BELLA: Berkeley Efficient Long-Read to Long-Read Aligner and Overlapper

Giulia Guidi[1,2,*], Marquita Ellis[1,2], Daniel Rokhsar[3,4], Katherine Yelick[1,2], Aydın Buluç[1,2,*]

[1]Computer Science Division, University of California at Berkeley, Berkeley, CA 94720, USA
[2]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[3]Department of Molecular and Cell Biology, University of California at Berkeley, Berkeley, CA 94720, USA
[4]DOE Joint Genome Institute, Walnut Creek, CA 94598, USA
[*]To whom correspondence should be addressed. **Contact**: gguidi@berkeley.edu.

Abstract

De novo assembly is the process of accurately reconstructing a genome sequence using only overlapping, error-containing DNA sequence fragments (*reads*) that redundantly sample a genome. While longer reads simplify genome assembly and improve the contiguity of the reconstruction, current long-read technologies come with high error rates. A crucial step of *de novo* genome assembly for long reads consists of finding overlapping reads. We present Berkeley Long-Read to Long-Read Aligner and Overlapper (BELLA), a novel algorithm for computing overlaps and alignments that balances the goals of recall (completeness) and precision (avoiding incorrect overlaps), consistently performing well on both, and doing so with reasonable compute time and memory usage.

We present a probabilistic model which demonstrates the soundness of using short, fixed length k-mers to detect overlaps, avoiding expensive pairwise alignment of each read against all others. We then introduce a notion of *reliable k-mers* based on our probabilistic model. The use of *reliable k-mers* eliminates both the k-mer set explosion, that would otherwise occur with highly erroneous reads, and the spurious overlaps from k-mers originating in repetitive regions. Finally, we present a new method for separating true (*genomic*) overlaps from false positives using a combination of alignment techniques and probabilistic modeling. Using this methodology the probability of false positives drops exponentially as the length of overlap between sequences increases. On both real and simulated data, BELLA on average outperforms previous tools in the F1 score, meaning that both precision and recall are the best or close to the best. On simulated data, BELLA achieves an average of 2.7% higher recall, 17.9% higher precision, and 10.9% higher F1 score than state-of-the-art tools, while remaining runtime performance competitive.

1 Introduction

Recent advancements in sequencing technologies have made large-scale genomic data more accessible than ever, enabling the characterization of genome structure and its variation between and within species. The analysis of data after sequencing is a challenging task. One of the biggest challenges for the analysis of high-throughput sequencing reads (i.e. short DNA fragments) is whole genome assembly (Zhang *et al.*, 2011), which is the process of aligning and merging DNA fragments in order to reconstruct the original sequence. More specifically, *de novo* genome assembly reconstructs a genome from redundantly sampled fragmentary DNA sequences, without prior knowledge of the genome. *De novo* genome assembly can generate sequences for previously uncharacterized genomes (Simpson and Durbin, 2012). Even for genomes that already have *reference* assemblies, *de novo* assembly can enable the identification of individual-specific genetic features without biases

from the reference, which represents only a single individual or composite. Thus through *de novo* assembly the full range of genetic variation in a species becomes accessible.

Current high-throughput sequencing methods can be divided in two main categories based on the length of the read: "short-read" and "long-read" technologies. The main limitation of short-read technologies (Bentley *et al.*, 2008) is their inability to deal with genomic repeats longer than the read length, resulting in fragmented assemblies due to ambiguous placements of sequences (Phillippy *et al.*, 2008; Nagarajan and Pop, 2009). Conversely, long-read technologies, offered by companies Pacific Biosciences (PacBio) with their Single-Molecule Real Time (SMRT) sequencing (Eid *et al.*, 2009) and Oxford Nanopore Technologies with their Nanopore sequencing (Goodwin *et al.*, 2015), generate long reads with average lengths reaching and often exceeding 10,000 base pairs (bp). These allow the resolution of complex genomic repetitions, enabling more accurate ensemble views. However, the improved read length of these technologies comes at the cost of lower accuracy, with error rates ranging from 5% to 35%. Nevertheless, errors are more random and more evenly distributed within PacBio long reads (Giordano *et al.*, 2017) compared to short-read technologies.

While short read assembly typically relies on the De Bruijn Graph (DBG) abstraction, a de Bruijn graph for long reads would be too fragmented to be useful due to the high error rates. Hence, an overwhelming majority of the state-of-the-art long read assemblers uses the Overlap-Layout-Consensus (OLC) paradigm (Berlin *et al.*, 2015). In OLC assembly the first step is the detection of overlaps between reads and the construction of an overlap graph. The OLC paradigm is used for long reads in part because significantly fewer reads are required to cover the genome (by a factor of 100X), limiting the size of the overlap graph. Highly-accurate overlap detection is a major computational bottleneck of the OLC assembly pipeline (Myers, 2014), mainly due to the compute-intensive nature of pairwise alignment.

At present, several algorithms are capable of overlapping error-prone long-read data with varying accuracy. The prevailing approach is to use an indexing data structure, such as a k-mer index table or a suffix array, to identify a set of initial candidate read pairs, thus mitigating the high cost of computing pairwise alignments in a second stage (Chu *et al.*, 2016).

The process of identifying a set of initial candidate read pairs, sometimes simply known as *overlapping*, affects both the accuracy and the algorithm runtime. Precise identification of initial candidate read pairs minimizes the pairwise alignment running time, while retaining all pairs that truly overlap in the genome. A solid mathematical model is critical for identifying these pairs. In addition, computationally efficient and highly accurate overlapping and alignment algorithms have the potential to improve existing long-read assemblers, enabling *de novo* reference assemblies and detection of genetic variations of higher quality.

The main contributions of this work are the following:

1. Using a Markov chain model (Markov, 1971), we demonstrate the soundness of using a k-mer seed-based approach for accurately identifying initial candidate read pairs.

2. We develop a simple procedure for pruning k-mers and prove that it retains nearly all true overlaps with high probability. The result is greater computational efficiency without loss of accuracy.

3. We reformulate the problem of overlap detection in terms of a sparse matrix-matrix multiplication, which enables the use of high-performance techniques not previously applied in the context of long read overlap and alignment.

4. We couple our overlap detection with a state-of-the-art seed-and-extend banded-alignment method (Döring *et al.*, 2008; Zhang *et al.*, 2000) to attain BELLA, all-to-all long-read aligner.

BELLA uses a new method based on Chernoff bounds to separate true alignments from false positives depending on the alignment score. We show that the probability of false positives drops exponentially as the length of overlap between sequences increases. On simulated data, BELLA achieves high recall (of true alignments) with negligible losses of precision.

2 Approach

Proposed Algorithm

The current work develops a computational efficient and highly accurate algorithm for overlap detection and alignment for long-read *de novo* genome assembly. The algorithm is implemented in a high-performance software package, called Berkeley Long-Read to Long-Read Aligner and Overlapper (BELLA). In this paper, we present the mathematical approaches that underlie the proposed work and describe the implementation.

BELLA uses a seed-based approach to detect overlaps in the context of *de novo* assembly. Such an approach parses the reads into k-mers (i.e. sub-strings of fixed length k), which are then used as feature vectors to identify overlaps amongst all reads. Using a mathematical approach based on the Markov chain model, we first show the feasibility of using a k-mer seed based approach for overlap detection of long-read data with high error rates.

However, not all k-mers are created equal in terms of their usefulness for detecting overlaps. For instance, the overwhelming majority of k-mers that occur only once in the dataset are errors (and are also not useful for seeding overlaps between pairs of reads). Similarly k-mers that occur more frequently than what would be expected given the sequencing depth and the error profile are likely to come from repetitive regions of the genome. It is a common practice to prune the k-mer space using various methodologies (Koren *et al.*, 2017; Lin *et al.*, 2016).

BELLA implements a novel method for filtering out k-mers that are likely to either contain errors or originate from a repetitive region. The k-mers that are retained by BELLA are considered to be *reliable*, where the reliability of a k-mer is defined as its probability of having originated from a unique (non-repetitive) region of the genome. We argue that unique k-mers are sufficient for overlap detection because long reads either (a) include long enough non-repetitive sections to identify overlaps using unique k-mers or (b) they are completely contained within a repeat, in which case their overlaps are ambiguous and uninformative to begin with. BELLA's reliable k-mer detection explicitly maximizes the retention of k-mers that belong to unique regions of the genome, using a probabilistic analysis given the error rate and the sequencing depth. BELLA estimates the error rate from the mean error probability of the data.

BELLA uses a sparse matrix to internally represent its data, where the rows are reads, columns are reliable k-mers, and a nonzero $\mathbf{A}(i, j) \neq 0$ contains the position of the jth reliable k-mer within ith read. Construction of this sparse matrix requires efficient k-mer counting.

Overlap detection is implemented in BELLA using Sparse Generalized Matrix Multiplication (SpGEMM), which allows our algorithm to achieve fast overlapping without using approximate approaches. The running time of BELLA's overlap detection phase is not quadratic in the number of reads as it never initiates a comparison between two reads if they do not share any k-mers. Sparse matrix multiplication is a highly flexible and efficient computational paradigm that enables better organization of computation and generality, because it is able to manipulate complex data structures such as the ones used in finding overlaps using shared k-mers. The implementation of this method within our pipeline enables the use of high-performance techniques previously not applied in the context of long-read alignment. It also allows continuing performance improvements

in this step due to the ever-improving optimized implementations of SpGEMM (Nagasaka *et al.*, 2018; Azad *et al.*, 2016; Dalton *et al.*, 2015).

BELLA's overlap detection has been coupled with an optimized state-of-the-art seed-and-extend algorithm (Döring *et al.*, 2008; Zhang *et al.*, 2000), meaning the alignment between two reads starts from a shared seed (identified in the previous overlap detection) and not necessarily from the beginning of reads. During the alignment stage, BELLA uses a new method to separate true alignments from false positives depending on the alignment score. We prove that the probability of false positives decreases exponentially as the length of overlap between reads increases. In the future, improvements on existing seed-and-extend algorithms and newly developed ones that would enable higher accuracy to be achieved could be integrated seamlessly into our pipeline.

Existing tools also implement approximate overlap detection using sketching. A sketch is a reduced space representation of a sequence. Multiple randomized hash functions convert k-mers into integer fingerprints and a subset of these is selected to represent the sketch of a sequence according to some criterion. For example, Berlin *et al.* (2015) retain only the smallest integer for each hash function and use the collection of these minimum valued fingerprints as sketch. These methods, while fast, are approximate because sketching is a lossy transformation. Conversely, BELLA uses an explicit k-mer representation, which allows us to couple our overlap detection with a seed-and-extend alignment to refine the output and to improve the precision of our algorithm.

