frontiers in Genetics

METHODS
published: 29 July 2020
doi: 10.3389/fgene.2020.00632

LROD: An Overlap Detection Algorithm for Long Reads Based on *k*-mer Distribution

*Junwei Luo [1], Ranran Chen [1], Xiaohong Zhang [1], Yan Wang [2], Huimin Luo [2], Chaokun Yan [2] and Zhanqiang Huo [1]**

[1] College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, China, [2] School of Computer and Information Engineering, Henan University, Kaifeng, China

Third-generation sequencing technologies can produce large numbers of long reads, which have been widely used in many fields. When using long reads for genome assembly, overlap detection between any pair of long reads is an important step. However, the sequencing error rate of third-generation sequencing technologies is very high, and obtaining accurate overlap detection results is still a challenging task. In this study, we present a long-read overlap detection (LROD) algorithm that can improve the accuracy of overlap detection results. To detect overlaps between two long reads, LROD first retains only the solid common *k*-mers between them. These *k*-mers can simplify the process of overlap detection. Second, LROD finds a chain (i.e., candidate overlap) that includes the consistent common *k*-mers. In this step, LROD proposes a two-stage strategy to evaluate whether two common *k*-mers are consistent. Finally, LROD uses a novel strategy to determine whether the candidate overlaps are true and to revise them. To verify the performance of LROD, three simulated and three real long-read datasets are used in the experiments. Compared with two other popular methods (MHAP and Minimap2), LROD can achieve good performance in terms of the F1-score, precision and recall. LROD is available from https://github.com/luojunwei/LROD.

Keywords: overlap detection, alignment, long read, *k*-mer distribution, the third generation sequencing technology

OPEN ACCESS

Edited by:
Jianxin Wang,
Central South University, China

Reviewed by:
Liang Zhao,
Hubei University of Medicine, China
Arjun Arkal Rao,
University of California, Santa Cruz, United States

****Correspondence:***
Zhanqiang Huo
hzq@hpu.edu.cn

Specialty section:
This article was submitted to Computational Genomics, a section of the journal Frontiers in Genetics

Received: *04 December 2019*
Accepted: *26 May 2020*
Published: *29 July 2020*

Citation:
Luo J, Chen R, Zhang X, Wang Y, Luo H, Yan C and Huo Z (2020) LROD: An Overlap Detection Algorithm for Long Reads Based on k-mer Distribution. Front. Genet. 11:632. doi: 10.3389/fgene.2020.00632

INTRODUCTION

Sequencing technologies fragment the genome into a large number of reads, and the process of recombining these reads into a complete DNA sequence is called genome assembly (Nagarajan and Pop, 2013; Ding and Guo, 2018). Read sequencing from next-generation sequencing (NGS) technology (Miller et al., 2010), is usually short, i.e., only a few hundred base pairs in length. Short reads commonly cannot be used to solve problems caused by long repetitive regions (Liao et al., 2020). In addition, NGS polymers commonly lead to some GC bias, which will affect the correctness of the genome assembly (Farrer et al., 2009; Luo et al., 2012). Compared with NGS, third-generation sequencing (TGS) technologies (Schadt et al., 2010), such as single-molecule real-time technology (SMRT) (Levene et al., 2003) and Oxford Nanopore technology (ONT) (Stoddart et al., 2009), can produce longer reads with an average length of 10 kb, with many exceeding 100 kb. This long-read length is sufficient to span most repetitive areas. TGS does not require any polymerase chain reaction process and therefore can avoid GC bias (Ross et al., 2013). It is worth noting that TGS has a higher sequencing random error rate than that of NGS technology. For instance, the sequencing error rate of SMRT is ∼15%, and the sequencing error rate of ONT can reach 20%. Fortunately, current studies have shown that high sequencing coverage can correct these random errors. Hence, these long reads are helpful for solving problems caused by sequencing bias and repetitive regions (Ummat and Bashir, 2014; Luo et al., 2016). Currently, many assemblers using long reads have been presented.

Overlap detection is usually the first step of assembly algorithms (Pevzner et al., 2001). For two long reads, the purpose of overlap detection is to determine whether overlap exists. If two long reads have an overlap, overlap detection tools will highlight the two regions from the two long reads separately, which can then be overlapped (Luo et al., 2015a,b).

To identify the overlaps among long reads, two major problems need to be addressed: (1) High sequencing error rate: the long reads from TGS always have high sequencing error rates. Therefore, it is difficult to obtain accurate overlap results. (2) Repetitive regions: long repetitive regions complicate the process of overlap detection. These two problems in the process of overlap detection should attract more attention.

At present, many overlap detection tools that are capable of detecting overlaps among error-prone long reads with different accuracy levels have been developed (Luo et al., 2020).

BLASR (Chaisson and Tesler, 2012) is designed to align long reads to a genome reference and can be used to detect overlaps among long reads. The performance of BLASR depends heavily on the parameter values. For instance, to achieve higher sensitivity, BLASR first needs to know the coverage of the long reads and then chooses the appropriate nBest and nCandidates (*n* is 10 by default) values, which should not be less than the coverage. The running time is also related to the two parameters. BLASR adopts complete aligning and can necessitate more computing resources than are required for downstream processes. DALIGNER (Myers, 2014) is a tool especially designed to detect overlaps among long reads. This method focuses on optimizing the running efficiency in response to the relatively poor running performance of constructing the FM-index suffix array/tree data structure (Ferragina and Manzini, 2005). Its implementation steps are as follows: (a) split long reads into blocks; (b) sort the *k*-mers in each block; and (c) merge the blocks. DALIGNER greatly improves the running efficiency. To increase speed and reduce memory usage, DALIGNER filters out some k-mers. MHAP (Berlin et al., 2015) is a MinHash algorithm (Broder, 1997) that relies on *k*-mer similarity to implement overlap detection for long reads. MHAP first indexes the *k*-mers with multiple hash functions. For all *k*-mers in the two long reads, MHAP builds a sketch list with the minimum value of the hash function and then finds the location of the overlap. Next, MHAP uses the shorter *k*-mers to repeat the previous steps to discover a more accurate overlap. The number of hash functions used by MHAP to build a sketch is always fixed. However, the lengths of the long reads are not equal, and their sensitivity is affected when the length of the reads varies widely. For shorter reads, MHAP inevitably wastes some memory. For longer reads, accurate overlap detection is difficult to achieve because too few *k*-mers exist. Minimap2 (Li, 2016, 2018) is an overlap detection tool, that applies the idea of the sketch from MHAP but uses minimizers as a simplified representation instead. Similar to MHAP, Minimap2 saves *k*-mers in a hash table. In addition, Minimap2 adopts a sorting strategy inspired by DALIGNER to improve the running efficiency.

