5). This number is smaller for other aligner-variant-caller pairs. This is, however, consistent with our observation in Section 3.1 that only a third of reads from breakpoint regions could be correctly split-aligned. Also, this performance is much worse than the ideal-scenario empirical performance bound we presented in Section 1 (Fig. 1), where we showed that around 60% of deletions had ambiguity value of 1. Overall, this reconfirms the serious limitations of

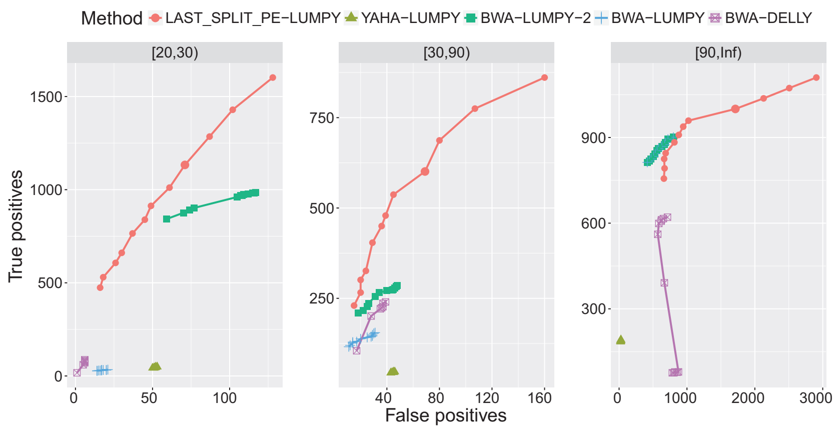

Fig. 5. Performance of aligner-and-variant-caller pairs. To reveal the impact of fragment size characteristics on the results, deletion calls have been grouped into size bins shown in the plot headers. For each aligner, a curve is obtained using the mapping quality value to filtering alignments that are passed to the variant caller. There are two curves for the BWA and LUMPY pair: BWA-LUMPY corresponding to the case of using LUMPY's default script for extracting split-alignments, which cannot catch small splits that are reported as a single-line SAM record, and BWA-LUMPY-2 corresponding to using ours, which resolves this issue

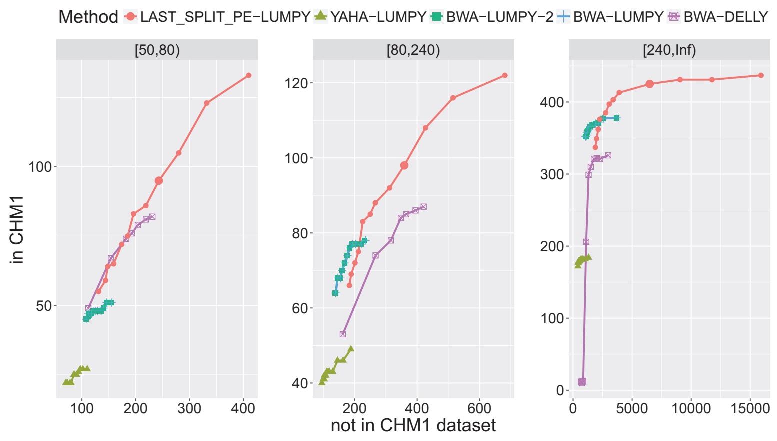

Fig. 6. Performance on real reads from the CHM1 cell line. The vertical (horizontal) axis is the number of calls that are (not) present in the CHM1 PacBio-based dataset. Deletion calls (≥20 bp) are grouped according to their size into 3 bins, shown in the plot header, 80 bp being the estimated standard deviation of the fragment size distribution of the SRR1514950 dataset