3 Methods

Overlapping Feasibility

Chaisson and Tesler (2012) proposed a theory for how long reads contain subsequences that may be used to anchor alignments to the reference genome. The sequences are modeled as random processes that generate error-free regions whose length is geometrically distributed, with each such region separated by an error (Giordano *et al.*, 2017). The result obtained from their theory is the minimum sequence length to have an *anchor* within a confidence interval.

Here, we present a theory on how these subsequences, also known as k-mers, can be used to anchor alignments between two long reads in the context of *de novo* assembly, allowing an accurate overlap discovery among all the reads in a dataset. The initial assumption of our model defines the probability of correctly sequencing a base as equal to:

$$p = (1 - e) \tag{1}$$

where e is the error rate of the sequencer. From this notion, we model the probability of observing k correct consecutive bases on both $read_1$ and $read_2$ as a Markov chain process (Markov, 1971).

The Markov chain process is characterized by a *transition matrix* $\mathbf{P}$ that includes the probabilities to move from one state to another. Each row-index *start* of $\mathbf{P}$ represents the starting state, and each column-index *end* of $\mathbf{P}$ represents the ending state. Each entry of $\mathbf{P}$ is a non-negative number indicating a *transition probability*. Our transition matrix has $(k+1)$ possible states, which lead to $(k+1)^2$ transition probabilities of moving from *start* to *end*. The probability of having one correct base on both reads is p^2. For any state except the *absorbing* state k, an error in at least one of the two sequences sets the model back to state 0, which happens with probability $1 - p^2$; otherwise, the Markov chain transition from state i to $i + 1$ happens with probability p^2. The absorbing state k cannot be abandoned, as both $read_1$ and $read_2$ have already seen k consecutive correct bases. Hence, its transition probability is 1.

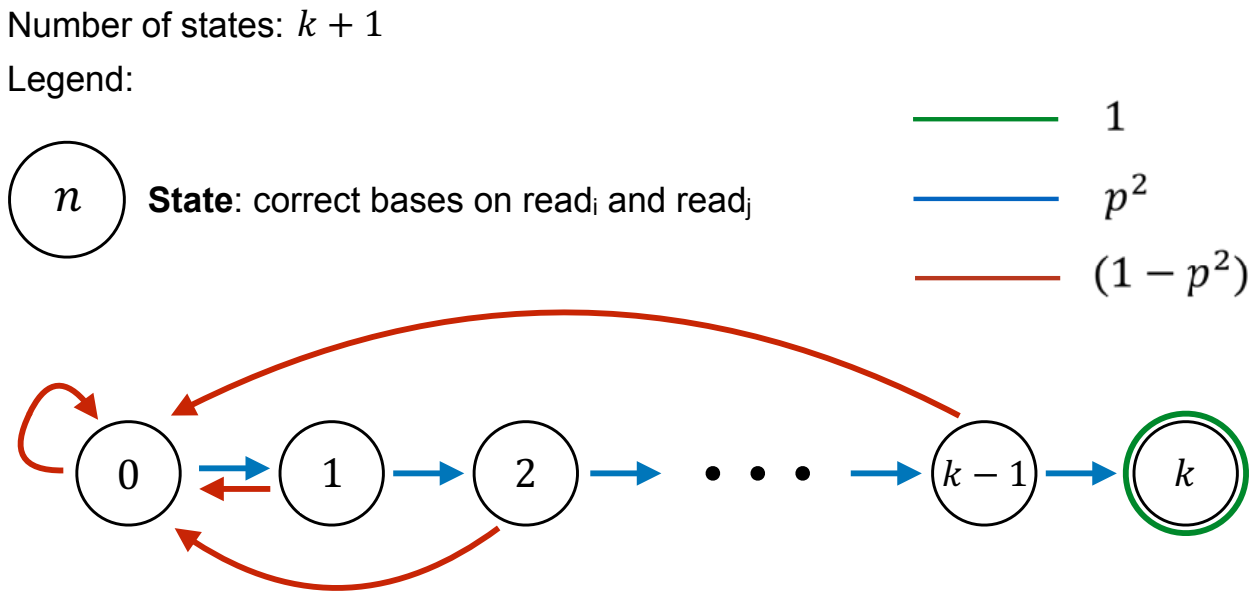

Figure 1: Proposed Markov chain model.

The figure describes the proposed Markov chain model to prove the feasibility of using short k-mers for overlap detection.

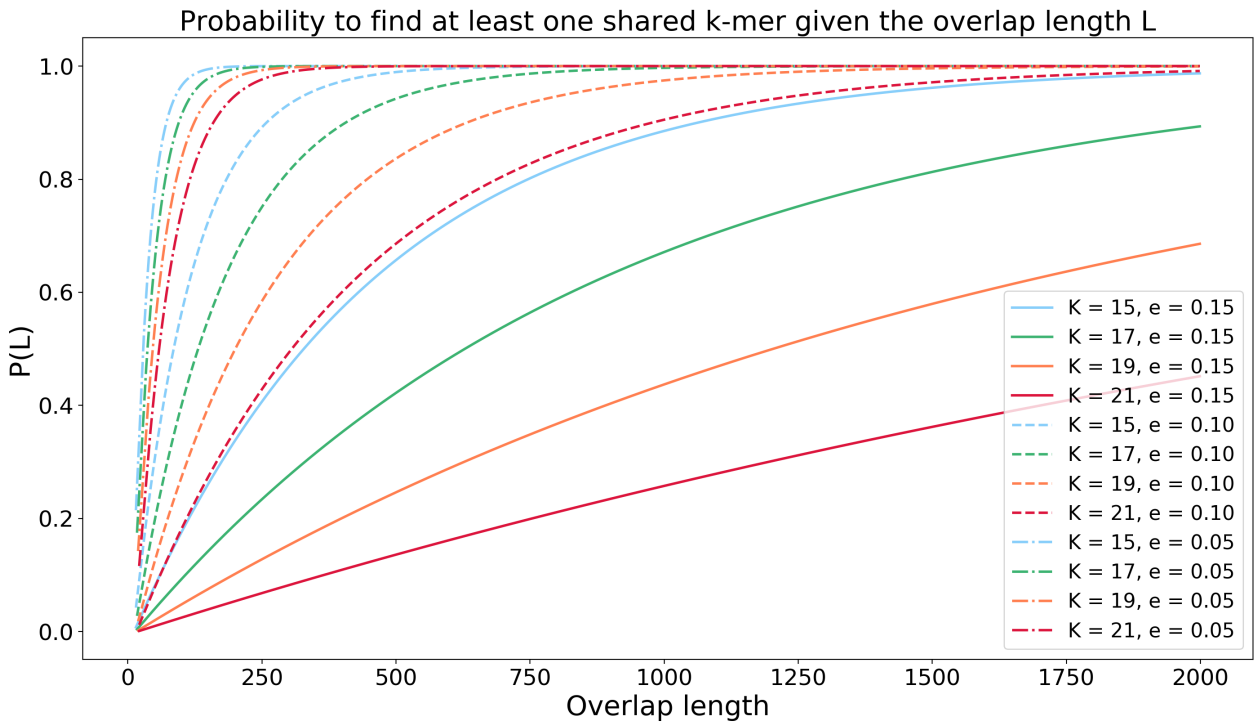

Figure 2: Outcome of the proposed Markov chain model.

The figure illustrates the outcome of the proposed model. The probability of success is set to $p = 0.85$, $p = 0.90$, and $p = 0.95$ representing error rates e of 0.15, 0.10, and 0.05.

Algorithm 1 Probability of observing at least one shared correct k-mer in an overlap region of length $\mathrm{L} > k$.

procedure EstimateSharedKmerProbability(k,L,p)
  $states \leftarrow (k+1)$
  $\mathbf{P} \leftarrow 0$ ▷ Entire matrix initialized to 0
  for $i \leftarrow 0$ to $states$ **do**
    $\mathbf{P}[i, 0] \leftarrow (1 - p^2)$
    $\mathbf{P}[i, i+1] \leftarrow p^2$
  end for
  $\mathbf{P}[states, states] \leftarrow 1$
  $\mathbf{v} \leftarrow (1, 0, \ldots, 0)$ ▷ Initialized to standard unit vector
  for $i \leftarrow 0$ to L **do** ▷ Compute $\mathbf{vP}^{\mathrm{L}}$ without exponentiation
    $\mathbf{v} \leftarrow \mathbf{vP}$
  end for
  return $\mathbf{v}[states]$
end procedure

The transition matrix is filled with the probabilities of moving from one state to another, resulting in the following (for the sake of illustration, k has been set to 5):

$$\mathbf{P} = \begin{bmatrix} (1-p^2) & (1-p^2) & (1-p^2) & (1-p^2) & (1-p^2) & 0 \\ p^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & p^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & p^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & p^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & p^2 & 1 \end{bmatrix} \tag{2}$$

One can then find the probability of being in any of the states after L steps in the Markov chain by computing the L^{th} power of the matrix $\mathbf{P}$. More efficiently, one can just compute $\mathbf{vP}^{\mathrm{L}}$, where $\mathbf{v} \leftarrow (1, 0, \ldots, 0)$ is the standard unit vector; this efficiently exploits L matrix-vector products. This approach is sufficient because we are only interested in the probability of being in the final absorbing state. The above operation leads to the probability of having one correct k-mer in the same location on both reads given a certain overlap region.

Figure 1 describes the process: each state contains the number of successful sequenced bases obtained up to this point on both reads, while the arrows represent the transition probabilities. Algorithm 1 shows this procedure in terms of pseudo-code. In it, p represents the probability of correctly sequencing a base on one read.

The proposed model is the driving factor behind the choice of the optimal k-mer length to be used during the overlap detection phase. Figure 2 illustrates the probabilities of finding one correct k-mer by varying the value of k, the error rate and the overlap length L. The model favors the selection of the shorter k-mer length in order to find a correct seed with high probability even in a narrow overlap region when the error rate is high, because the probability of a k-mer being correct decreases approximately geometrically as its length increases. With decreasing error rate, however, a larger k would be preferable since it would decrease the amount of k-mers coming from repetitive regions of the genome.

Reliable k-mers

Repetitive regions of the genome lead to certain k-mers occurring frequently in the input reads. k-mers from these regions pose two problems for pairwise overlapping and alignment. First, their presence increases the computational cost, both at the overlapping stage and at the alignment stage, because these k-mers generate numerous and possibly spurious overlaps. Second, they often do not provide valuable information.