In this paper, we develop an approach named long-read overlap detection (LROD) to detect overlaps among long reads based on the *k*-mer distribution. The main contributions of LROD are the following. To address the problem caused by sequencing errors and repetitive regions, for two long reads, LROD first finds all solid common *k*-mers between them. Solid *k*-mers might not include sequencing errors and may come from repetitive regions, which helps to simplify the process of overlap detection. Then, LROD employs a two-stage strategy to determine whether two common *k*-mers are consistent, i.e., whether the middle regions between them overlap. Then, LROD finds a chain that comprises consistent common *k*-mers, which indicates a candidate overlap. Finally, LROD revises the candidate overlap and utilizes a new evaluation method to determine whether the candidate overlap is true. The experimental results demonstrate that LROD performs better than MHAP and Minimap2 in terms of the F1-score.

MATERIALS AND METHODS

In this paper, LROD is used to detect the overlaps among long reads based on the distribution of *k*-mers. One *k*-mer is a substring with a length *k* in long reads. Suppose that the length of a long read is *L*; then, the number of *k*-mers in the long read is $(L - k + 1)$. LROD first identifies solid *k*-mers and keeps them, removing all other *k*-mers. Then, LROD constructs a *k*-mer hash table. For two long reads, LROD uses Algorithm 1 to detect any overlap between them. As shown in Algorithm 1, LROD first finds the common *k*-mer set (CKS) between two long reads R_1 and R_2. Second, based on the CKS, LROD attempts to search a chain for consistent common *k*-mers, which correspond to a candidate overlap. Third, LROD further evaluates the candidate and determines the final overlap. In the following sections, we describe each step in detail.

Selecting Solid *k*-mers

For two long reads, LROD utilizes the common *k*-mers between them to determine whether they overlap. However, the high sequencing error rate of TGS usually leads to negative common *k*-mers, and repetitive regions can cause a position contradiction among common *k*-mers. For a long-read dataset, one *k*-mer with a small frequency commonly includes sequencing errors, whereas, one *k*-mer with a large frequency usually originates from a repetitive region (Liu et al., 2013). Hence, LROD selects only *k*-mers whose frequencies are in the interval $[f_{\min}, f_{\max}]$ as solid *k*-mers, where $f_{\min}$ and $f_{\max}$ are two thresholds that

Abbreviations: A, adenine; T, thymine; C, cytosine; G, guanine; NGS, next generation sequencing; PCR, polymerase chain re-action; SMRT, single molecule sequencing; PB, pacific biosciences; SMRT, single molecule real time; ONT, oxford nanopore technology; OLC, overlap-layout-consensus; MHAP, MinHash alignment process; BLAST, basic local alignment search tool; BLAT, the BLAST like alignment tool; BLASR, basic local alignment with successive refinement; SSE, streaming SIMD extensions.

are calculated by LROD. Using only solid k-mers allows LROD to avoid some problems caused by sequencing errors and repetitive regions.

Before calculating the values of $f_{\min}$ and $f_{\max}$, LROD should determine the value of k. A larger value of k helps to resolve repetition-related problems, but decreases the number of common k-mers between two long reads. A smaller value of k will introduce more negative common k-mers, and complicate the process of overlap detection. LROD sets k to 15 by default. Next, LROD uses the method described below to select solid k-mers.

For a long-read dataset, LROD first uses DSK (Rizk et al., 2013), a k-mer counting program, to calculate the frequency of each k-mer in the dataset. If the frequency of a k-mer is one, then only one read contains this k-mer, which means that it is useless for finding any overlap between two long reads. For LROD, the minimum frequency of the k-mer is 2, that is, $f_{\min} = 2$ by default. This threshold can filter out a large number of k-mers that might include sequencing errors.

The value of $f_{\max}$ should be determined based on the coverage of the long-read set and the characteristics of the genome. If $f_{\max}$ is high, some k-mers from repetitive regions may be kept in the following steps. If $f_{\max}$ is low, some k-mers that do not originate from repetitive regions may be ignored. LROD develops a method to calculate $f_{\max}$ based on the frequency of k-mers. $F(x)$ refers to the number of k-mers whose frequency is x, $x = 1, 2, 3\ldots, h$, where h is the maximum k-mer frequency. For example, a k-mer set {AAT, ATA, TAG, TAG, AGT, ATA, AAT, AGT, AGT} exists. For this k-mer set, no k-mer appears once, and thus, $F(1) = 0$. $F(2) = 3$, which means that three k-mers appear twice, i.e., "AAT, ATA, TAG." $F(3) = 1$, which means that only one k-mer, i.e., "AGT," is repeated three times.

Then, $S(y)$ is used to calculate the cumulative sum of $F(x)$, as described in equation 1. When f is the smallest value such that $S(f) > \theta^* S(h)$, we set $f_{\max} = f$, and $\theta = 0.9$ by default, and $S(h)$ is the total frequency of k-mers whose frequencies are not smaller than 2.

$$S(y) = \sum_{x=f_{\min}}^{y} F(x) \tag{1}$$

After determining the interval $[f_{\min}, f_{\max}]$, the k-mers whose frequencies are not in this interval are ignored in subsequent steps. Overlap detection using only solid k-mers can minimize the impacts of sequencing errors and repetitive regions, and improve the accuracy of results.

The remaining solid k-mers are indexed by using a k-mer hash table with the k-mers as keys. For a specific k-mer, the k-mer hash table enables LROD to quickly identify the long reads that include it. At the same time, LROD can find locations and orientations in these long reads. As a result, LROD can quickly find a CKS for two long reads based on the k-mer hash table.

Detecting Overlap Between Two Long Reads

LROD selects two long reads R_1 and R_2 to detect whether they overlap. If they overlap, LROD will give the region in R_1 that overlaps with another region in R_2. The process of overlap detection for R_1 and R_2 is described below.

