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 reconstructing genomes from DNA fragments (reads), which may contain redundancy and errors. Longer reads simplify assembly and improve contiguity of the output, but 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), which implement a novel approach to compute overlaps using Sparse Generalized Matrix Multiplication (SpGEMM). We present a probabilistic model which demonstrates the soundness of using short, fixed length k-mers to detect overlaps, avoiding expensive pairwise alignment of all reads 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 happen with highly erroneous reads and the spurious overlaps due to k-mers originating from repetitive regions. Finally, we present a new method to separate true alignments from false positives depending on the alignment score. Using this methodology, which is employed in BELLA's *precise mode*, the probability of false positives drops exponentially as the length of overlap between sequences increases. On simulated data, BELLA achieves an average of 2.26% higher recall than state-of-the-art tools in its *sensitive mode* and 18.90% higher precision than state-of-the-art tools in its *precise mode*, while being performance competitive.

1 Introduction

Recent advancements in sequencing technologies have made large-scale genomic data more accessible, promoting a new, data-driven approach to the study of biology. 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 [33], which is the process of aligning and merging DNA fragments in order to reconstruct the original sequence. More specifically, *de novo* genome assembly reconstructs the genome from the sequenced DNA without guidance from a reference genome.

De novo assembly guarantees that the full range of genetic variation of a specific organism will be assessed; thus enabling the identification of individual-specific genetic features in a way that is not biased by any reference [31]. Furthermore, *de novo* assembly can generate new sequence assemblies for previously uncharacterized genomes [31].

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 [3] is their inability to deal with consistent accuracy with genomic repeats longer than

the read length, resulting in fragmented assemblies with ambiguous placements of sequences [30, 28]. Conversely, long-read technologies, offered by companies Pacific Biosciences (PacBio) with their Single-Molecule Real Time (SMRT) sequencing [12] and Oxford Nanopore Technologies with their Nanopore sequencing [15], 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 and leading to new disease models with improved predictive power. 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 [14] compared to short-read technologies.

While the assembly of short reads typically relies on the De Brujin Graph (DBG) abstraction, an overwhelming majority of the state-of-the-art long read assemblers uses the Overlap-Layout-Consensus (OLC) paradigm [4]. This is partly because longer reads mean that significantly fewer reads are required to cover the genome (by a factor of 100X), hence limiting the size of the overlap graph. In OLC assembly, the first step is the detection of overlaps between reads to construct an *overlap graph*. Highly-accurate overlap detection is a major computational bottleneck of the OLC assembly pipeline [27], 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 [8].

The process of identifying a set of initial candidate read pairs, sometimes simply known as *overlapping*, effects both the accuracy and the algorithm runtime. High error rates complicate initial overlapping, because it becomes hard to separate signal from the noise. Therefore, it is crucial to develop a solid mathematical approach to identify initial candidate read pairs. This allows us to (a) not waste computational resources, and (b) not omit informative data whose exclusion could lead to significant losses in terms of accuracy. In addition, computationally efficient and highly accurate overlapping and alignment algorithms have the potential to improve existing long-read assemblers, enabling reference assemblies and detection of genetic variations of higher quality.

The main contributions of this work are the following:

1. We demonstrate the soundness of using a k-mer seed-based approach for accurately identifying initial candidate read pairs through a mathematical approach that relies on the Markov chain model [25].

2. We include a simple procedure for pruning k-mers and prove that it retains nearly all valuable information 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 [11, 34] to attain BELLA, all-to-all long-read aligner. In its sensitive mode, BELLA achieves high recall with negligible losses of precision. BELLA's precise mode uses a new method based on Chernoff bounds to separate true alignments from false positives depending on the alignment score. In this mode, we show that the probability of false positives drops exponentially as the length of overlap between sequences increases.

2 Methods

2.1 Proposed Algorithm

The proposed work is a computationally 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 details.

BELLA uses a seed-based approach to detect overlaps in the context of *de novo* assembly; such approach partitions 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 show the feasibility of using a k-mer seed based approach for overlap detection of long-read data with high error rates (Section 2.2).

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. Similarly, k-mers that occurs 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 [18, 24].

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 retainment of k-mers that belong to unique regions of the genome, using a probabilistic analysis given the error rate and the sequencing depth (Section 2.3).

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 (Section 2.4).

Overlap detection is implemented in BELLA using Sparse Generalized Matrix Multiplication (SpGEMM), which allows our algorithm to achieve fast overlapping without using approximate approaches (Sections 2.5). 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 [9, 2].