Our argument here is that k-mers coming from a repetitive region in the genome can be ignored for the purpose of seed-based overlapping. This is because either (a) the read is longer than the repeat, in which case there should be enough sequence data from the non-repeat section to find overlaps, or (b) the read is shorter than the repeat, in which case the read will not be particularly useful for downstream tasks such as *de novo* assembly.

Following the terminology proposed by Lin *et al.* (2016), we identify k-mers that do not exist in the genome as *non-genomic*, thus characterizing k-mers present in the genome as *genomic*. A genomic k-mer can be *repeated*, if it is present multiple times in the genome, or *unique*, if it is not. One can think of the presence of k-mers within each read as that read's feature vector. For the reasons discussed above, the feature vector should include all the unique k-mers, as they often are the most informative features.

Since we do not know the genome prior to assembly, we must estimate the genomic uniqueness of k-mers from our sample of redundant, error-containing reads. In this section, we provide a mathematically grounded procedure that chooses a frequency range for k-mers that we consider to be *reliable*. The basic question that guides the reliable k-mer selection procedure is the following: "Given that a k-mer is sequenced from a unique (non-repeat) region of the genome, what is the probability it will occur at least m times in the input data?". For a genome G sequenced at depth d, the conditional modeled probability is:

$$Pr(\textsc{freq}(k\text{-}mer, G, d) \geq m \mid \textsc{count}(\textsc{map}(k\text{-}mer, G) = 1) \tag{3}$$

where $\textsc{map}(k\text{-}mer, G)$ is the set of locations in the genome G where k-mer can be mapped, $\textsc{count}()$ function computes the cardinality of a given input set, and $\textsc{freq}(k\text{-}mer, G, d)$ is the expected number of occurrences of k-mer within sequenced reads, assuming each region of G is sequenced d times (*sequencing depth*). In that sense, BELLA's approach to select reliable k-mers diverges sharply from how Lin *et al.* (2016) selects their *solid strings*. While solid strings discards infrequent k-mers, our model discards highly-repetitive k-mers, arguing that (a) unique k-mers are sufficient to find informative overlaps, and (b) a unique k-mer has low probability of occurring frequently. However, both approaches are justifiable within their respective problem statements since our model is applied to the OLC paradigm and the one developed by Lin et al. is applied to the DBG paradigm.

The probability of a k-mer being sequenced correctly is approximately $(1-e)^k$, where e is the error rate. The probability of correctly sequencing a k-mer once can be generalized to obtain the probability of seeing it multiple times in the data, considering that each correct sequencing of that k-mer is an independent event. For example, if the sequencing depth is d, the probability of observing a unique k-mer k_i in the input data d times is approximatively $(1-e)^{dk}$. More generally, the number of times a unique k length section of the genome is sequenced correctly when the sequencing depth is d follows a binomial distribution:

$$B(n = d, p = (1-e)^k) \tag{4}$$

where n is the number of trials and p is the probability of success. Consequently, we derive that

Algorithm 2 Reliable k-mer range: Selection of the lower bound (l)

procedure $l \leftarrow$ LowerBound(d, e, k) ▷ d: depth, e: error rate, k: k-mer length
  $sum \leftarrow 0$ ▷ Cumulative sum
  $m \leftarrow 2$ ▷ The k-mer multiplicity in the input
  while $(sum < \epsilon)$ **do**
    $probability \leftarrow P(d, e, k, m)$
    $sum \leftarrow sum + probability$
    $m \leftarrow m + 1$
  end while
  $l \leftarrow m - 1$
  return l ▷ The lower bound
end procedure

Algorithm 3 Reliable k-mer range: Selection of the upper bound (u)

procedure $u \leftarrow$ UpperBound(d, e, k) ▷ d: depth, e: error rate, k: k-mer length
  $sum \leftarrow 0$ ▷ Cumulative sum
  $m \leftarrow d$ ▷ The k-mer multiplicity in the input
  while $(sum < \epsilon)$ **do**
    $probability \leftarrow P(d, e, k, m)$
    $sum \leftarrow sum + probability$
    $m \leftarrow m - 1$
  end while
  $u \leftarrow m + 1$
  return u ▷ The upper bound
end procedure

the probability of observing a k-mer k_i (which corresponds to a unique, non-repetitive region of the genome) m times within a sequencing input data with depth d is:

$$Pr(m; d, (1-e)^k) = \binom{d}{m}(1-e)^{km}(1-(1-e)^k)^{(d-m)} \tag{5}$$

where m is the multiplicity of the k-mer in the input data, e is the error rate, d is the sequencer depth, and k is the k-mer length. Given the values of d, e, and k, the curve $Pr(m; d, (1-e)^k)$ can be computed.

For low values of m, equation 5 might not well approximate the expected number of occurrences of a k-mer k_i, because other sections of the genome could have been morphed into k_i by mistake. However, for medium and high values of m such recurrent morphing is unlikely, especially for values of k high enough to be unique in the genome being sequenced.

Equation 5 is used to identify the range of reliable k-mers. To select the lower bound l, we compute $Pr(m; d, (1-e)^k)$ for each multiplicity m and cumulatively sum up these probabilities, starting from $m = 2$. The cumulative sum does not start from $m = 1$ because a k-mer appearing a single time in the input data (and therefore appearing on a single read) cannot be used to identify an overlap between two reads. The lower bound is the smaller m value after which the cumulative sum exceeds a user-defined threshold ϵ (Algorithm 2). The choice of l matters when the sequencing error rate is relatively low ($\approx 5\%$) or when the sequencing coverage is high ($\approx 50 - 60\times$), or both. This is because in those cases, a k-mer with small multiplicity has a high probability to be incorrect.

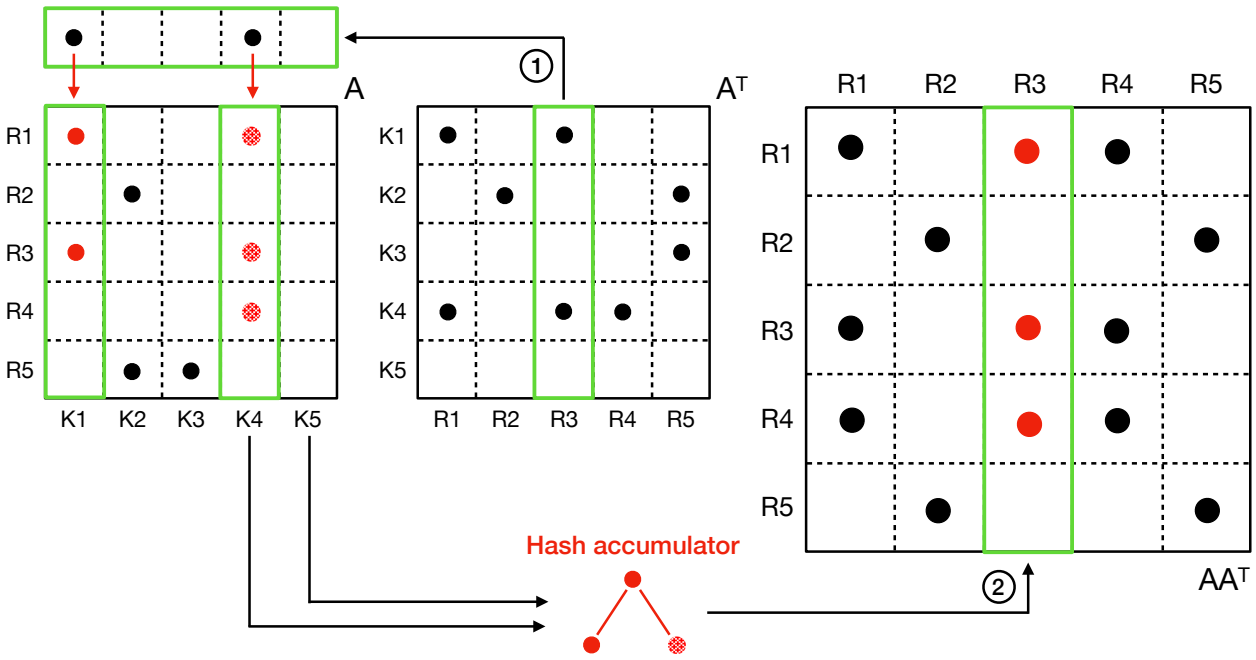

Figure 3: Column-by-column sparse matrix multiplication.

$\mathbf{A}^\mathsf{T}(:, R_3)$ is selected as the "active column": its non-zero elements define which columns of $\mathbf{A}$ need to be considered, by looking at their corresponding row indexes in $\mathbf{A}^\mathsf{T}$.

The upper bound u is chosen by following a similar procedure. Here, the probabilities are cumulatively summed up starting from the largest possible value of m (i.e. d). In this case, u is the largest value of m after which the cumulative sum exceeds the user-defined threshold ϵ (see Algorithm 3). The k-mers that appear more frequently than u have too low a probability of belonging to a unique region of the genome, and multi-mapped k-mers would lead to an increase of computational cost, and potentially to mis-assemblies.

K-mers appearing with greater multiplicities than u and those appearing with smaller multiplicities than l in the input set are discarded and not used as read features in the downstream algorithm. Our reliable k-mer selection procedure discards at most 2ϵ useful information in terms of k-mers that can be used for overlap discovery.

To summarize, both l and u vary with depth, error rate and k-mer length. They are computed by employing the user's estimates for those properties of the dataset.

Sparse Matrix Construction and Multiplication

The overlapper takes the k-mer dictionary from the previous module as input and removes from the dictionary k-mers with occurrences that are outside the reliable range $[l, u]$. It saves only the lexicographically smaller k-mer between itself and its reverse complement.

BELLA uses a sparse matrix format to store its data and sparse matrix-matrix multiplication (SpGEMM) to identify overlaps. Sparse matrices express the data access patterns in a concise and clear manner, allowing better organization of computation. The sparse matrix $\mathbf{A}$, also known as the *data matrix*, is a $|reads|$-by-$|k\text{-}mers|$ matrix with reads as its rows and the entries of the k-mer dictionary as its columns. If the jth reliable k-mer is present in the ith read, the cell (i, j) of $\mathbf{A}$ is non-zero. $\mathbf{A}$ is then multiplied by its transpose, $\mathbf{A}^\mathsf{T}$, yielding a sparse *overlap matrix* $\mathbf{A}\mathbf{A}^\mathsf{T}$ of

dimensions $|reads|$-by-$|reads|$. Each non-zero cell (i, j) of the overlap matrix contains the number of shared k-mers between the ith read and the jth read and the corresponding positions in the corresponding read pair of (at most) two shared k-mers.