Algorithm 1: Finding_overlap_region (R_1, R_2, k, k_s, α, β, γ, ε)

```
Input: two long reads
Output: determines the final overlap
Begin
    Finding the common k-mer set CKS between $R_1$ and $R_2$;
    Removing forward or reverse common k-mers from CKS;
    Sorting common k-mers in CKS;
    $m \leftarrow |CKS|$;
    $count \leftarrow 5$;
    if $m < count$ then
        return NULL;
    end if
    $i \leftarrow 0$;
    while $i < m$ do
        if CKS[$i$] is not visited then
            CKS[$i$] is visited;
            chain ← Chaining_from_start (CKS, $i$, $k$, $k_s$, $\alpha, \beta, \gamma$);
            if chain != NULL then
                all common k-mers in the chain are visited;
                region ← Evaluate_candidate overlap_region (CKS, chain, $\alpha, \gamma, \varepsilon$);
                if region != NULL then
                    return region;
                else
                    $i$++;
                end if
            else
                $i$++;
            end if
        end if
    end while
    return NULL;
End
```

Finding a Common k-mer Set Between R_1 and R_2

First, LROD extracts k-mers from R_1 with a step (s, 1 in default), and selects k-mers which appear in R_2 through the k-mer hash table. Then, LROD gets a common k-mer set (CKS). If one k-mer in a long read appears twice or more in another long read, LROD deletes it from CKS. The i-th common k-mer in CKS is represented by a four-tuples $(P_{1i}, O_{1i}, P_{2i}, O_{2i})$. P_{1i} and P_{2i} are the starting positions of the common k-mer in R_1 and R_2 respectively. O_{1i} and O_{2i} are the orientations of the common k-mer in R_1 and R_2, respectively. If the i-th common k-mer has the same orientation ($O_{1i} = O_{2i}$), the common k-mer is a positive common k-mer, otherwise, it is an opposite common k-mer. LROD uses M to represent the number of positive common k-mers, and N to indicate the number of opposite common k-mers. If $M > N$ and $M > count$ ($count = 5$), LROD keeps the positive common k-mers and ignores the opposite common k-mers. If $N > M$ and

Algorithm 2: Chaining_from_start (CKS, start, k, k_s, α,β, γ)

```
Input: the starting common k-mer
Output: find a chain which consists of some consistent
  common k-mers
Begin
    $m \leftarrow |CKS|$;
    end ← start + 1;
    chain ← NULL;
    Adding CKS[start] to the chain;
    while start $<$ $m$ and end $<=$ $m$ do
        result ← Determine_consistent (CKS[start],
        CKS[end], $k$, $k_s$, $\alpha$,$\beta$, $\gamma$);
        if result != true then
            end++;
            continue;
        end if
        Adding CKS[end] to the chain;
        start ← end;
        end ← end + 1;
    end while
    if |chain| > 2 then
        return chain;
    else
        return NULL;
    end if
End
```

Algorithm 3: Determine_consistent (CKS[start], CKS[end], k, k_s, α,β,γ)

```
Input: common k-mers
Output: whether two common k-mers are consistent
Begin:
    if determine_consistent_1(CKS[start], CKS[end],
        $\alpha$, $\gamma$) != true then
        return determine_consistent_2(CKS[$i$], CKS[$j$],
        $k$, $k_s$, β);
    endif
    return true;
End
```

Algorithm 4: Determine_consistent_1 (CKS[start], CKS[end], k, k_s, α,γ)

```
Input: common k-mers
Output: whether two common k-mers are consistent
Begin
    Getting the positions of the start and end common
  k-mers:
    $P_{1i}, P_{2i}, P_{1j}, P_{2j}$ ($P_{1i} < P_{1j}$);
    $D_1 \leftarrow |P_{1j} - P_{1i}|$;
    $D_2 \leftarrow |P_{2j} - P_{2i}|$;
    if forward common k-mers then
        if ($P_{1i} < P_{1j}$ and $P_{2i} < P_{2j}$) and ($D_1 < \alpha$ && $D_2$
        $< \alpha$) and((Max($D_1$, $D_2$) – Min($D_1$,$D_2$)) /
        Max($D_1$,$D_2$) $< \gamma$) then
            return true;
        else
            return false;
        end if
    end if
    if reverse common k-mers then
        if ($P_{1i} < P_{1j}$ and $P_{2i} > P_{2j}$) and ($D_1 < \alpha$ and
        $D_2 < \alpha$) and ((Max($D_1$, $D_2$) – Min($D_1$,$D_2$))
        / Max($D_1$,$D_2$) $< \gamma$) then
            return true;
        else
            return false;
        end if
    end if
End
```

$N > count$, LROD keeps the opposite k-mers and ignores the positive k-mers. The remaining common k-mers in the CKS are sorted in ascending order based on their positions in R_1. If R_1 and R_2 do not satisfy any of the above two conditions, LROD concludes that they do not have an overlap and processes another pair of long reads.

Chaining

In this step, LROD aims to find a chain from the CKS that consists of some consistent common k-mers, and corresponds to a candidate overlap. Algorithm 2 shows the pseudocode of chaining. First, the starting common k-mer is added to the chain. Then, LROD searches for the first subsequent common k-mer, which is consistent with the previous common k-mer in the chain. When one consistent common k-mer is found, it is appended to the chain. LROD repeats this process until all common k-mers are visited. Finally, LROD obtains a chain.

The most important issue in finding a chain is how to decide whether two common k-mers are consistent. For two common k-mers, their distances in the two long reads can be calculated. When the two distances are large or differ too much, the two common k-mers might be inconsistent. LROD employs a two-stage strategy for this issue, an example of which is shown in **Figure 1**. Algorithms 3–5 show the pseudocode for determining whether two common k-mers are consistent. In the first stage as shown in Algorithm 4, LROD presents some conditions to evaluate. If they cannot be determined in the first stage, LROD uses the second stage as shown in Algorithm 5 to further analyse whether they are consistent based on smaller k_s-mers ($k_s < k$).