BELLA's overlap detection has been coupled with an optimized state-of-the-art seed-and-extend algorithm [11, 34], 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. BELLA implements two different modes for the alignment step. The *sensitive mode* is fast and achieves high recall, while the *precise mode* is slightly slower but achieves significantly higher precision. The precise mode uses a new method to separate true alignments from false positives depending on the alignment score. In precise mode, we prove that the probability of false positives decreases exponentially as

the length of overlap between reads increases. The methods underlying sensitive and precise modes are described in Sections 2.6 and Section 2.7, respectively. 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. Randomized, multiple 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. [4] 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 not a lossless 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.

Section 2.8 describes the evaluation design implemented to obtain the results presented in Section 3, which are discussed in Section 4.

2.2 Overlapping Feasibility

Chaisson and Tesler [6] have proposed a theory on 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 with geometric length, each separated by an error [14]. The result obtained from their theory is the number of bases to be sequenced 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:

$$p = (1 - e)$$

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 [25].

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.

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

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

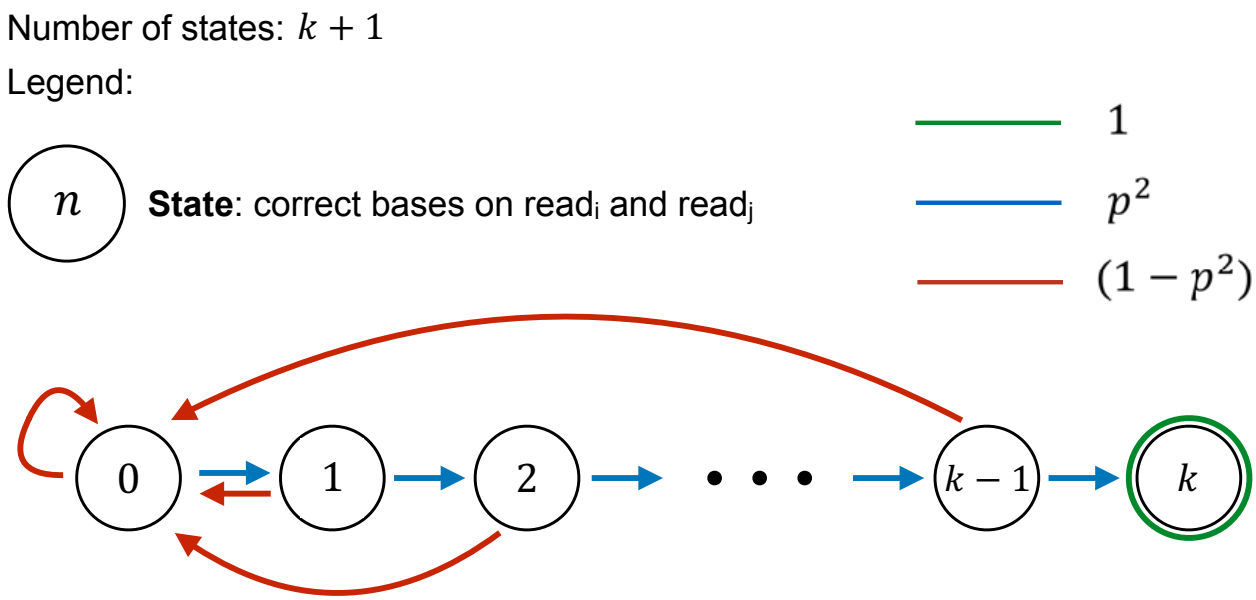

Figure 1: Proposed Markov chain model.

The figure describes the proposed model to prove the feasibilioty of using short k-mers for overlap detection. Each state n contains the number of successes obtained so far both on one read and on the other one. The arrows represent the transition probabilities.

Then, one can find the probability of being in any of the states after L steps in the Markov chain by computing the Lth power of the matrix $\mathbf{P}$. More efficiently, one can just compute $\mathbf{v}\mathbf{P}^{\mathrm{L}}$, 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 as its length increases. However, as error rate decreases a larger k would be preferable since it would decrease the amount of k-mers coming from repetitive regions of the genome.

2.3 Reliable k-mers

Repetitive regions of the genome lead to k-mers occurring frequently in the input read set. 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 high-frequency k-mers enforce spurious alignment attempts. 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