The column-by-column sparse matrix multiplication is implemented efficiently using the Compressed Sparse Columns (CSC) format for storing sparse matrices. However, other options are certainly possible in the future, which is one of the advantages of our work. Any novel sparse matrix format and multiplication algorithm would be applicable to the overlapping problem and would enable continued performance improvements since multiple software packages already implement this primitive, including Intel MKL and Sandia's KokkosKernels (Deveci *et al.*, 2017).

The SpGEMM algorithm shown in Figure 3 is functionally equivalent to a k-mer based seed-index table, which is common in other long-read alignment codes. However, the CSC format allows true constant-time random access to columns as opposed to hash tables. More importantly, the computational problem of accumulating the contributions from multiple shared k-mers to each pair of reads is handled automatically by the choice of appropriate data structures within SpGEMM. Figure 3 illustrates the merging operation of BELLA, which uses a hash table data structure indexed by the row indexes of $\mathbf{A}$, following the multi-threaded implementation proposed by Nagasaka *et al.* (2018). Finally, the contents of the hash table are stored into a column of the final matrix once all required nonzeros for that column are accumulated.

In the resulting sparse overlap matrix $\mathbf{AA}^\mathsf{T}$, each non-zero cell (i, j) is a structure composed of an integer value storing the number of shared k-mers, and an integer array of size 4 storing the position on read i and on read j of two shared k-mers (or just one if the pair shares just one k-mer). To enable this special multiplication, which performs scalar multiplication and additions differently than with standard floating point numbers, we use the semiring abstraction (Kepner and Gilbert, 2011). Multiplication on a semiring allows the user to overload scalar multiplication and addition operations and still use the same SpGEMM algorithm. Many existing SpGEMM implementations support user-defined semirings, including those that implement the GraphBLAS API (Buluç *et al.*, 2017).

Increasing the genome size also increases the memory requirements for building the final overlap matrix. For large genomes, it is possible that the sparse overlap matrix $\mathbf{AA}^\mathsf{T}$ would not fit in memory even if the data matrix $\mathbf{A}$ does. BELLA avoids this situation by dividing the multiplication into batches based on the available RAM. At each stage, only a batch of columns of the overlap matrix are created. The set of nonzeros in that batch of the overlap matrix are immediately tested for alignments (as described in Section 3). The pairs that pass the alignment test are written to the output file of BELLA so that the current batch of overlap matrix can be discarded.

Given the nature of our problem, the sparse overlap matrix $\mathbf{AA}^\mathsf{T}$ is a symmetric matrix. Thus, we compute the multiplication using only the lower triangle of $\mathbf{A}$, avoiding computing the pairwise alignment twice for each pair. Currently, there are no known specialized SpGEMM implementations for $\mathbf{AA}^\mathsf{T}$ that store and operate only on $\mathbf{A}$, but we hope to develop one in the future. This would have cut the memory requirements in half. The obvious solution of computing inner projects of rows of $\mathbf{A}$ is suboptimal, because it has to perform $\Omega(|reads|^2)$ inner products even though the majority of inner products are zero. By contrast, our column-by-column implementation runs faster than $O(|reads|^2)$ whenever the overlap matrix $\mathbf{AA}^\mathsf{T}$ is sparse. Given that the main purpose of the overlapping process is used to filter candidate pairs, the overlap matrix tends to be sparse. In practice, Table 4 shows that the overlap matrix is more than 99% sparse in all tested cases.

Pairwise Alignment

BELLA avoids over-filtering in the overlap stage in order to achieve high recall. However, overlapping alone is not sufficient for high precision because the number of shared k-mers between read pairs is not a sufficiently informative feature for filtering false positives. High precision is desirable for avoiding wasted work in subsequent stages of *de novo* assembly. To achieve high precision, BELLA filters candidate read pairs by performing fast (i.e., approximately linear time) pairwise alignments.

For pairwise alignment, BELLA employs a seed-and-extend alignment algorithm. In contrast to approaches that rely on sketches or minimizers, such as Minimap and MHAP, seed-and-extend alignment can be performed directly using the k-mers from BELLA's overlap stage. BELLA's alignment module is based on an x-drop implementation proposed by Zhang *et al.* (2000) and implemented in the SeqAn library (Döring *et al.*, 2008), a `C++` library for sequence analysis. However, these can easily be changed in the future as more advanced pairwise alignment implementations are developed.

BELLA inputs at most two seed k-mers to the x-drop alignment algorithm. If the read pair shares more than two k-mers, two k-mers are chosen such that they are at least n bases apart for the following reason: a second alignment performed with an overlapping or nearby k-mer is likely to be only marginally better than the first, but to cost as much computationally. BELLA's default setting is $n = 1,000$. Our experiments showed that further increasing the number of seeds did not increase accuracy by a large enough margin to justify the extra computational cost, suggesting that using at most two seeds for x-drop alignment represents a good compromise between accuracy and computational cost.

For each read pair in the overlap matrix, the alignment is extended from one- or two-seed k-mers until the alignment score drops x points below the best score seen so far. For example, if the best score seen thus far were 200 with $x = 3$, the algorithm would stop the alignment when the score dropped to 197 or lower. The errors could be non-consecutive: supposing a linear scoring matrix, the score could go down to 198, get back to 199 and finally go down again to 197, at which point the stop criterion is verified and the alignment ends. Once the alignment is complete, if the best score is lower than a threshold n, the pair of sequences is discarded.

The choice of k, the x-drop value, and the alignment score threshold, all influence the final recall and precision. Our theoretical model for tuning these parameters assumes the typical read error-rates of high-throughput PacBio sequencing ($10 - 15\%$). These errors are randomly distributed, and the majority of them are indels, erroneous insertions or deletions of nucleotides (Giordano *et al.*, 2017). We argue that an x-drop factor excessively small, such as $x = 3$, might be too strict for PacBio data as, for example, 3 consecutive insertions on one sequence at the beginning of the alignment would cause the exclusion of a true overlapping pair from the output. Larger values of x could increase the amount of true positives while also making it easier to differentiate true alignments from false positives.

A fixed alignment threshold might not capture the real alignment. Instead, we expect the alignment score to increase as the overlap length between two sequences increases. Consequently, the alignment threshold would benefit from an adaptive behavior according to the estimated overlap length between two sequences that show evidence of overlap, resulting in an *adaptive threshold*. The choice of the scoring matrix used in the pairwise alignment step can justify the alignment score threshold being a linear function of the estimated overlap length.

Given an estimated overlap region of length L and the probability $p = q^2$ of getting a correct base on both sequences, we would expect $m = p \cdot L$ correct matches within that overlap region. The alignment score χ, corresponding to the edit distance, can be written as follows:

$$\chi = \alpha m - \beta(L - m) = \alpha pL - \beta(L - pL) \tag{6}$$

where m is the number of matches, L is the length of the overlap region, α is the value associated with a match in the scoring matrix while β is the penalty for mismatch or a gap/indel ($\alpha, \beta > 0$). Given these assumptions, we define the ratio φ between χ and the estimated overlap length L as:

$$\varphi = \frac{\chi}{L} = \alpha p - \beta(1 - p). \tag{7}$$

The expected value of φ is equal to $2 \cdot p - 1$, if an exact alignment algorithm is used. We would like to define a cutoff in the form of $(1 - \delta)\varphi$, so that we retain pairs over this cutoff as true alignments and discard remaining pairs. We use a Chernoff bound (Chernoff *et al.*, 1952; Hoeffding, 1963) to define the value of δ, proving that there is only a small probability of missing a true overlap of length $L \geq 2000$ bp (which is the minimum overlap length for a sequence to be considered a true-positive) when using the above-defined cutoff. We derived the following Chernoff bound:

$$\mathbf{Pr}[X \leq (1 - \delta)\mu_x] \leq e^{-\delta^2 pL}. \tag{8}$$

Given two sequences that indeed overlap by $L = 2000$, the probability of their alignment score being below the mean by more than 20% ($\delta = 0.2$) is $\leq 7.90 \times 10^{-26}$. The derivation of the above formula is reported in Supplementary Material. BELLA achieved high value of recall and precision among state-of-the-art software tools, with an x-drop value of $x = 7$ and an adaptive threshold derived from the scoring matrix and the with $\delta = 0.2$ cutoff rate.

Lastly, BELLA outputs alignments in a format similar to BLASR/MHAP (Chaisson and Tesler, 2012; Berlin *et al.*, 2015), which outputs a `.m4` file. For each pair of reads that pass both the overlapping and alignment stage filters, the output includes a line with: the respective (a) identifiers of the reads, (b) number of shared k-mers, (c) alignment score, (d) strand information (n if the reads belong to the same strand, c if they are not, following DALIGNER (Myers, 2014) convention), (e) start and end positions of the alignment in the first read, (f) start and end positions of the alignment in the second read, and (g) lengths of the reads. If a pair of reads share two k-mers and the respective alignment scores both exceed the alignment score threshold, the higher-scoring alignment information is output.

4 Evaluation

The datasets used for evaluation are listed in Table 1. We selected genomes with varying size and complexity since analysis results are sensitive to these features (Li *et al.*, 2012).

As performance metrics, we used recall, precision, F1 score, and running time. Recall is defined as the fraction of true positives of the aligner/overlapper over the total size of the *ground truth*; precision is the fraction of true positives of the aligner/overlapper over the total number of elements found by the aligner/overlapper (i.e., total size of its true and false positives combined). F1 score is the harmonic average of the precision and recall. F1 score is a better metric to use when seeking a balance between precision and recall. Overall, a tool that consistently performs well in both precision and recall, rather than achieving low scores in either, is most valuable. F1 is a good indicator of that.

More formally,

Table 1: Data sets used for evaluation.

For each data set, we listed the name of the dataset, the scientific name of the species, the sequencing technology used to obtain the raw data together with the sequencing depth, the source link, and the size of the `fastq` file containing the raw reads. PBSIM is the PacBio read simulator (Ono *et al.*, 2012).