For two common k-mers (P_{1i}, O_{1i}, P_{2i}, O_{2i}) and (P_{1j}, O_{1j}, P_{2j}, O_{2j}), two distances $D_1 = |P_{1j} - P_{1i}|$ and $D_2 = |P_{2j} - P_{2i}|$ can be calculated. In Algorithm 4, LROD uses C1, C2, C3, and C4 to evaluate their consistency. The four conditions are listed below. C1 means that they are forward, while C2 means the reverse. Due to the high sequencing error rate of TGS, we should allow a distance between two consecutive consistent common k-mers. However, the larger the distance is, the greater the number of sequencing errors that exist. Hence, C3 specifies the distance threshold α between two consistent common k-mers. However, α is difficult to determine. A small α will miss some consistent

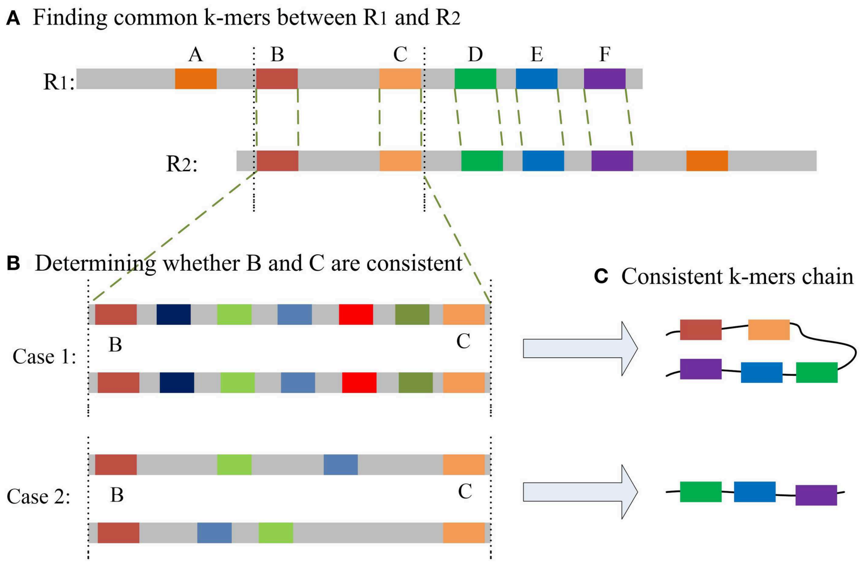
A Finding common k-mers between R1 and R2
A
B
C
D
E
F
R1:
R2:
B Determining whether B and C are consistent
Case 1:
B
C
Case 2:
B
C
C Consistent k-mers chain

FIGURE 1 | (A) A–F are the common k-mers in the two long reads, D–F is the common consistent k-mer that can be judged in the first stage. The distance between B and C is slightly longer, but less than β. **(B)** Aligning sequences between B and C with smaller k_s-mers in the second stage may result in two situations. Case 1: Multiple k_s-mers can be compared between B and C; Case2: there are no common consistent k_s-mers between B and C. **(C)** According to different situations, two different consistent chains will be generated.

common k-mers, and a large α will accept more inconsistent common k-mers. In this stage, LROD adopts a small value of α (400 by default) to select consistent common k-mers with high confidence. C4 specifies the maximum difference between D_1 and D_2 (γ =0.3). When the two common k-mers satisfy the conditions, LROD considers them to be consistent.

C1: $P_{1i} < P_{1j}$ and $P_{2i} < P_{2j}$;

C2: $P_{1i} < P_{1j}$ and $P_{2i} > P_{2j}$;

C3: $D_1 < \alpha$ and $D_2 < \alpha$;

C4: $(\mathrm{Max}(D_1, D_2) - \mathrm{Min}(D_1,D_2)) / \mathrm{Max}(D_1,D_2) < \gamma$

If Algorithm 4 returns false, LROD will utilize Algorithm 5 to further evaluate the consistency between the two common k-mers. In Algorithm 5, LROD adopts a large β (1,500 by default), which provides more candidate common k-mers. To identify correct consistent common k-mers, LROD finds small k_s-mers ($k_s < k$) from two regions in R_1 and R_2 between these two common k-mers. If two common k_s-mers satisfy C3 and C4, they will be linked. If LROD can find a path from the starting common k_s-mer to the ending common k_s-mer, then Algorithm 5 returns true.

After obtaining the chain, the number of common k-mers in the chain should be larger than 2. Finally, if the common k-mers in the chain are positive, LROD concludes that R_1 and R_2 possibly come from the same strand. Otherwise, they might come from reverse strands. Moreover, LROD obtains two draft overlaps $[P_1, P_n + k]$ and $[Q_1, Q_n + k]$ on R_1 and R_2, respectively, where P_1 and Q_1 are the first k-mers in the chain, and P_n and P_n are the last k-mers in the chain.

Determining the Final Overlap

Due to sequencing errors, the above candidate overlap may deviate somewhat from the real overlap. Suppose the true overlap on R_1 is $[SP_1, EP_1]$, and the true overlap on R_2 is $[SP_2, EP_2]$. The lengths of R_1 and R_2 are Len_1 and Len_2, respectively. LROD uses the following method to revise the candidate overlap and obtain the true overlap for R_1 and R_2. An example is shown in **Figure 2**.

(1) If $P_1 > Q_1$ and $Len_1 - P_n <= Len_2 - Q_n$, $SP_1 = P_1 - Q_1$, $EP_1 = Len_1$; $SP_2 = 1$, $EP_2 = Q_n + Len_1 - P_{n+1}$, as shown in **Figure 1A**.

(2) If $P_1 < Q_1$ and $Len_1 - P_n <= Len_2 - Q_n$, $SP_1 = 1$, $EP_1 = Len_1$; $SP_2 = Q_1 - P_1$, $EP_2 = Q_n + Len_1 - P_{n+1}$, as shown in **Figure 1B**.

(3) If $P_1 > Q_1$ and $Len_1 - P_n > Len_2 - Q_n$, $SP_1 = P_1 - Q_1$, $EP_1 = P_n + Len_2 - Q_n$; $SP_2 = 1$, $EP_2 = Len_2$.

(4) If $P_1 < Q_1$ and $Len_1 - P_n > Len_2 - Q_n$, $SP_1 = 1$, $EP_1 = P_n + Len_2 - Q_n$; $SP_2 = Q_1 - P_1$, $EP_2 = Len_2$.

After the above processing, the real overlap on R_1 and R_2 can be obtained.

As shown in **Figure 2**, the length of the overlap on R_1 is $OverlapLenR_1 = EP_1 - SP_1$, and the length of the overlap on R_2 is $OverlapLenR_2 = EP_2 - SP_2$. We use $MaxOverlapLen$ and $MinOverlapLen$ to represent the maximum overlap length and the minimum overlap length, respectively. $MaxOverlapLen = \max(OverlapLenR_1, OverlapLenR_2)$, and $MinOverlapLen = \min(OverlapLenR_1, OverlapLenR_2)$.