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 ← (k + 1)
    P ← 0                                   ▷ Entire matrix initialized to 0
    for i ← 0 to states do
        P[i, 0] ← (1 − p^2)
        P[i, i + 1] ← p^2
    end for
    P[states, states] ← 1
    v ← (1, 0, . . . , 0)                   ▷ Initialized to standard unit vector
    for i ← 0 to L do                       ▷ Compute vP^L without exponentiation
        v ← vP
    end for
    return v[states]
end procedure
```

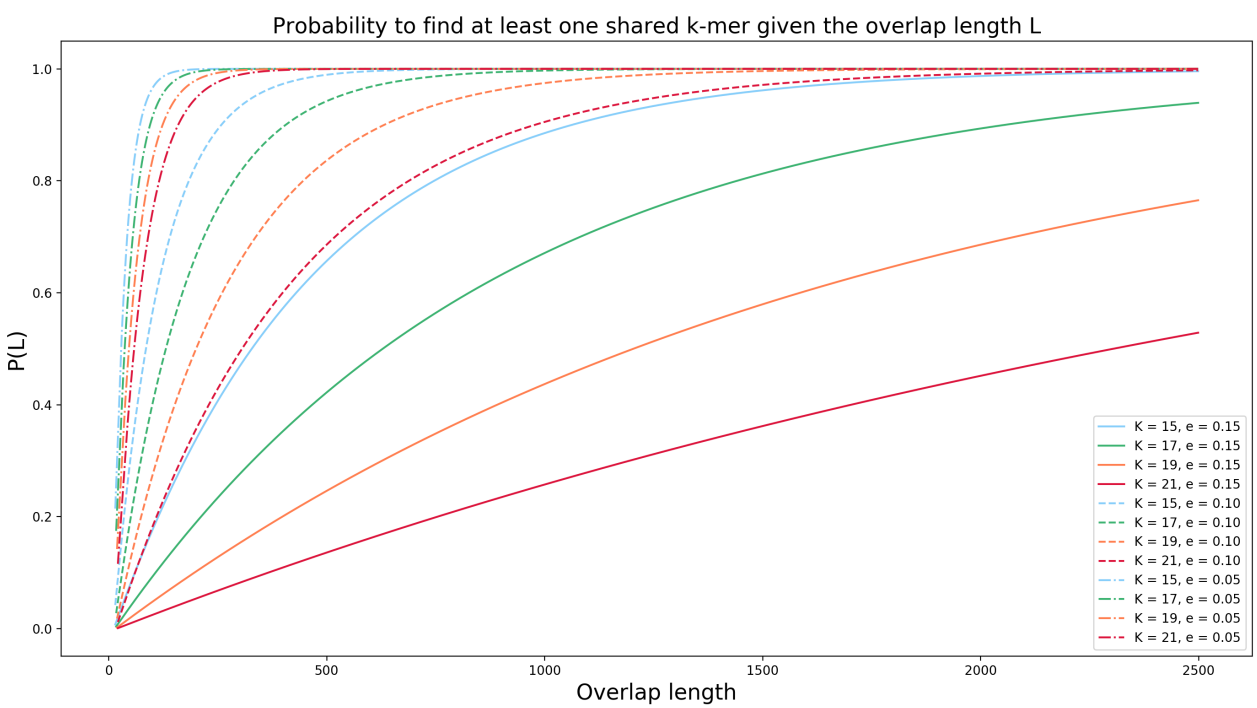

Figure 2: Outcome of the proposed Markov chain model.

The probability of having one correct k-mer on two sequences given an overlap region and the error rate. 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. The probability of at least one correct k-mer between two sequences decreases increasing the k-mer length. However, smaller error rates allow to use larger values of k.

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 Li [24], 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.

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:

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

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. In that sense, BELLA's approach to select reliable k-mers diverges sharply from how Lin et al. [24] 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 histogram in Figure 3 shows the distribution of k-mers in the genome given their occurrence in the input set of reads. In order to obtain the histogram, we mapped k-mers from the input reads to the reference genome. Please note that the y-axis is reported in logarithmic scale and this can create some initial misreadings. For instance, an overwhelming majority of k-mers that occur twice are actually non-genomic even though the bar is mostly green. The histogram shows that the majority of k-mers in the right tail either occur multiple times or do not occur in the genome at all. 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.

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:

$$B(n = d, p = (1-e)^k)$$

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

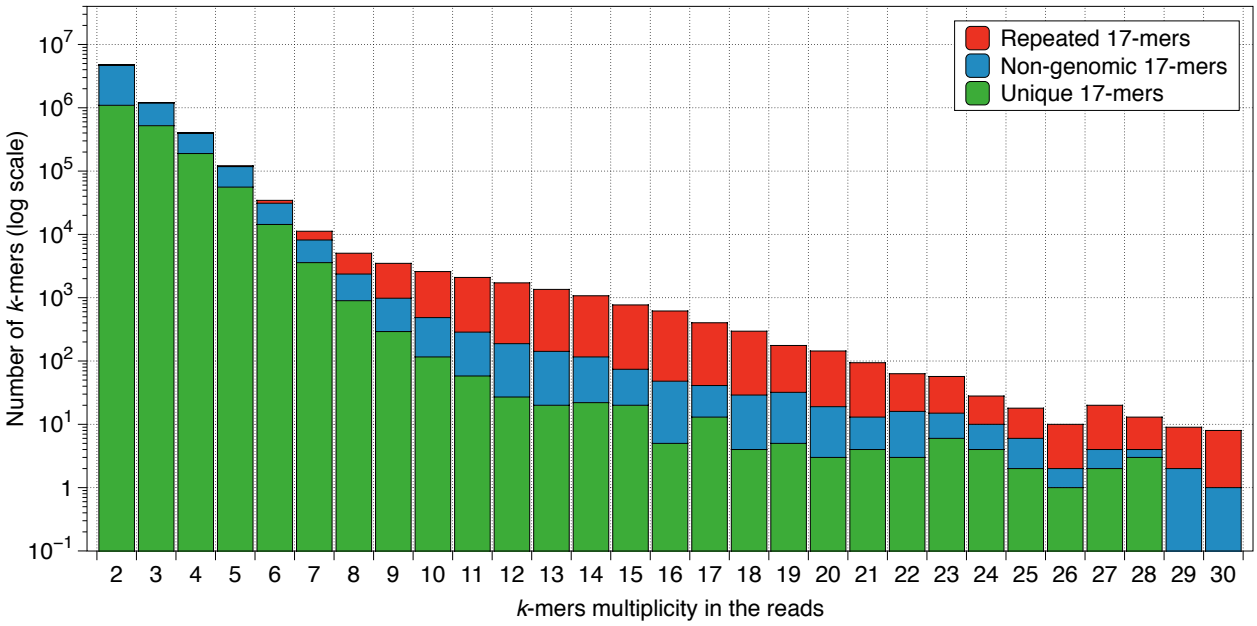

Figure 3: Distribution of k-mers in the genome given their occurrence.

The histogram 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 $e = 0.15$). The height of each bar in the histogram represents the number k-mers with that specific frequency. We would like to emphasize that the y-axis is logarithmic. Different colors identify the amount of unique, non-genomic, and repeated k-mers for each frequency.

$$Pr(m; d, (1-e)^k) = \binom{d}{m}(1-e)^{km}(1-(1-e)^k)^{(d-m)} \tag{2}$$

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 2 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 2 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.

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

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

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

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.

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

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

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 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.

2.4 BELLA k-mer Counting

The first step of BELLA's implementation is a high-performance multithreaded k-mer counter, which enables an efficient k-mer filtering based on the reliable k-mers model. A Bloom filter, a common optimization structure in k-mer counting [26], is used in combination with a multi-threaded cuckoo hash table [22]. An accurate sizing of the Bloom filter is crucial to avoid a dynamic resizing of the data structure, as this would negatively affect performance by increasing the runtime of BELLA. Therefore, we use the HyperLogLog cardinality estimator [13] to compute approximate counts for the number of unique entries (in our case k-mers) in the dataset.

The final output of the k-mer counting step is the construction of a k-mer dictionary where every entry is unique and each k-mer is associated with its occurrence in the input set.