Short Name	Species and Strain	Sequencing Technology	Source Link	Fastq Size
E.coli (Sample)	*Escherichia coli* MG1655 strain	PacBio RS II P5-C3 30X	https://bit.ly/2EEq3JM (CBCB)	266 MB
E.coli	*Escherichia coli* MG1655 strain	PacBio RS II P4-C2 100X	https://bit.ly/2POV1Qs (NCBI)	929 MB
C.elegans 40X	*Caenorhabditis elegans* Bristol mutant strain	PacBio RS II P6-C4 40X	https://bit.ly/2SU7Tqs (PacBio)	8.90 GB
P. aeruginosa	*Pseudomonas aeruginosa* PAO1	PacBio PBSIM CLR 30X	https://bit.ly/2PQmCB5 (NCBI)	359 MB
V. vulnificus	*Vibrio vulnificus* YJ016	PacBio PBSIM CLR 30X	https://bit.ly/2UTHNpw (NCBI)	288 MB
A. baumannii	*Acinetobacter baumannii*	PacBio PBSIM CLR 30X	https://bit.ly/2rHBhF4 (NCBI)	248 MB
C.elegans 20X	*Caenorhabditis elegans*	PacBio PBSIM CLR 20X	https://bit.ly/2EzoDzF (NCBI)	3.75 GB

$$recall = \frac{TP}{TP + FN} \tag{9}$$

$$precision = \frac{TP}{TP + FP} \tag{10}$$

$$\text{F1 } score = 2 \cdot \frac{recall \cdot precision}{recall + precision} \tag{11}$$

where TP is size of true positives, FP is the size of false positives, and FN is the size of false negatives. We consider a read pair as true-positive if the sequences align for at least 2 kb in the reference genome. We derived the threshold $t = 2$ kb from the procedure proposed by Li (2016), and generated the ground truth using Minimap2. A detailed description of our evaluation procedure and ground truth generation can be found in Supplementary Material.

Results

We evaluated BELLA against several state-of-the-art overlappers, using both simulated and real PacBio data. The simulated data was generated using PBSIM (Ono *et al.*, 2012) with an error rate of 15%. The results are shown in Table 2 and Table 3, respectively. The last column of each table indicates whether the respective overlapper also performs alignment on overlapping reads.

Simulated data has the advantage that ground truth is known. Since the reads are extracted from a fixed genome, the overlaps are known by construction. Table 2 shows that MECAT, Minimap2, and MHAP made significant trade-offs in recall versus precision; while MECAT had the highest precision, it had missed a large number of the true overlaps. In contrast, both BELLA and BLASR were consistently strong (typically over 80%) in both precision and recall, but BLASR had a much higher computational cost (on average, 4.3× slower than BELLA). The quality consistency of BELLA is reflected in its F1 scores, which outperformed all the others. Minimap2 was the fastest tool for the whole set of simulated data, but performed only overlap and not alignment.

For the evaluation with real data, since exact error rates are unknown, BELLA estimates the mean error rate e from the read set, using the per-base quality scores provided with the data. The results are shown in Table 3. Although BLASR performed reasonably well on the synthetic data, it achieved very low recall for certain real data sets (as low as 3.23% recall for *C. elegans* 40X). DALIGNER proved to be the fastest of the tools that performs alignment, but BELLA was within

Table 2: Recall, precision, F1 score, and time comparison (synthetic data).

The last column indicates whether the considered aligner does actual alignment or just overlap detection. Precision, recall, and F1 score are reported in percentage. Bold font indicates best performance and underlined font indicates second-best performance. DALIGNER failed on all simulated data sets, and hence is not included in this result set.

Data Set	Overlapper	Recall	Precision	F1 Score	Time (s)	Alignment
P. aeruginosa	BELLA	**99.63**	88.66	**93.83**	268.57	Y
	BLASR	87.39	90.51	88.92	553.50	Y
	MECAT	38.40	**95.20**	54.73	39.70	N
	Minimap2	98.60	72.03	83.25	**19.06**	N
	MHAP	72.68	63.39	67.72	132.38	N
V. vulnificus	BELLA	**99.33**	80.34	**88.83**	57.08	Y
	BLASR	87.31	84.74	86.01	363.59	Y
	MECAT	43.56	**88.89**	58.47	39.46	N
	Minimap2	95.99	73.46	83.23	**12.66**	N
	MHAP	74.52	45.08	56.18	126.89	N
A. baumannii	BELLA	**99.51**	82.53	**90.23**	58.24	Y
	BLASR	89.51	84.58	86.98	325.50	Y
	MECAT	46.31	**90.39**	61.24	34.95	N
	Minimap2	96.10	71.02	81.68	**9.42**	N
	MHAP	76.88	28.78	41.88	131.64	N
C. elegans 20X	BELLA	**97.87**	85.81	**91.44**	2,491.50	Y
	BLASR	95.74	78.22	86.10	7,919.94	Y
	MECAT	13.46	**95.09**	23.58	311.90	N
	Minimap2	94.93	69.90	80.51	**209.21**	N
	MHAP	82.57	6.41	11.90	5,623.41	N

2.5× of its runtime, and dominated in quality (precision and recall). As on the synthetic data, BLASR was the slowest by a significant margin.

Finally, Table 4 shows the number and the percentage of candidate overlaps discarded by BELLA, before and after the alignment step, on four representative data sets. For comparison, the number of reads squared (the potential all-to-all pairwise alignment workload) is also provided. The overlap detection procedure of BELLA was able to filter over **99%** of all read pairs before performing pairwise alignment.

5 Discussion

BELLA proposes a computationally efficient and highly accurate approach for overlapping and aligning noisy long reads, based on mathematical models that minimize the cost of overlap detection, without missing large numbers of true overlaps. The evaluation results in Tables 2, 3, and 4 demonstrate BELLA's superior accuracy compared to state-of-the-art software on simulated data where reliable ground truth information is known, and competitive accuracy on real data where the ground truth is based on the results of read to reference alignment. The runtime of BELLA is, for the most part, within the average of competitive overlappers, which is noteworthy (given that BELLA also outputs alignments). Further, these alignments are sufficiently accurate to facilitate

Table 3: Recall, precision, F1 score, and time comparison (real data).

Recall, precision, F1 score, and time comparison (real data). The last column indicates whether the considered aligner does actual alignment or just overlap detection. Precision, recall, and F1 score are reported in percentage. Bold font indicates best performance and underlined font indicates second-best performance.

Data Set	Overlapper	Recall	Precision	F1 Score	Time (s)	Alignment
E. coli (Sample)	BELLA	77.60	**81.89**	79.69	56.65	Y
	DALIGNER	87.03	59.96	71.00	24.24	Y
	BLASR	77.64	77.84	77.74	301.72	Y
	MECAT	78.41	78.28	78.34	30.78	N
	Minimap2	**93.28**	71.38	**80.87**	**14.77**	N
	MHAP	79.71	66.36	72.42	54.79	N
E. coli	BELLA	64.32	68.84	66.50	317.01	Y
	DALIGNER	80.39	51.99	63.14	200.53	Y
	BLASR	35.43	71.13	47.30	1,508.69	Y
	MECAT	54.61	**72.18**	62.18	**125.27**	N
	Minimap2	**82.40**	56.62	**67.12**	167.51	N
	MHAP	67.84	44.10	53.45	303.12	N
C. elegans 40X	BELLA	70.28	68.81	69.54	9,925.58	Y
	DALIGNER	61.10	56.87	58.91	6,641.82	Y
	BLASR	3.23	18.26	10.27	114,744.57	Y
	MECAT	73.05	**75.09**	**74.06**	**1,014.12**	N
	Minimap2	**95.18**	33.43	49.48	1,644.61	N
	MHAP	69.72	37.87	49.08	8,371.50	N

Table 4: Filtering performance of BELLA.

The table shows (1) the number of all possible read pairs in the data set (before filtering) (2) the number of read pairs after BELLA's overlap stage filtering and (3) the number and percentage (in brackets) of read pairs after BELLA's pairwise alignment filtering.

Dataset	Read[2]	Overlapping Pair	Pairwise Alignment
E. coli (Sample)	285 M	2.27M (0.796%)	331K (0.116%)
C. elegans 40X	177 G	1.37G (0.774%)	12.10M (0.007%)
P. aeruginosa 30X	4 G	14.8M (0.370%)	1.56M (0.039%)
C. elegans 20X	449 G	414M (0.092%)	10.76M (0.002%)

the downstream analysis and assembly tasks.

On simulated data, BELLA achieved both high precision and recall, thus obtaining the best F1 score on the whole set of data. Table 2 shows that BELLA outperformed its closest competitor, Minimap2, with an average of 2.7% higher recall, 17.9% higher precision, and 10.9% higher F1 score. For *P. aeruginosa*, the Minimap2's recall was comparable to that achieved by BELLA; for the remaining data sets, BELLA clearly achieved the highest recall.

On real data, BELLA had the highest precision for *E. coli* (Sample) data set and consistently high F1 scores, which are comparable with the scores of BELLA's main competitor, Minimap2, for the two *E. coli* data sets. Notably, BELLA had a 40.5% higher F1 score than Minimap2 for *C. elegans* 40X. Overall, the top performer on one data set becomes one of the worst on some other

data set whereas BELLA's F1 score is consistently within 6.5% of the top entry.

If not user-defined, BELLA estimates the error rate e from the read set. If the error rate e is known, we recommend providing it to BELLA. Along with the depth and k-mer length, e is one of the parameters which BELLA uses to compute the reliable k-mer range, and the adaptive alignment score threshold. Consequently, the error rate of the dataset influences which k-mers are retained for overlap detection, which in turn affects BELLA's recall and precision.

Tables 2 and 3 show that BELLA achieved higher values of recall on synthetic data compared to real data. This behavior could be related to how ground truth sets are generated. Using synthetic data, the exact location from which a read originates is known, since this location was computed in a deterministic way by the tool (PBSIM) that generated this data. When real data is used, the location of reads is determined by Minimap2. However, there is no guarantee that Minimap2 correctly locates all the alignments, or that it finds every single correct alignment. Hence, BELLA could potentially find more correct alignments than the ones identified by Minimap2. Therefore, it is possible that BELLA's true accuracy on real data is higher in reality. We plan to investigate these issues deeper in the future.

BELLA saves and uses at most two shared k-mers as seeds for the alignment step. These k-mers are chosen such that they are at least n bases apart. This avoids additional, computationally expensive pairwise alignments with overlapping or nearby k-mers. If performance is not a concern, the user can ignore this optimization and use all available k-mers (separated by at least n bases) as seeds. For example, when using all reliable k-mers separated by at least $1,000$ bps, BELLA achieved 4.1% higher recall (80.84%), 0.1% higher precision (81.96%), and 2.1% higher F1 score (81.39%) for *E. coli* (Sample), thus outperforming Minimap2 for quality on this data set. This quality improvement cost a $\approx 2\times$ slower runtime; this ultimately influenced our choice of selecting at most two shared k-mers.

On the other hand, if recall and speed (and not necessarily precision) are the user's main interest, the user can skip the pairwise alignment and use the overlapping module alone; in which case, BELLA achieves its maximum recall. For example, given *E. coli* (Sample), BELLA achieved a recall of 94.02% with a speedup of $\approx 2\times$ with respect to the value reported in Table 3, outperforming MECAT with respect to time.