When R_1 and R_2 satisfy the following three conditions, LROD considers R_1 and R_2 to have an overlap. The overlaps are $[SP_1, EP_1]$ and $[SP_2, EP_2]$ on R_1 and R_2, respectively. Otherwise, no overlap exists between them. ε is the threshold of the overlap length (500 by default).

(1) $(EP_1 - SP_1) - (P_n + k - P_1) < \alpha$ and $(EP_2 - SP_2) - (Q_n + k - Q_1) < \alpha$;

(2) $MinOverlapLen > \varepsilon$;

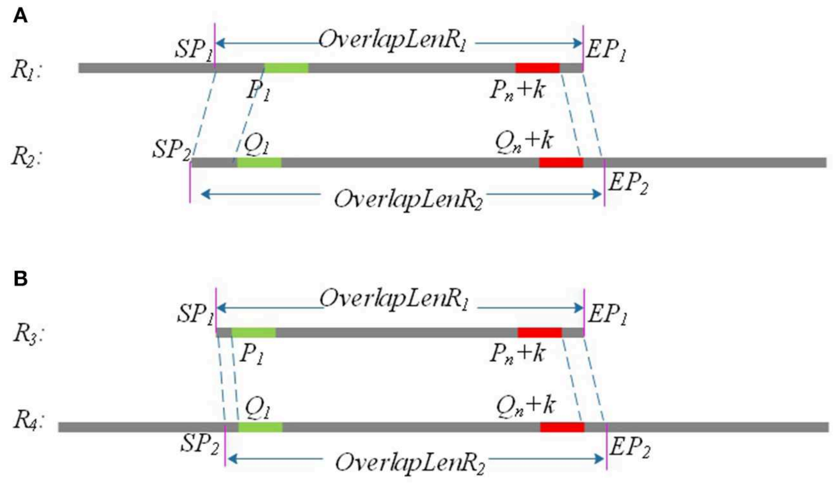

FIGURE 2 | P_1 and Q_1 are the start positions of the common positive k-mer on R_1 and R_2, and ($P_n + k$) and ($Q_n + k$) are the end positions of the common positive k-mer on R_1 and R_2. SP_1 and EP_1 are the start position and end position of the final overlap on R_1, respectively, and SP_2 and EP_2 are the start position and end position of the final overlap on R_2. **(A)** Partial overlap: The right end of one long read aligning with the left end of the other long read. **(B)** Full overlap: A long read is completely aligned to a part of the other long read.

```
Algorithm 5: Determine_consistent _2 (CKS[start], CKS[end], $k$, $k_s$, $\beta$)
  Input: common $k$-mers
  Output: whether two common $k$-mers are consistent
  Begin:
      Getting the positions of the start and end common k-mers:
      $P_{1i}$, $P_{2i}$, $P_{1j}$, $P_{2j}$ ($P_{1i} < P_{1j}$);
      $D_1 \leftarrow |P_{1j} - P_{1i}|$ and $D_2 \leftarrow |P_{2j} - P_{2i}|$;
      if $D_1 > \beta$ or $D_2 > \beta$ then
          return false;
      end if
      For forward common $k$-mers, find the common $k_s$-mer set between $[P_{1i} + k - k_s, P_{1j} + k_s]$ in $R_1$ and $[P_{2i} + k - k_s, P_{2j} + k_s]$ in $R_2$. For reverse common $k$-mers, find the common $k_s$-mer set between $[P_{1i} + k - k_s, P_{1j} + k_s]$ in $R_1$ and $[P_{2j} - k_s, P_{2i} - k + k_s]$ in $R_2$.
      if there is a chain which starting from the starting point to the ending point then
          return true;
      else
          return false;
      end if
  End
```

(3) $(MaxOverlapLen - MinOverlapLen) / MaxOverlapLen < \gamma$.

Parameters of LROD

LROD has nine tunable parameters that can affect the final experimental results. These parameters can be categorized into three groups. The first group contains five parameters, namely, $f_{\min}$, θ, s, k, and k_s, which are used to determine the common k-mers between two long reads. More solid k-mers can be retained by increasing the value of $f_{\min}$ and reducing the value of θ, though doing so will also increase the number of false solid k-mers. We assign 1 as the default value for parameter s, which can retain all solid k-mers and find all common k-mers between two long reads. Generally, the detection of overlaps based on short k-mers ($k = 9$) is sensitive, while Minimap2 sets $k = 15$ to detect overlaps. Therefore, we set $k = 15$ and $k_s = 9$ by default to balance the sensitivity and specificity. Specifically, the memory requirement and running time of LROD decrease when the values of these parameters increase. The second group consists of three parameters, represented as α, β, and γ, which are utilized to evaluate whether two common k-mers are consistent. Minimap2 uses a 500 bp band-width to find collinear minimizers. Here, we set the default α to 400 to select consistent common k-mers with high confidence. Moreover, we use $\beta = 1{,}500$ and k_s-mers ($k_s < k$) to further determine whether they are consistent. Essentially, if two common k-mers are consistent, the two distances between them should be similar. Thus, when the difference between the two distances is larger than γ (0.3 by default), they are regarded as inconsistent by LROD. The values of these parameters will need to be changed according to the sequencing error rate of TGS. If the error rate is low, reducing the values of these parameters could improve the precision of the result. The last group contains parameter ε. LROD outputs only the overlaps whose lengths are larger than ε.

To further examine the impact of these parameters on the results of overlap detection, we conducted LROD with different values of four parameters: α, β, γ, and θ. The results are shown in the **Supplementary Material**.

RESULTS AND DISCUSSION

To verify the effectiveness of the proposed method in this paper, we used three simulated and three real datasets to benchmark LROD, MHAP and Minimap2; the performance was verified

Algorithm 6: Evaluate_candidate_overlap_region(CKS, chain, α, γ, ε)

```
Input: CKS, chain, $\alpha$, $\gamma$, ε
Output: True or False
Begin
        Getting draft overlap based on chain: $[SP_1, EP_1]$
        and $[SP_2, EP_2]$;
        $OverlapLenR_1 \leftarrow EP_1 - SP_1$;
        $OverlapLenR_2 \leftarrow EP_2 - SP_2$;
        $MaxOverlapLen \leftarrow \max(OverlapLenR_1, OverlapLenR_2)$;
        $MinOverlapLen \leftarrow \min(OverlapLenR_1, OverlapLenR_2)$;
        if $(EP_1 - SP_1) - (P_n + k - P_1) < \alpha$ and $(EP_2 - SP_2) - (Q_n + k - Q_1) < \alpha$
            and $MinOverlapLen > \varepsilon$ and
            $(MaxOverlapLen - MinOverlapLen) / MaxOverlapLen < \gamma$ then
            return true;
        end if
        return false;
End
```

TABLE 1 | Details of datasets.