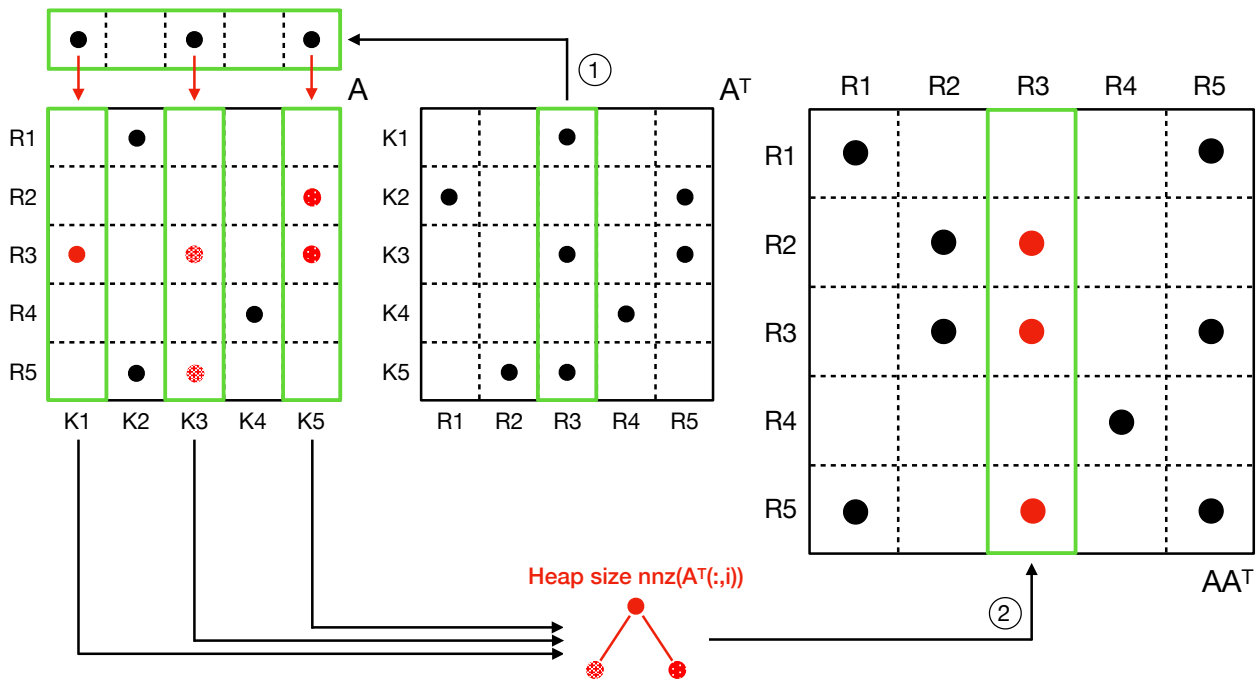

Figure 4: Example of column-by-column sparse matrix multiplication [2].

$\mathbf{A}^\mathsf{T}(:, R_3)$ 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 $\mathbf{A}^\mathsf{T}$. In the example shown here, the non-zero elements of column R_3 of $\mathbf{A}^\mathsf{T}$ are in rows K_1, K_3, and K_5. Therefore, columns K_1, K_3, and K_5 of A will be selected for the sparse matrix multiplication. Considering $\mathbf{A}(:, k_1)$ at first, the heap at index $i = R_3$ stores the result of the operation applied to the element contained in $\mathbf{A}(R_3,k_1)$ and the one contained in $\mathbf{A}^\mathsf{T}(k_1,R_3)$. This procedure is repeated on $\mathbf{A}(:, j)$, where j is the index of the columns defined by the active column of $\mathbf{A}^\mathsf{T}$. If there are multiple elements with the same row-index (R_3, in this case) in the considered batch of columns of $\mathbf{A}$, their values are summed on the fly. Finally, the contents of the heap are stored in the same sparse format as the input into a column of the final matrix once all required columns are accumulated.

2.5 Sparse Matrix Creation 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. It saves just 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 an $|reads|$-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 [10].

The SpGEMM algorithm in Figure 4 is functionally equivalent to a k-mer based seed-index table, which is common in many long-read alignment software. 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 4 illustrates the merging operation of BELLA, which uses a priority queue (heap) data structure indexed by the row indexes of $\mathbf{A}$, following the multi-threaded implementation proposed by Azad et al. [2]. Finally, the contents of the heap 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 [17]. 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 [5].

Increasing the genome size also increases the RAM requirements to build the final sparse matrix. For this reason, BELLA writes the intermediate results of the multiplication to file, avoiding the construction of the final sparse matrix in memory. The transpose matrix $\mathbf{A}^\mathsf{T}$ is divided into batches of columns, based on the available RAM. Given that the multiplication is multi-threaded, each thread owns a pointer to $\mathbf{A}$ and one to a batch of $\mathbf{A}^\mathsf{T}$. When a thread terminates, the results of the thread are written to a shared file, producing the output file of BELLA.

Currently, there are no known specialized SpGEMM implementations for $\mathbf{AA}^\mathsf{T}$ that stores and operates only $\mathbf{A}$, 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 do not produce a nonzero in the output. By contrast, our column-by-column implementation runs faster than $O(|reads|^2)$ for when the matrix is sufficiently sparse.

2.6 Sensitive Alignment

The overlap detection step of the *de novo* assembly pipeline tends to sacrifice precision in favor of recall. It is important for an overlap detector to achieve high recall, as one cannot recover lost data but can always filter further if needed. Instead of making the overlap filtering step more stringent, BELLA applies an optimized pairwise alignment for pairs that pass the overlapping step. From empirical observations, it was noted that the number of shared k-mers between reads is not a feature informative enough to predict accurate overlaps on its own.

To improve precision, and therefore not to waste computational power in the subsequent stages of *de novo* assembly, in BELLA a seed-and-extend alignment algorithm was coupled with our overlap detection stage. In contrast to other fast overlapping software like Minimap and MHAP that use sketches and minimizers, our algorithm can perform a seed-and-extend procedure thanks to its explicit k-mer based overlapping. The algorithm integrated within BELLA is based on an x-drop implementation proposed by Zhang et al. [34] and belonging to the SeqAn library [11], a high-performance `C++` library for sequence analysis. However, this algorithm can easily be changed in the future as more advanced pairwise alignment algorithms, such as the Suzuki-Kasahara [32]