Table 4 showed that BELLA can filter over 99% of all read pairs yet achieve high values of recall. This demonstrates the high-quality filtering performance of our overlap detection; it excludes a massive amount of non-overlapping read pairs without losing correct ones.

We note that parallel SpGEMM implementations offer a path for efficient parallelization of the BELLA methods described above.

6 Conclusion

Single-molecule sequencing technologies have favored overlap-based approaches (OLC) for *de novo* assembly, enabling highly accurate reconstruction of complex genomes. However, overlap detection using long-read data is a major computational bottleneck. High error rates further complicate overlap detection since it becomes hard to separate signal from the noise.

We presented BELLA, a computationally efficient and highly accurate long-read to long-read aligner and overlapper. BELLA uses a k-mer based approach to detect overlaps between noisy, long-read data. We demonstrated the feasibility of this approach through a mathematical model based on Markov chains. BELLA provides a novel algorithm for pruning k-mers that are unlikely to be useful in overlap detection and whose presence would only incur unnecessary computational costs. This reliable k-mers detection algorithm explicitly maximizes the probability of retaining

k-mers that belong to unique regions of the genome.

To achieve fast overlapping without sketching, BELLA uses sparse matrix-matrix multiplication and utilizes high-performance software and libraries developed for this sparse matrix subroutine. Any novel sparse matrix format and multiplication algorithm would be applicable to overlap detection and enable continued performance improvements. BELLA's overlap detection has been coupled with a state-of-the-art seed-and-extend banded-alignment method. We developed and implemented a new method to separate true alignments from false positives depending on the alignment score. This method demonstrated that the probability of false positives decreases exponentially as the length of overlap between sequences increases.

BELLA achieves higher recall and F1 scores than state-of-the-art tools on simulated data and high values of precision and F1 score on real data, while being performance competitive, thus demonstrating the validity of our approach. Future work includes a further characterization of real data features and the development of a high-performance pairwise alignment algorithm to improve BELLA's performance.

Acknowledgements

Thanks to Heng Li for the help with Minimap2 and to Rob Egan and Steven Hofmeyr for valuable discussions.

Funding

This work is supported by the Advanced Scientific Computing Research (ASCR) program within the Office of Science of the DOE under contract number DE-AC02-05CH11231. We used resources of the NERSC supported by the Office of Science of the DOE under Contract No. DEAC02-05CH11231. This research was also supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

References

Angluin, D. and Valiant, L. G. (1979). Fast probabilistic algorithms for hamiltonian circuits and matchings. *Journal of Computer and system Sciences*, **18**(2), 155–193.

Azad, A., Ballard, G., Buluc, A., Demmel, J., Grigori, L., Schwartz, O., Toledo, S., and Williams, S. (2016). Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. *SIAM Journal on Scientific Computing*, **38**(6), C624–C651.

Bentley, D. R., Balasubramanian, S., Swerdlow, H. P., Smith, G. P., Milton, J., Brown, C. G., Hall, K. P., Evers, D. J., Barnes, C. L., Bignell, H. R., *et al.* (2008). Accurate whole human genome sequencing using reversible terminator chemistry. *nature*, **456**(7218), 53–59.

Berlin, K., Koren, S., Chin, C.-S., Drake, J. P., Landolin, J. M., and Phillippy, A. M. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, **33**(6), 623–630.

Buluç, A., Mattson, T., McMillan, S., Moreira, J., and Yang, C. (2017). Design of the graphblas API for C. In *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 643–652. IEEE.

Chaisson, M. J. and Tesler, G. (2012). Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC bioinformatics*, **13**(1), 238.

Chernoff, H. *et al.* (1952). A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, **23**(4), 493–507.

Chu, J., Mohamadi, H., Warren, R. L., Yang, C., and Birol, I. (2016). Innovations and challenges in detecting long read overlaps: an evaluation of the state-of-the-art. *Bioinformatics*, **33**(8), 1261–1270.

Dalton, S., Olson, L., and Bell, N. (2015). Optimizing sparse matrix—matrix multiplication for the GPU. *ACM Transactions on Mathematical Software (TOMS)*, **41**(4), 25.

Deveci, M., Trott, C., and Rajamanickam, S. (2017). Performance-portable sparse matrix-matrix multiplication for many-core architectures. In *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 693–702. IEEE.

Döring, A., Weese, D., Rausch, T., and Reinert, K. (2008). SeqAn an efficient, generic C++ library for sequence analysis. *BMC bioinformatics*, **9**(1), 11.

Eid, J., Fehr, A., Gray, J., Luong, K., Lyle, J., Otto, G., Peluso, P., Rank, D., Baybayan, P., Bettman, B., *et al.* (2009). Real-time DNA sequencing from single polymerase molecules. *Science*, **323**(5910), 133–138.

Giordano, F., Aigrain, L., Quail, M. A., Coupland, P., Bonfield, J. K., Davies, R. M., Tischler, G., Jackson, D. K., Keane, T. M., Li, J., *et al.* (2017). De novo yeast genome assemblies from MinION, PacBio and MiSeq platforms. *Scientific reports*, **7**(1), 3935.

Goodwin, S., Gurtowski, J., Ethe-Sayers, S., Deshpande, P., Schatz, M. C., and McCombie, W. R. (2015). Oxford nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome research*, **25**(11), 1750–1756.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, **58**(301), 13–30.

Kepner, J. and Gilbert, J. (2011). *Graph algorithms in the language of linear algebra*. SIAM.

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., and Phillippy, A. M. (2017). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, **27**(5), 722–736.

Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*.

Li, H. (2016). Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, **32**(14), 2103–2110.

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). The sequence alignment/map format and samtools. *Bioinformatics*, **25**(16), 2078–2079.

Li, Z., Chen, Y., Mu, D., Yuan, J., Shi, Y., Zhang, H., Gan, J., Li, N., Hu, X., Liu, B., *et al.* (2012). Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph. *Briefings in functional genomics*, **11**(1), 25–37.

Lin, Y., Yuan, J., Kolmogorov, M., Shen, M. W., Chaisson, M., and Pevzner, P. A. (2016). Assembly of long error-prone reads using de bruijn graphs. *Proceedings of the National Academy of Sciences*, **113**(52), E8396–E8405.

Markov, A. (1971). Extension of the limit theorems of probability theory to a sum of variables connected in a chain.

Myers, G. (2014). Efficient local alignment discovery amongst noisy long reads. In *International Workshop on Algorithms in Bioinformatics*, pages 52–67. Springer.

Nagarajan, N. and Pop, M. (2009). Parametric complexity of sequence assembly: theory and applications to next generation sequencing. *Journal of computational biology*, **16**(7), 897–908.

Nagasaka, Y., Matsuoka, S., Azad, A., and Buluç, A. (2018). High-performance sparse matrix-matrix products on intel knl and multicore architectures. *arXiv preprint arXiv:1804.01698*.

Nethercote, N., Walsh, R., and Fitzhardinge, J. (2006). Building workload characterization tools with valgrind. *Invited tutorial, IEEE International Symposium on Workload Characterization (IISWC 2006)*.

Ono, Y., Asai, K., and Hamada, M. (2012). Pbsim: Pacbio reads simulator—toward accurate genome assembly. *Bioinformatics*, **29**(1), 119–121.

Phillippy, A. M., Schatz, M. C., and Pop, M. (2008). Genome assembly forensics: finding the elusive mis-assembly. *Genome biology*, **9**(3), R55.

Simpson, J. T. and Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data structures. *Genome research*, **22**(3), 549–556.

Zhang, W., Chen, J., Yang, Y., Tang, Y., Shang, J., and Shen, B. (2011). A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PloS one*, **6**(3), e17915.

Zhang, Z., Schwartz, S., Wagner, L., and Miller, W. (2000). A greedy algorithm for aligning DNA sequences. *Journal of Computational biology*, **7**(1-2), 203–214.

Supplementary Material

A Error Rate of the Input Read Set

The error probability P_{err} of a quality score Q is computed for each position of a read as proposed by Ono *et al.* (2012):

$$P_{err} = 10^{\frac{-Q}{10}}$$

The error rate e of a given data set is then computed averaging the error probability P_{err} over the entire read set.

B Unique k-mer Distribution in Sample Data

The plot in Figure 4 shows the distribution of k-mers in the genome given their occurrence in the input set of reads. In order to obtain the plot, we mapped k-mers from from a $d = 30$ sampling of the *E. coli* reference genome. The plot shows that the majority of unique k-mers are concentrated in the left side and corresponds to small occurrences in the input of the considered k-mer. Note that the histogram depends on sequencing depth. The majority of k-mers in the right tail either occur multiple times or do not occur in the genome at all. Note that the y-axis is reported in logarithmic scale; hence the amount of unique k-mers in the right tail is extremely small.

C Chernoff Bound for Overlap Detection

We will use a Chernoff bound (Chernoff *et al.*, 1952; Hoeffding, 1963) to define the value of δ used in Section 3. This way, we will prove that there is a very small probability of missing a true overlap of length $L \geq 2000$ *bp* when using the $(1-\delta)\varphi$ cutoff defined in Section 3. Recall that 2000 *bp* is the minimum overlap length for a sequence to be considered a true-positive as described in Section 4.