Datasets	Genomic length (Mbp)	Average length of reads(bp)	Number of read	Coverage
E. coli-10	~4.6	6,555	6,955	~10
E. coli-20	~4.6	6,619	13,911	~20
chr20-10	~6.4	6,621	96,574	~10
E. coli_Real	~4.6	4,185	6,972	~7
C. elegans_Real	~99.9	4,091	188,559	~77
Human_Real	~3,157	25,890	461,247	~3.78

with k-mer lengths of $k = 13$ and $k = 15$, respectively. The three real datasets are from genomes of *Escherichia coli* (*E. coli*), *Caenorhabditis elegans* (*C. elegans*), and *humans*, which were sequenced by SMRTs. The real datasets related to *E. coli* and *C. elegans* are available from schatzlab.cshl.edu/data/ectools/. The real human dataset is NA20300 (SRR9683669). The three real datasets are herein referred to as *E. coli_Real*, *C. elegans_Real,* and *Human_Real*. In this paper, we used SURVIVOR (Jeffares et al., 2017) to obtain three simulated datasets: 10X coverage *E. coli* (*E. coli-10*), 20X coverage *E. coli* (*E. coli-20*), and 10X coverage human chromosome 20 (*chr20-10*). For these long-read datasets, we retained the long reads whose lengths were longer than 2,000 bp in the following experiments. **Table 1** shows the details of the long-read datasets, including the genomic length, average length of reads, number of reads, and coverage.

For the three simulated datasets, we can directly obtain the real overlaps among the long reads. For the three real datasets, we used BLASR to align these long reads against the reference genomes. We retained only those reads whose aligning quality was >85%, and the long reads were completely aligned on the genome reference. Then, we were able to acquire real overlaps among these long reads based on their alignment positions. The obtained overlaps were used to evaluate the performance of the overlap detection tools. All tools were run with 10 threads on a computer with 128 GB of memory. During the experiments, the wall time of LROD can be reduced by adopting a larger thread number.

Results

For the real human dataset, when $k = 13$, Minimap2 and LROD did not end with 10 threads after 10 days, and the memory requirement of MHAP was larger than the memory capacity of our computer (128 GB). Therefore, we do not give the results for the human dataset with $k = 13$.

As shown in **Table 2** and **Figure 3**, LROD obtained satisfactory results for these datasets. Especially in the simulated datasets, for most cases, the precision, recall, and F1-score of LROD were higher than those of Minimap2 and MHAP. Although LROD was slightly inferior to Minimap2 in terms of the F1-score for *E. coli_Real*, it also achieved a similar performance to that of Minimap2. When $k = 15$, as shown in **Table 2** and **Figure 4**, LROD was superior to the other two tools on the basis of the F1-score.

Simulated Datasets

For *E. coli-10*, *E. coli-20,* and *chr20-10*, LROD consistently had the best results in terms of the precision, recall, and F1-score. For these three datasets, the precision of LROD was consistently above 90%. Although the recall of LROD did not reach 90%, all values were >85% and higher than those of the other two tools. The average F1-scores for LROD were 5% and 20% higher than those of Minimap2 and MHAP, respectively. The precision of LROD exceeded 92%, and the recall of LROD exceeded 83%. Both the precision and recall of LROD were higher than those of Minimap2 in both the precision and recall. Therefore, the F1-score of LROD ranked first for these datasets.

Real Datasets

As shown in **Table 2** and **Figures 3**, **4**, for *E. coli_Real*, the F1-scores of LROD and Minimap2 were >95%. Although LROD did not obtain the highest F1-score, it was very close to that of Minimap2. For *C. elegans_Real*, the precisions of Minimap2 and MHAP were smaller than that of LROD. Both Minimap2 and LROD showed good recall, and LROD obtained the best F1-score. For *Human_Real*, the F1-scores of Minimap2 and LROD were similar. Note that the memory requirement of MHAP exceeded the memory capacity of our computer; hence, we did not give its result.

Running Time and Memory Requirements

We compared the computational requirements among the three tools for the six datasets. The results are shown in **Table 3**. The memory consumption of MHAP is very large, and Minimap2 has the lowest memory consumption. Although the memory consumption of LROD is greater than that of Minimap2, it is smaller than that of MHAP. In terms of running time (CPU time), LROD is better than MHAP. The surprising running time

TABLE 2 | Overlap detection on the datasets.

k	Dataset	Precision			Recall			F1-score		
		MHAP	Minimap2	LROD	MHAP	Minimap2	LROD	MHAP	Minimap2	LROD
k = 13	*E. coli-10*	0.871	0.866	0.935	0.599	0.837	0.887	0.710	0.851	0.910
	E. coli-20	0.859	0.855	0.924	0.597	0.827	0.875	0.704	0.841	0.899
	chr20-10	0.685	0.752	0.933	0.612	0.829	0.893	0.646	0.788	0.912
	E. coli_Real	0.967	0.987	0.976	0.875	0.969	0.948	0.919	0.978	0.962
	C. elegan_Real	0.362	0.685	0.752	0.746	0.917	0.909	0.487	0.785	0.824
k = 15	*E. coli-10*	0.878	0.834	0.941	0.490	0.759	0.849	0.629	0.795	0.893
	E. coli-20	0.866	0.825	0.930	0.487	0.751	0.831	0.624	0.786	0.878
	chr20-10	0.734	0.851	0.942	0.504	0.746	0.855	0.598	0.795	0.896
	E. coli_Real	0.963	0.957	0.964	0.798	0.953	0.924	0.873	0.955	0.943
	C. elegan_Real	0.729	0.789	0.897	0.706	0.947	0.958	0.717	0.861	0.926
	Human_Real	–	0.779	0.736	–	0.667	0.706	–	0.719	0.720