algorithm published recently, are developed.

In BELLA, at most two seed k-mers are given as input to the x-drop alignment algorithm, even if the read pair shares more k-mers. If the pair shares more than two k-mers, two representative k-mers are chosen at random. 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 (as will be shown in Section 3), suggesting that using two seeds chosen at random for x-drop alignment represents a good compromise between accuracy and computational cost. Further shared k-mers are neglected: for instance, if a pair shares 10 k-mers, only the position of two k-mers (chosen at random) is saved and used in the seed-and-extend alignment step.

For each read pair in the overlap matrix, the alignment is extended from one- or two-seed k-mers until the running alignment score drops of x points below the best score seen so far. For example, if the best score seen were 200 base pair (bp) and $x = 3$, the algorithm would stop the alignment when the score became equal to, for example, 197 bp. The errors could be non-consecutive: 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 default values of x-drop and minimum alignment score, which are user-tunable, are equal $x = 3$ and $n = 50$, the same default values of the original alignment algorithm.

These values were used to define the sensitive mode of BELLA whose whose effect on increasing recall while keeping runtine low are presented in Section 3.

Close attention to the meaning of alignment score has led us to the formulation of a rule to filter out false-positive alignments. A pair of overlapping sequences could obtain an alignment score high enough to pass the threshold despite being a false positive. This is especially true within highly repetitive genomes where a repetitive region is expected to be longer than $n = 50$ bp. We argue that an alignment corresponding to a true overlapping pair should extend until the end of the two sequences; we defined such an alignment as "alignment on edge". An alignment is strictly defined "on edge" if it ends (a) on both ends of $read_1$, (b) on both ends of $read_2$, or (c) on one end of $read_1$ and on another end on $read_2$.

The high error rates ($\approx 10 - 15\%$) of long-read data have led us to consider relaxations of the above-mentioned definition. First, we have introduced a parameter ϵ, for which if an alignment ends within ϵ bases of the end of a read it is defined "approximately extending" to the end of the read, and therefore it is considered an alignment on edge. Second, if the alignment ends on only one end of a read, the pair is still classified as true overlapping. As an additional relaxation, it would be possible to apply the filtering rule only when the evidence of overlap is weak; for example, when an overlapping pair is supported by a single shared k-mer.

This constraint is implemented in BELLA as an optional setting and it improves the precision of BELLA with negligible loss in recall, as shown in Table S2.

The output of BELLA is similar to that of BLASR/MHAP [6, 4], which is in `.m4` format. For each pair of reads that passes both the overlapping stage and the alignment stage, the output includes a line with: (a) the identifiers of the reads, (b) the number of shared k-mers, (c) the alignment score, (d) the strand information (n if the reads belong to the same strand, c if they are not, following DALIGNER [27] convention), (e) the start and end positions of the alignment in the first read, (f) the start and end positions of the alignment in the second read, (g) the lengths of the reads. If a pair has two shared k-mers and the alignment score of both seeds passes the alignment thresholds, the seed with the highest alignment score is chosen as the only output.

2.7 Precise Alignment

The sensitive mode seed-and-extend alignment improves our precision with negligible losses in terms of recall. However, BELLA's recall and precision are tunable based on the length of k, the x-drop value, and the alignment score threshold. As we believe a user could benefit from a different setting, we propose arguments in favor of more strict alignment settings and introduce the *precise mode* of BELLA.