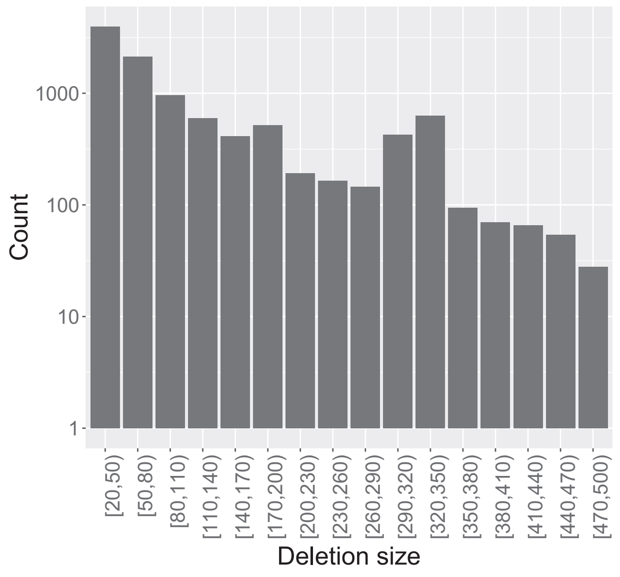

Fig. 7. Size distribution of deletions (≥20 bp) in the CHM1 dataset

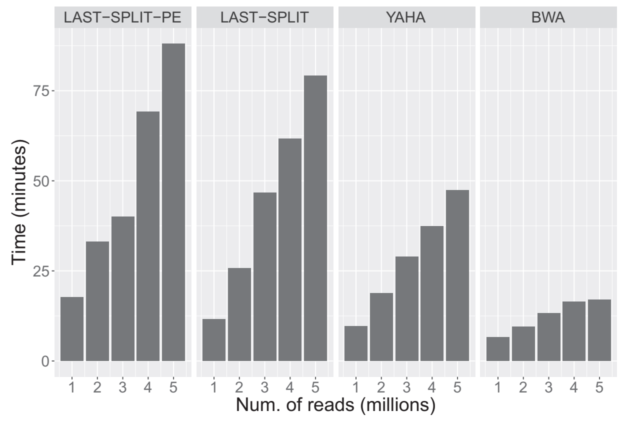

Fig. 8. Growth of running time for aligning 1–5 million 100 bp reads to the reference human genome. Multi-threading was turned off for all aligners. The runtimes of LAST-SPLIT-PE are for the whole process which includes computing local alignments and initial column probabilities using LAST, and the probability-updating procedure described in this work

short reads in detecting structural variants, even the supposedly simpler case of deletions.

3.4 Evaluation on a real dataset

Next we evaluated the performance of the aligner–caller combinations using real reads. The SRX652547 is a dataset containing short reads from the CHM1 cell line. For our evaluation, we used its subset, SRR1514950, which contains roughly 216 million pairs of 101 bp-long reads. For the same cell line, there exists a compilation of structural variants, which have been obtained from analyzing PacBio long reads (Chaisson *et al.*, 2015). As in the simulated case, we evaluate the methods on their deletion calls, using as positive examples all deletions ≥20 bp (totalling 11 273 deletions) from the PacBio-based study.

The results are shown in Figure 6, where we have again grouped the calls based on their sizes and the standard deviation of the fragment size distribution, which we estimated to be 80 bp. The first group however starts at 50 bp—this is to accommodate the fact that the CHM1 dataset is conspicuously depleted in deletions <50 bp (compare Fig. 7 with Fig. 2). We can observe similar trends as in the case of the simulated dataset, with LAST-SPLIT-PE-LUMPY exhibiting significantly higher sensitivity than other methods.

3.5 Running time and memory usage

Figure 8 shows the running time of the aligners. All methods scale linearly with input size. Comparing the running time of LAST-SPLIT-PE and LAST-SPLIT, we can see that our paired-end computation takes only a small amount of extra time. While the running time of LAST-SPLIT-PE is slower than BWA, we provide multi-threading option to handle large datasets within reasonable time bounds—e.g. using 16 threads, we can align reads amounting to coverage-40 of the whole human genome, in under a day. The local alignment phase can be made faster—at the cost of sensitivity—by adjusting parameters such as the seed search step size parameter k (Supplementary Material).

The memory usage of each tool is dominated by the size of its reference index. LAST-SPLIT and LAST-SPLIT-PE build a suffix array, BWA constructs an index based on the Burrows-Wheeler transform, and YAHA builds a lookup table for *k*-mers. Consequently, for a human reference genome, memory usage amounts to 20 GB for LAST-SPLIT and LAST-SPLIT-PE, 5.2 GB for BWA, and 12 GB with $k = 11$ for YAHA.