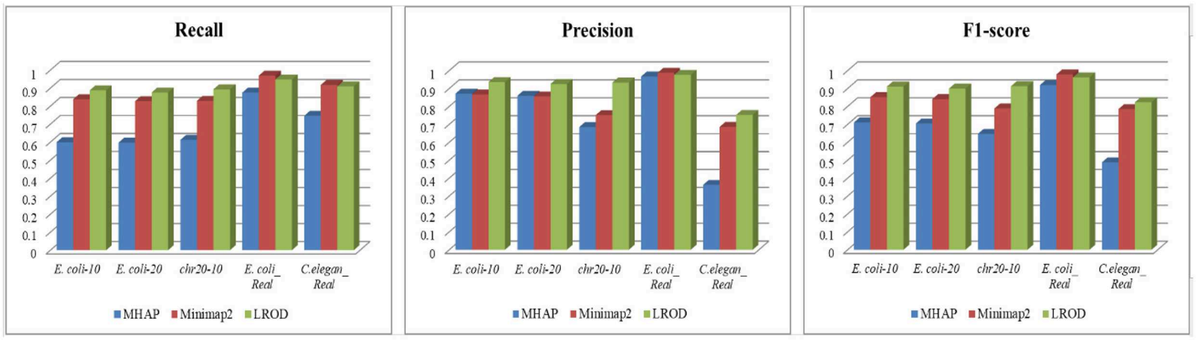

FIGURE 3 | Overlap detection with *k* = 13.

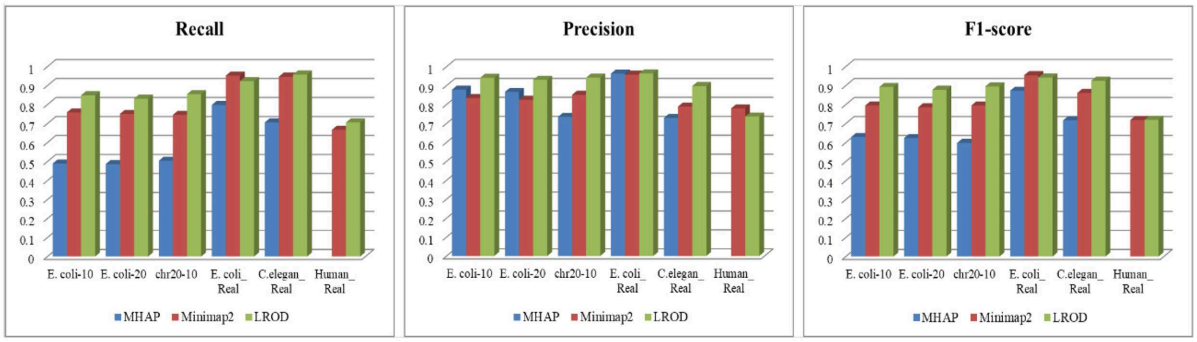

FIGURE 4 | Overlap detection with *k* = 15.

and memory consumption of Minimap2 have attracted attention from many researchers. Note that all tools can use more threads to reduce the wall time.

Discussion

MHAP is a MinHash algorithm used to implement overlap detection based on *k*-mer similarity. For two long reads, MHAP builds a fixed number of *k*-mer sketch lists with the minimum value of the hash functions and then finds the location of the overlap. Only a certain number of *k*-mers are left for each read because the length of each read is different; the shorter the reads are, the more accurate the MHAP detection. Minimap2 ingeniously combines the advantages of its predecessor's algorithms, including DALIGNER, MHAP, and GraphMap (Sović et al., 2016). Minimap2 adopts the streaming SIMD extension instruction calculation method. Hence, the efficiency of Minimap2 is very high. LROD is an overlap detection algorithm based on the *k*-mer distribution. It starts with solid

TABLE 3 | Running time and memory.

k	Dataset	MHAP		Minimap2		LROD	
		Running time	Memory (Mb)	Running time	Memory (Mb)	Running time	Memory (Mb)
$k = 13$	*E. coli-10*	4 m 29 s	39,703	0 m 45 s	1,251	2 m 4 s	1,491
	E. coli-20	9 m 15 s	39,971	2 m 30 s	1,612	5 m 5 s	2,336
	chr20-10	87 m 44 s	42,748	125 m 26 s	8,441	59 m 1 s	11,487
	E. coli_Real	4 m 14 s	39,501	0 m 26 s	1,129	1 m 36 s	1,238
	C. elegans_Real	28,813 m 15 s	42,156	1,753 m 38 s	25,940	14,625 m 23 s	36,708
$k = 15$	*E. coli-10*	6 m 3 s	41,163	0 m 17 s	2,644	2 m 14 s	3,326
	E. coli-20	12 m 34 s	42,959	0 m 48 s	2,972	5 m 51 s	4,248
	chr20-10	88 m 18 s	43,641	21 m 52 s	7,625	46 m 38 s	11,906
	E. coli_Real	4 m 42 s	41,099	0 m 1 s	2,513	1 m 40 s	3,036
	C. elegans_Real	377 m 17 s	44,414	292 m 45 s	15,522	620 m 16 s	17,418
	Human_Real	–	–	424 m 42 s	35,814	4,402 m 10 s	51,906

k-mers selected based on the frequency distribution of *k*-mers for the entire long-read dataset. Although LROD performs well-according to the experimental results, it has an obvious shortcoming in terms of running time. In the future, we will focus on improving the LROD calculation performance module, increasing the calculation speed, and reducing the running time.

CONCLUSION

In this paper, we develop an overlap detection tool named LROD, which performs well in detecting overlaps among the long reads obtained from TGS technology. LROD first selects solid *k*-mers, which can reduce the computation time and memory requirements and can avoid some problems caused by sequencing errors and repetitive regions. For two long reads, LROD first finds their common *k*-mers. Second, LROD adopts a two-stage strategy to detect whether two common *k*-mers are consistent and to search for a chain that corresponds to a candidate overlap. Finally, LROD further evaluates the candidate overlap and determines the real overlap between the two long reads. The experimental results on three simulated datasets and three real datasets show that LROD can obtain satisfactory overlap detection results in terms of the precision, recall, and F1-score.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found at Schatz Lab (http://schatzlab.cshl.edu/data/ectools/; http://schatzlab.cshl.edu/data/nanocorr/). All source codes for LROD are available at Github (https://github.com/luojunwei/LROD).

AUTHOR CONTRIBUTIONS

JL, ZH, and RC proposed the method and designed the experiments. JL and RC wrote the paper. YW and XZ provided guidance for this paper. HL and CY provided support for the completion of the experiment. All authors contributed to the article and approved the submitted version.