In the following, we consider a typical high-throughput PacBio sequencing experiment where the raw long-read data contains $\approx 10 - 15\%$ errors that are randomly distributed. The majority of these errors are indels, which are erroneous insertions or deletions of nucleotides [14]. We argue that an x-drop factor excessively small, such as $x = 3$ in the sensitive mode, 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.

Furthermore, a fixed alignment threshold might not capture the real alignment behavior. 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 correct matches within that overlap region, with $m = p \cdot L$. The alignment score χ, corresponding to the edit distance, can be written as follows:

$$\chi = \alpha m - \beta(L - m) = \alpha p L - \beta(L - pL) \tag{3}$$

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:

$$\varphi = \frac{\chi}{L} = \alpha p - \beta(1 - p). \tag{4}$$

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 [7, 16] 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, as described in Section 2.8) when using the above-defined cutoff.

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 [1], which is defined for $0 \leq \delta \leq 1$ as:

$$\mathbf{Pr}[Z \leq (1 - \delta)\mu_z] \leq e^{\frac{-\delta^2 \mu_z}{2}}. \tag{5}$$

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

$$X_i = \begin{cases} \alpha, & \text{with probability } p \\ -\beta, & \text{with probability } 1-p \end{cases} \tag{6}$$

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. 4. 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:

$$Z_i = \frac{X_i + \beta}{\alpha + \beta}, \text{ where } Z_i \in \{0, 1\}. \tag{7}$$

From the linearity of expectation, we have

$$E[Z] = E\left[\frac{X + \beta}{\alpha + \beta}\right] = \frac{E[X] + \beta L}{\alpha + \beta} = \frac{(2p-1)L + \beta L}{\alpha + \beta}. \tag{8}$$

Substituting Eq. 7 and Eq. 8 in Eq. 5 and simplifying using our scoring matrix of $\alpha, \beta = 1$, we obtain the final expression:

$$\mathbf{Pr}[X \leq (1-\delta)\mu_x] \leq e^{-\delta^2 pL}, \text{ with } E[X] = \mu_x. \tag{9}$$

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 10% ($\delta = 0.1$) is $\leq 5.30 \times 10^{-7}$. This bound assumes exact alignment (i.e., an x-drop value $x \to \infty$). Nevertheless, the Markov Chain model introduced in Section 2.2 proves that the probability of missing a true alignment due to x-drop is extremely small. Notably, that probability of having x consecutive errors that would force alignment to stop prematurely is $\approx 2.96 \times 10^{-5}$ for $x = 7$ when considering an overlap region $L = 2000$ bp and an error rate of 15%.

An x-drop value of $x = 7$ and an adaptive threshold derived from the scoring matrix and the cutoff rate with $\delta = 0.1$ define the precise mode of BELLA. This setting resulted in significant increase in precision, allowing BELLA to achieve the highest values of precision with respect to state-of-the-art software while maintaining high values of recall. These benefits come at the cost of higher runtime with respect to the sensitive mode, as presented in Section 3.

2.8 Evaluation Design

Table 1: The data sets used for evaluation. PBSIM is the PacBio read simulator [29].

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://goo.gl/Z75V5R (CBCB)	266 MB
E.coli	*Escherichia coli* MG1655 strain	PacBio RS II P5-C3 30X	https://goo.gl/mhp9ud (NCBI)	791 MB
C.elegans 40X	*Caenorhabditis elegans* Bristol mutant strain	PacBio RS II P6-C4 40X	https://goo.gl/iRvL4v (PacBio)	8.90 GB
P. aeruginosa	*Pseudomonas aeruginosa* PAO1	PacBio PBSIM CLR 30X	https://goo.gl/2M7ktK (NCBI)	359 MB
V. vulnificus	*Vibrio vulnificus* YJ016	PacBio PBSIM CLR 30X	https://goo.gl/ztxX9B (NCBI)	288 MB
A. baumannii	*Acinetobacter baumannii*	PacBio PBSIM CLR 30X	https://goo.gl/Y89rKv (NCBI)	248 MB
C.elegans 20X	*Caenorhabditis elegans*	PacBio PBSIM CLR 20X	https://goo.gl/Y2gXZx (NCBI)	3.75 GB