Let Z be a sum of independent random variables $\{Z_i\}$, with $E[Z] = \mu_z$; we assume for simplicity that $Z_i \in \{0, 1\}$, for all $i \leq L$. The Chernoff bound defines an upper bound of the probability of Z to deviate of a certain quantity δ from its expected value. Specifically we use a corollary of the multiplicative Chernoff bound (Angluin and Valiant, 1979), which is defined for $0 \leq \delta \leq 1$ as:

$$\mathbf{Pr}[Z \leq (1-\delta)\mu_z] \leq e^{\frac{-\delta^2\mu_z}{2}}. \tag{12}$$

In order to obtain the Chernoff bound for the ratio φ, we consider a random variable $X_i \in \{-\beta, \alpha\}$ such that:

$$X_i = \begin{cases} \alpha, & \text{with probability } p \\ -\beta, & \text{with probability } 1-p \end{cases} \tag{13}$$

where $\alpha, \beta > 0$ still are the values associated to a match and to a mismatch or a gap/indel in the scoring matrix, respectively; its expected value $E[X_i]$ is exactly equal to φ in Eq. 7. Since the Chernoff bound is defined for a sum of independent random variables $Z_i \in \{0, 1\}$, we need to move from $X_i \in \{-\beta, \alpha\}$ to $Z_i \in \{0, 1\}$. Therefore, we define a new random variable $Y_i = X_i + \beta$ as a linear transformation of X_i, which can assume values $\{0, \alpha+\beta\}$. Given $E[Y_i] = E[X_i]+\beta = (\alpha+\beta)p$, we can normalize Y_i to obtain the desired random variable Z_i:

$$Z_i = \frac{X_i + \beta}{\alpha + \beta}, \text{ where } Z_i \in \{0, 1\}. \tag{14}$$

From the linearity of expectation, we have

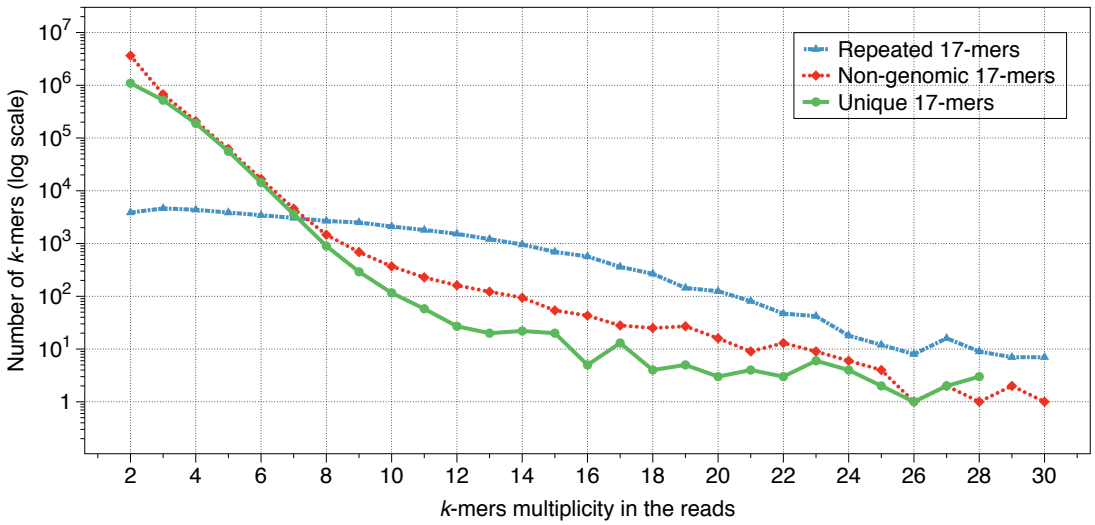

Figure 4: Unique k-mer distribution in sample data.

The plot is obtained using the Escherichia coli str. K-12 substr. MG1655 genome, downloaded from the National Center for Biotechnology Information (depth $d = 30$, k-mer length $k = 17$, and error rate $e = 0.15$). Each data point in the plot represents the number k-mers with that specific frequency. Different colors identify the amount of unique, non-genomic, and repeated k-mers for each frequency. High-frequency k-mers lead statistically to spurious overlaps.

$$E[Z] = E\left[\frac{X+\beta}{\alpha+\beta}\right] = \frac{E[X]+\beta L}{\alpha+\beta} = \frac{(2p-1)L+\beta L}{\alpha+\beta}. \tag{15}$$

Substituting Eq. 14 and Eq. 15 in Eq. 12 and simplifying using our scoring matrix of $\alpha, \beta = 1$, we obtain the final expression:

$$\mathbf{Pr}[X \leq (1-\delta)\mu_x] \leq e^{-\delta^2 pL}, \text{ with } E[X] = \mu_x. \tag{16}$$

To interpret this bound, consider an error rate of 15%. Given two sequences that indeed overlap by $L = 2000$, the probability of their alignment score being below the mean by more than 20% ($\delta = 0.2$) is $\leq 7.90 \times 10^{-26}$.

D Evaluation Procedure

We applied two different procedures to obtain values of recall and precision, one for real data and one for synthetic data. For each genome, a file containing the alignment positions of the input reads in the corresponding reference genome represents the *ground truth*.

The definition of ground truth is valid for both real and synthetic data. However, the ground truth generation procedure differs for the two categories of data as described in the following sections.

E Synthetic Data Set

Generating ground truth is simpler when dealing with synthetic data, and we accomplished it by running the read simulator of Pacific Biosciences, PBSIM (Ono *et al.*, 2012). The output of

the simulator comprises the following files: one (or more, depending on the size of the genome) files in `fastq` format (i.e. the input file of BELLA and other software) and one (or more) files in multiple-alignment format (MAF). The MAF file stores the information about where reads in the genome were synthesized; it is the equivalent of the single-mapped read-to-reference alignment ground truth generated for the real datasets using Minimap2. In this case, the exact location where a read was created is known, and therefore only single-mapped ground truth exists. We processed the MAF file (for synthetic data) and the two ground truth files (for real data) to extract the information required to compute the number of true overlaps: the read identifiers and the start and end positions in the genome.

The information stored in the ground truth was used to (a) compute the amount of true overlaps, which constitutes the denominator in the formula to calculate recall, and (b) store reads information about their sub-reference sequence, and the start and end position in the reference. If the genome has more than one chromosome, we define the sub-reference sequence as the chromosome to which the read belongs; otherwise, the sub-reference sequence is the same for all the reads in a set. For each tool, we discarded self-paired reads (i.e., alignments of reads to themselves). Subsequently, we used the information stored in the ground truth to distinguish true positives from other read pairs. As previously defined, a read pair is considered a true positive if the two sequences align for at least $\tau = 2$ kb, considering the start and end positions of their alignment in the reference genome (reported in the ground truth).

Table 5: Recall, precision, and F1 score comparison with multi-mapped ground truth.

The last column indicates whether the considered aligner does actual alignment or just overlap detection. Precision, recall, and F1 score are reported in percentage. Bold font indicates best performance and underlined font indicates second-best performance. Times are the same as in Table 3.

Data Set	Overlapper	Recall	Precision	F1 Score
E. coli (Sample)	BELLA	66.99	91.18	77.24
	DALIGNER	<u>81.64</u>	73.00	77.08
	BLASR	71.10	**92.63**	<u>80.45</u>
	MECAT	70.95	<u>92.50</u>	80.30
	Minimap2	**87.43**	87.01	**87.22**
	MHAP	72.79	79.00	75.77
E. coli	BELLA	49.37	87.51	<u>63.13</u>
	DALIGNER	<u>67.58</u>	73.33	70.34
	BLASR	27.83	<u>94.79</u>	43.03
	MECAT	42.26	**95.33**	58.56
	Minimap2	**70.53**	81.67	**75.69**
	MHAP	55.68	61.38	58.39
C. elegans 40X	BELLA	53.99	<u>79.83</u>	<u>64.42</u>
	DALIGNER	50.74	71.61	59.40
	BLASR	2.60	22.33	4.66
	MECAT	<u>57.77</u>	**90.37**	**70.48**
	Minimap2	**77.61**	41.44	54.03
	MHAP	56.37	46.58	51.01

First, we estimated the overlap length from the information contained in the output of the considered tool. For this estimate, we used the length of the sequences, and the start and end

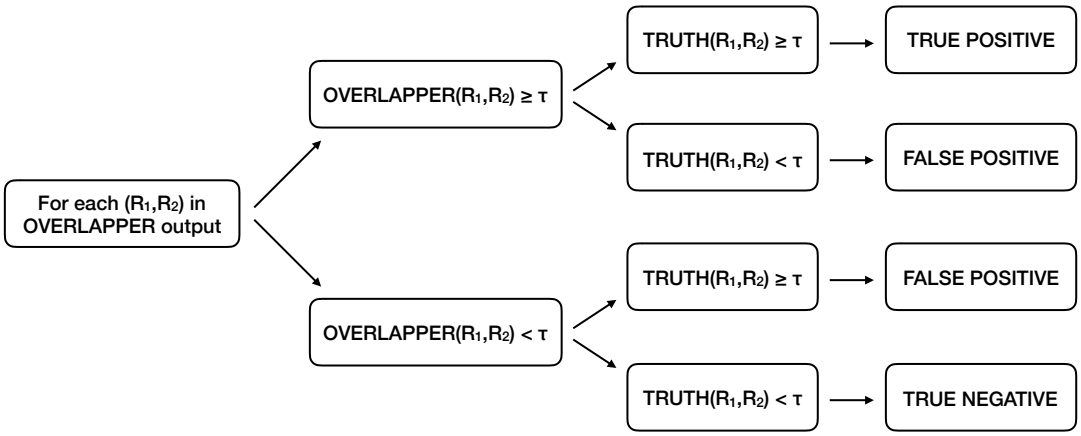

Figure 5: Diagram summarizing the evaluation procedure.

(R_1, R_2) represents a pair of sequences, while τ is the threshold defining a true-positive overlap. τ is set to $2,000$ bp in our default setting.

overlap position on each sequence. If the pair does not satisfy this condition, but the pair is overlapping according to the ground truth, the pair it then counted as a false positive. If the pair does not appear in the ground truth, it is then counted as a true negative. If the overlap estimate is greater or equal to τ, then the pair is counted as a true positive if it is an overlap according to the ground truth, otherwise it is counted as a false positive. Given that a tool can report multiple locations for a given read-read pair, the overlap estimation is computed on the longest overlap a read-read pair presents in the output file of the considered tool. This choice maximizes the retention of true positives. This way, each read-read pair can contribute at most one to the total number of true positives, and one to the total number of identified overlaps. Diagram in Figure 5 summarizes the above described evaluation procedure.

F Real Data Set

The procedure to generate the ground truth for real data is inspired by the one presented by Li (2016), with one main difference. Our procedure generates two different ground truth files for real data (instead of one as in Li's approach): a *single-mapped* file and a *multi-mapped* one; Li's ground truth coincides with our single-mapped file. We used single-mapped ground truths to obtain the results shown in Tables 2 and 3.

Following, we also consider the multi-mapped case because it is not always possible to be certain with real data, whether a read originated from one part of the genome or another, especially in the presence of repeats and high error rates. Hence, choosing a single mapping of a read could lead to the erroneous exclusion of valid read-to-reference alignments from the ground truth. It is our belief that a good overlapper/aligner should also be able to find correct overlaps that correspond to multi-mapped locations in the genome, as they could be useful for dealing with repeats in later stages of genome assembly. Therefore, the accuracy of each software was evaluated using the multi-mapped ground truth as well, as shown in Table 5. Naturally, the recall of any software would be lower (and its precision higher) if it were tested against a multi-mapped ground truth, compared to if it were tested against a single-mapped ground truth, since the former is a super-set of the latter.

We mapped sequences against the reference genome by using both BWA-MEM and Minimap2 as we wanted to use the tool that minimizes the amount of incorrect (or missing) mapping positions; erroneous mappings would negatively affect our evaluation on real datasets. Results shown in Table 6 suggest that Minimap2 may be a more suitable tool to map long reads to reference. Consequently, we evaluated the output of the overlappers against ground truth generated with Minimap2 to obtain the results.