FUNDING

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61972134, 61602156, 61802113, and 61772557, Henan Provincial Department of Science and Technology Research Project under Grant No. 192102210118, Doctor Foundation of Henan Polytechnic University under Grant No. B2018-36, and Henan University Scientific and Technological Innovation Team Support Program under Grant No. 19IRTSTHN012.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fgene.2020.00632/full#supplementary-material

REFERENCES

Berlin, K., Koren, S., Chin, C., Drake, J. P., Landolin, J. M., and Phillippy, A. M. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nat. Biotechnol.* 33, 623–630. doi: 10.1038/nbt.3238

Broder, A. Z. (1997). "On the resemblance and containment of documents," in *Proceedings of the Compression and Complexity of Sequences 1997 (Cat. No. 97TB100171)* (Salerno: IEEE), 21–29. doi: 10.1109/SEQUEN.1997.666900

Chaisson, M. J., and Tesler, G. (2012). Mapping single mole-cule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics* 13:238. doi: 10.1186/1471-2105-13-238

Ding, X., and Guo, X. (2018). A survey of SNP data analysis. *Big Data Mining Anal.* 1, 3–20. doi: 10.26599/BDMA.2018.9020015

Farrer, R. A., Kemen, E., Jones, J. D., and Studholme, D. J. (2009). *De novo* assembly of the *Pseudomonas syringae* pv. syringae B728a genome using

Illumina/Solexa short sequence reads. *FEMS Microbiol. Lett.* 291, 103–111. doi: 10.1111/j.1574-6968.2008.01441.x

Ferragina, P., and Manzini, G. (2005). Indexing compressed text. *J. ACM* 52, 552–581. doi: 10.1145/1082036.1082039

Jeffares, D. C., Jolly, C., Hoti, M., Speed, D., Shaw, L. P., Rallis, C., et al. (2017). Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast. *Nat. Commun.* 8:14061. doi: 10.1038/ncomms14061

Levene, M., Korlach, J., Turner, S., Foquet, M., Craighead, H. G., and Webb, W. W. (2003). Zero-mode waveguides for single-molecule analysis at high concentrations. *Science* 299, 682–686. doi: 10.1126/science.1079700

Li, H. (2016). Minimap and miniasm: fast mapping and *de novo* assembly for noisy long sequences. *Bioinformatics* 32, 2103–2110. doi: 10.1093/bioinformatics/btw152

Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 34, 3094–3100. doi: 10.1093/bioinformatics/bty191

Liao, X., Li, M., Luo, J., Zou, Y., Wu, F., Pan, Y., et al. (2020). Improving *de novo* assembly based on read classification. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 17, 177–188. doi: 10.1109/TCBB.2018.2861380

Liu, B., Shi, Y., Yuan, J., Hu, X., Zhang, H., Li, N., et al. (2013). Estimation of genomic characteristics by analyzing kmer frequency in *de novo* genome projects. *arXiv (preprints).* arXiv:1308.2012.

Luo, C., Tsementzi, D., Kyrpides, N. C., Read, T. D., and Konstantinidis, K. T. (2012). Direct comparisons of illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample. *PLoS ONE* 7:e30087. doi: 10.1371/journal.pone.0030087

Luo, J., Wang, J., Li, W., Zhang, Z., Wu, F., Li, M., et al. (2015a). EPGA2: memory-efficient *de novo* assembler. *Bioinformatics* 31, 3988–3990. doi: 10.1093/bioinformatics/btv487

Luo, J., Wang, J., Shang, J., Luo, H., Li, M., Wu, F., et al. (2020). GapReduce: a gap filling algorithm based on partitioned read sets. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 17, 877–886. doi: 10.1109/TCBB.2018.2789909

Luo, J., Wang, J., Zhang, Z., Li, M., and Wu, F. (2016). BOSS: a novel scaffolding algorithm based on an optimized scaffold graph. *Bioinformatics* 33, 169–176. doi: 10.1093/bioinformatics/btw597

Luo, J., Wang, J., Zhang, Z., Wu, F., Li, M., and Pan, Y. (2015b). EPGA: *de novo* assembly using the distributions of reads and insert size. *Bioinformatics* 31, 825–833. doi: 10.1093/bioinformatics/btu762

Miller, J. R., Koren, S., and Sutton, G. (2010). Assembly algorithms for next-generation sequencing data. *Genomics* 95, 315–327. doi: 10.1016/j.ygeno.2010.03.001

Myers, G. (2014). "Efficient local alignment discovery amongst noisy long reads," in *International Workshop on Algorithms in Bioinformatics* (*Berlin*; *Heidelberg*: Springer*)*, 52–67. doi: 10.1007/978-3-662-44753-6_5

Nagarajan, N., and Pop, M. (2013). Sequence assembly demystified. *Nat. Rev. Genet.* 14, 157–167. doi: 10.1038/nrg3367

Pevzner, P. A., Tang, H., and Waterman, M. S. (2001). An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. U.S.A.* 98, 9748–9753. doi: 10.1073/pnas.171285098

Rizk, G., Lavenier, D., and Chikhi, R. (2013). DSK: k-mer counting with very low memory usage. *Bioinformatics* 29, 652–653. doi: 10.1093/bioinformatics/btt020

Ross, M. G., Russ, C., Costello, M., Hollinger, A., Lennon, N., Hegarty, R., et al. (2013). Characterizing and measuring bias in sequence data. *Genome Biol.* 14:R51. doi: 10.1186/gb-2013-14-5-r51

Schadt, E. E., Turner, S. W., and Kasarskis, A. (2010). A window into third-generation sequencing. *Hum. Mol. Genet.* 19:R227–R240. doi: 10.1093/hmg/ddq416

Sović, I., KriŽanovic, K., Skala, K., and Sikic, M. (2016). Evaluation of hybrid and non-hybrid methods for *de novo* assembly of nanopore reads. *Bioinformatics* 32, 2582–2589. doi: 10.1093/bioinformatics/btw237

Stoddart, D., Heron, A. J., Mikhailova, E., Maglia, G., and Bayley, H. (2009). Single-nucleotide discrimination in immobilized DNA oligonucleotides with a biological nanopore. *Proc. Natl. Acad. Sci. U.S.A.* 106, 7702-7707. doi: 10.1073/pnas.0901054106

Ummat, A., and Bashir, A. (2014). Resolving complex tandem repeats with long reads. *Bioinformatics* 30, 3491–3498. doi: 10.1093/bioinformatics/btu437

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.