The datasets used for evaluation are listed in Table 1, which includes 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. We selected genomes with varying size and complexity as the outcome is often dependent on these features of the genome [23].

As performance metrics, we used recall, precision, 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). More formally,

$$recall = \frac{TP}{TP + FN}, \quad precision = \frac{TP}{TP + FP},$$

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 [20].

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; conversely, the ground truth generation procedure differs for the two categories of data.

Real Data Set The procedure to generate the ground truth for real data is inspired by the one presented by Li [20], with a 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 also consider the multi-mapped case because it is not always possible to be certain whether a read originated from one part of the genome or another, especially in the presence of repeats and high error rates. Hence, picking a single mapping of a read could lead to the erroneous exclusion of valid read-to-reference alignments from the ground truth, due to sequencing errors and genome repetitiveness. It is our belief that a good overlapper/aligner should be able to also find correct overlaps that correspond to multi-mapped locations in the genome, as they could be useful for dealing with repeats in the later stages of genome assembly. Therefore, the accuracy of each software was evaluated separately using single-mapped and multi-mapped ground truth files. 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 want 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 Supplementary Material S1 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 truths generated with Minimap2 to obtain the results shown in Section 3. Commands to generate the ground truth for a real dataset with both BWA-MEM and Minimap2 are reported in Supplementary Material S3.

Synthetic Data Set Generating a ground truth is simpler when dealing with synthetic data, and we accomplished it by running the read simulator of Pacific Biosciences, PBSIM [29]. 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 a 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 true positives count: 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 (defined as reads that aligns to themselves). Subsequently, we used the information stored in the ground truth to discriminate if a read pair is a true positive or not. As previously defined, a read pair is considered a true positive if the two sequences align for at least 2 kb, considering the start and end positions of their alignment in the reference genome (reported in the ground truth). If the pair does not satisfy this condition, 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 start and end overlap position on each sequence. 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. If the estimate is smaller than 2 kb, we discarded the considered pair from the total amount of overlaps identified by the tool. This is because even if the overlap were real, the ground truth would not consider it as a true positive. This way, each read-read pair can contribute with at most one true positive to the total number of true positives, and with one increment to the total amount of identified overlaps.

2.9 Experimental Settings

The alignment results were obtained running BELLA in both sensitve 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 goal. Furthermore, BELLA has been run with different input values of error rate on real datasets since error rate for these is not unknown and it is a key factor in choosing the amount of reliable k-mers used during overlap detection.

The evaluation of the performance of the competitor software was performed using Minimap version `0.2`, 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 -maxLCPLength 16 -minMatch 12 -m 4 -bestn 50 -noSplitSubreads`, where `nproc` specifies the number of threads used to run BLASR, `maxLCP Lenght` 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" [4] 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 ("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 one exception: we collected the results