4 Discussion

We have proposed a split-alignment method that combines alignment column probabilities obtained from probabilistic alignment techniques, with information from read pairs from paired-end sequencing. We demonstrated that our method produces more accurate split-alignments than existing methods, and as a consequence leads to more accurate identification of large variants from whole genome DNA-sequencing reads.

Regarding the generality of our tool, while this work describes its usage in conjunction with LAST, the proposed framework can be applied to any alignment tool that can report alignment column probabilities. If an aligner reports high-scoring local alignments, but does not provide column probabilities, it might still be theoretically possible to estimate them using the method described in (Frith and Kawaguchi, 2015). Also while we have based our evaluation on finding large deletions, our method is applicable to other kinds of structural variants such as chromosomal translocations or inversions.

Split-alignments arise in scenarios other than DNA read alignment. Analysis of RNA-seq data often require aligning short reads that span splicing junctions. It was recently reported that most RNA-seq alignment tools do not perform well in the task of identifying splicing junctions when the dataset is challenging (Baruzzo *et al.*, 2017). Applying our idea might be one way to improve prediction accuracy. However since in the case of RNA-seq aligners, other sources of information such as splicing signals and genomic annotation are also important, we do not investigate this issue in this article.

Several experimental techniques that use high-throughput sequencers to investigate RNA structure and interaction have been proposed recently (Aw *et al.*, 2016; Ramani *et al.*, 2015). A common theme among these methods is that they generate chimeric reads which are signals of interacting RNA bases. The analysis pipeline for such data requires first that the read be split-aligned to the genome or transcriptome. This might be another interesting area to apply our method.

As we showed in Figure 1 and as indicated by our benchmarking results, a major concern of short reads is that their length imposes a serious limitation in the ability to identify simple deletions, let alone complex rearrangements. In this regard, the recent emergence of long-read technologies, such as Oxford NanoPore and PacBio, is a promising development. However, it seems they are still far from being adapted as widely as short-read technology—for instance, of the entries in the DDBJ DRA archive, the number of Illumina-based studies is many orders larger than the number of those using PacBio or Oxford Nanopore. For the time-being, short reads will perhaps continue to be the mainstay for a wide range of sequencing-based studies, and therefore every measure to improve alignment accuracy by using supplemental information is important.

Acknowledgements

We thank Martin Frith for multiple rounds of insightful discussions throughout this project, Kentaro Mizutani for helping set up the computational environment, and Tony Kuo, Risa Kagawuchi, and Hugues Richard for helpful comments on the article.

Funding

This work was partly supported by JSPS KAKENHI [grant number 16H06279].

Conflict of Interest: none declared.

References

Aw,J.G. *et al.* (2016) In vivo mapping of eukaryotic RNA interactomes reveals principles of higher-order organization and regulation. *Mol. Cell*, **62**, 603–617.

Baruzzo,G. *et al.* (2017) Simulation-based comprehensive benchmarking of rna-seq aligners. *Nat. Methods*, **14**, 135–139.

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

Durbin,R. *et al.* (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, Cambridge University Press.

Faust,G.G. and Hall,I.M. (2012) Yaha: fast and flexible long-read alignment with optimal breakpoint detection. *Bioinformatics*, **28**, 2417–2424.

Frith,M.C. and Kawaguchi,R. (2015) Split-alignment of genomes finds orthologies more accurately. *Genome Biol.*, **16**, 106.

Layer,R.M. *et al.* (2014) Lumpy: a probabilistic framework for structural variant discovery. *Genome Biol.*, **15**, R84.

Levy,S. *et al.* (2007) The diploid genome sequence of an individual human. *PLoS Biol.*, **5**, e254.

Ramani,V. *et al.* (2015) High-throughput determination of RNA structure by proximity ligation. *Nat. Biotechnol.*, **33**, 980–984.

Rausch,T. *et al.* (2012) Delly: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, **28**, i333.

Zhao,H. and Zhao,F. (2015) Breakseek: a breakpoint-based algorithm for full spectral range indel detection. *Nucleic Acids Res.*, **43**, 6701–6713.