G Ground Truth Generation for Real Data

As with the original procedure presented by Li (2016), reads were aligned against the reference genome using BWA-MEM `0.7.17-r1188` (Li, 2013) with the following commands:

```
(1) bwa index <reference>.fasta
(2) bwa mem -x pacbio <reference>.fasta <input>.fastq >
<output>.sam
```

When using Minimap2 to map sequences to reference, the command is the following:

```
(3) minimap2 -ax map-pb <reference>.fasta
<input>.fastq > <output>.sam
```

The outputs of both BWA-MEM and Minimap2 were filtered in order to exclude non-mapped reads and alignments with mapping-quality lower that 10. The filtering was done using Samtools (Li *et al.*, 2009) with the following command:

```
(4) samtools view -h -Sq 10 -F 4 <output>.sam >
<multi-mapped>.sam
```

The output `<multi-mapped>.sam` obtained from the above operation constitutes the multi-mapped ground truth. The single-mapped ground truth `<single-mapped>.sam` was obtained by applying a second filtering step, using the following command:

```
(5) samtools view -h <multi-mapped>.sam | grep -v -e
'XA:Z:' -e
'SA:Z:' | samtools view -S > <single-mapped>.sam
```

through which command sequences mapped to multiple locations of the genome are removed, outputting only single-mapped sequences.

H BWA-MEM versus Minimap2 Reference Alignment

Table 6 shows recall and precision of each tool when its performance is evaluated toward a ground truth generated using both BWA-MEM and Minimap2. Precision significantly increases when Minimap2 is used to generate the ground truth, while the loss in recall is negligible. Overall, the results suggest that Minimap2 is more effective in mapping sequences to reference, reducing the amount of missing alignments with respect to BWA-MEM. The author of both Minimap2 and BWA-MEM himself stated[1] that Minimap2 is better than BWA-MEM to align sequences to

[1] https://lh3.github.io/2018/04/02/minimap2-and-the-future-of-bwa

Table 6: Comparison between BWA-MEM and Minimap2 in mapping against reference.

The table shows recall and precision of whole set of tools when their outputs are compared against ground truth generated using Minimap2 (2nd and 3rd columns) or using BWA-MEM (4th and 5th columns). The last two columns reports the percentage change of recall and precision when Minimap2 is used as ground truth instead of BWA-MEM. Brackets refer to negative values.

		Minimap2		BWA-MEM		Change	
Data Set	Tool	Recall	Precision	Recall	Precision	Recall	Precision
E. coli (Sample)	BELLA	77.60	81.88	80.30	76.68	(3.36)	6.78
	DALIGNER	87.03	59.96	87.50	54.47	(0.54)	10.08
	BLASR	77.64	77.84	78.94	71.44	(1.65)	8.96
	MECAT	78.41	78.28	80.30	72.21	(2.35)	8.41
	Minimap2	93.28	71.38	93.80	64.78	(0.55)	10.19
	MHAP	79.71	66.36	80.93	60.72	(1.51)	9.29
E. coli	BELLA	64.31	68.83	68.55	61.91	(6.19)	11.18
	DALIGNER	80.39	51.99	82.59	44.99	(2.66)	15.56
	BLASR	35.43	71.13	37.71	63.50	(6.05)	12.06
	MECAT	54.61	72.18	58.56	64.84	(6.75)	11.32
	Minimap2	82.40	56.62	83.63	48.32	(1.47)	17.18
	MHAP	67.84	44.10	70.61	38.51	(3.92)	14.52
C. elegans 40X	BELLA	70.28	68.81	70.98	60.72	(0.99)	13.32
	DALIGNER	61.10	56.87	58.58	47.68	4.30	19.27
	BLASR	3.23	18.26	3.04	15.03	6.25	21.49
	MECAT	73.05	75.09	71.20	63.79	2.60	17.71
	Minimap2	95.18	33.43	92.47	28.32	2.93	18.04
	MHAP	69.72	37.87	71.87	34.01	(2.99)	11.35

reference when long-read data is used.

I Evaluation, Technical Details

The alignment results were obtained running BELLA in both sensitive and precise mode. In all our experiments, BELLA used $k = 17$ for uniformity; it is worth mentioning that k could possibly be fine-tuned to better match the user's goals.

The evaluation of the performance of the competing software was performed using MECAT `1.0`, Minimap2 `2.7`, BLASR `1.3.1. 12420`, MHAP `2.1.3`, and DALIGNER `1.0`. BLASR was run with the following settings: `-nproc 64 -maxLCP- Length 16 -minMatch 12 -m 4 -bestn 50 -noSplitSubreads`, where `nproc` specifies the number of threads used to run BLASR, `maxLCP Length` indicates that the alignment of a read to the reference is stopped when its length reaches, 16 in our case (this command is useful when the query is part of the reference, which is the case when computing pairwise alignment for *de novo* assembly), `minMatch` is the minimum seed length, `m` indicates which file format of the output is used, `bestn` is the number of best alignments reported, and `noSplitSubreads` does not split input sequences at adapters; MHAP was run with: `--settings 3 --store-full-id --num-threads 64`, where `3` is the "sensitive mode" (Berlin *et al.*, 2015) and `store-full- id` stores sequences by their original identifier instead of a numerical index; MHAP was also run using 64 threads, as indicated by `num-threads`.

We collected the results on a dual-socket computer with two 16-core Intel Xeon E5-2698 v3

Table 7: Peak memory usage for two representative data sets. The numbers are reported in GB.

Dataset	BELLA	DALIGNER	BLASR	MECAT	Minimap2
E. coli (Sample)	13.97	10.48	3.24	9.93	1.50
E. coli	27.08	19.76	6.52	19.98	4.70

("Haswell") processors, each running at 2.3 GHz with 128GB DDR4 2133 MHz memory, using 64-way thread parallelism on a single compute node, with few exceptions. Tools that could not run to completion for particular datasets with the available 128GB RAM were re-run on machines with the same processor model but larger RAMs (256GB and 512GB). In order to run MHAP `v2.1.3` on *C. elegans* 40X and *C. elegans* 20X, we increased the Java heap space to fit our largest machine, since the default 32 GB setting (used for all the other datasets) led to out-of-memory failures. Additionally, we collected the results from DALIGNER on *C. elegans* 40X using 32 threads; DALIGNER `v1.0` with DAZZDB `v1.0` encountered segmentation faults due to memory overflow at 64 threads (-T64), even with the available parameters adjusted for minimal per-process memory footprint (without externally forcing dataset reduction) and using our largest-memory machine.

Finally, DALIGNER results are only reported for real PacBio data because it requires single cell[2] PacBio sequencing data (Myers, 2014). For all of the simulated datasets, DALIGNER failed-fast, producing the associated error, "`Pacbio header line format error`". Our attempts to reformat the simulated data (to "fool" DALIGNER)[3] failed with the same error.

J Memory Usage

The memory usage of each software tool was measured using Valgrind 3.13.0 (Nethercote *et al.*, 2006), with the following command: `valgrind --tool=massif --pages-as-heap=yes`. Besides the necessary use of `-g`, there were no changes to the compilation of any tool over that described in Section I, with one exception. MHAP was compiled with the addition of `java -Djava.compiler=NONE`. Further, the memory usage data was collected on the same compute nodes as those in Section I, with the largest available RAM. As in Section I, DALIGNER encountered early segmentation faults when run with `-T64`, due to its per-thread memory demands. The results shown were collected with `-T32`. Table 7 shows the final results, except for MHAP, as it exceeded the maximum time limit of 48h.

K Scalability

Figure 6 shows the strong scaling curves of BELLA for two representative data sets to measure its parallel performance. The data was collected on the same compute nodes as those in Section I. The 64-way thread parallelism is achieved exploiting hyper-threading as each node has 32 cores and each of them supports 2 hyper-threads.

In strong scaling, the problem size stays fixed but the number of processing elements are increased. In strong scaling, a program is considered to scale linearly if the speedup is equal to the number of processing elements used, corresponding to the number of threads in our case. Given the amount of time to complete a given run using one thread t_1 and the amount of time to complete the same run with n threads is t_n, we can define the strong scaling efficiency (as a percentage of linear) as:

[2]https://github.com/thegenemyers/DAZZ_DB/issues/10
[3]https://github.com/PacificBiosciences/FALCON/issues/27

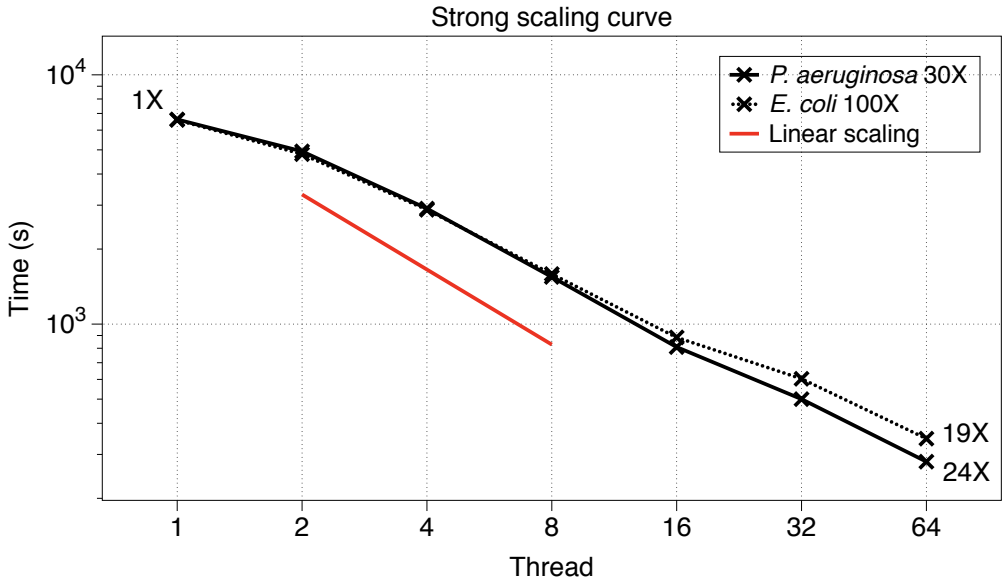

Figure 6: The figure shows how BELLA times vary varying the number of threads for two sample data sets.

$$\frac{t_1}{(n \cdot t_n)} \cdot 100 \tag{17}$$

BELLA with 64-way thread parallelism had a strong scaling efficiency of 37% and a speedup of 24× for *P. aeruginosa* 30X, while its strong scaling efficiency and speedup were 30% and 19× for *E. coli* 100X.