Table 2: Results for four synthetic PacBio datasets.

The data has been simulated with an error rate of 15% using the PacBio read simulator PBSIM [29]. The fields included are data set name, alignment tool, recall, precision, running time, and if the considered aligner does actual alignment or just overlap detection. See Table 1 for further information on each data set. DALIGNER failed on all simulated data sets for reasons further described in 2.9, and hence is not included in this result set. Precision and recall values are reported in percentage.

Data Set	Tool	Recall (%)	Precision (%)	Time (s)	Alignment
P. aeruginosa	BELLA `sensitive mode`	**99.63**	47.53	119.86	Y
	BELLA `precise mode`	99.59	89.05	630.00	Y
	BLASR	87.39	**90.51**	553.50	Y
	Minimap	76.63	86.25	**17.87**	N
	Minimap2	99.10	72.40	19.06	N
	MHAP	72.75	63.45	132.38	N
V. vulnificus	BELLA `sensitive mode`	**99.44**	56.61	69.29	Y
	BELLA `precise mode`	99.31	83.21	129.93	Y
	BLASR	87.69	**84.74**	363.59	Y
	Minimap	76.34	61.98	16.17	N
	Minimap2	96.72	74.02	**12.66**	N
	MHAP	74.59	45.17	126.89	N
A. baumannii	BELLA `sensitive mode`	**99.58**	56.45	61.81	Y
	BELLA `precise mode`	99.47	83.92	127.11	Y
	BLASR	89.77	**84.56**	325.50	Y
	Minimap	79.69	83.96	11.67	N
	Minimap2	96.90	71.63	**9.42**	N
	MHAP	76.95	28.84	131.64	N
C. elegans 20X	BELLA `sensitive mode`	**98.63**	47.05	1,636.95	Y
	BELLA `precise mode`	97.70	**86.88**	4,384.55	Y
	BLASR	95.75	78.29	7,919.94	Y
	Minimap	81.40	78.15	351.54	N
	Minimap2	95.77	70.62	**209.21**	N
	MHAP	82.64	6.43	6,218.39	N

from DALIGNER on *C. elegans* 40X using 32 threads for the following reason. 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 machine. 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. 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). DALIGNER results are only granted for real PacBio data because it requires single cell[1] PacBio sequencing data (Myers, 2014). For all of the simulated datasets, DALIGNER failed quickly, producing the associated error "`Pacbio header line format error`". Our attempts to reformat the simulated

[1] https://github.com/thegenemyers/DAZZ_DB/issues/10

data (to "fool" DALIGNER)[2] failed with the same error.

3 Results

BELLA in both sensitive mode and precise mode and the other software were run on four different PacBio datasets simulated with an error rate of 15% using PBSIM [29]. Table 2 shows that BELLA's sensitive mode had the highest values of recall on all datasets, with an average 2.26% improvement over Minimap2 which is the main competitor. The precise mode of BELLA had similar values of recall to its sensitive mode, but it showed an average 66.91% improvement in precision over its sensitive mode. BELLA's precise mode had the best precision on *C. elegans* 20X; on the remaining datasets, BLASR had an average 1.41% improvement of precision over BELLA's precise mode, which, however, showed on average values of recall 10.02% higher than BLASR and 2.40× speedup, except for *P. aeruginosa* dataset where BLASR was 1.14× faster. On average, BELLA's precise mode had an improvement of 1.96% in recall and of 18.90% in precision over the main competitor, Minimap2. Minimap2 was the fastest tool with the whole set of data.

Furthermore, BELLA and other software were run on the three real PacBio datasets and were evaluated by using both multi-mapped and single-mapped ground truth. In Table 3, the parameter e reported alongside BELLA represents the assumed error rate of the dataset. Given that we do know the exact error for real data, we tested BELLA with different input values of e as this parameter influences the amount of reliable k-mer BELLA uses to detect overlaps. BELLA's sensitive mode with $e = 0.11$ achieved the highest values of recall with the whole set of data with an average improvement over the second-best tool of 2.42% when tested against single-mapped ground truth and of 3.41% when tested against multi-mapped ground truth. Using the single-mapped ground truth, BELLA's precise mode had the best precision with $e = 0.15$ on *E. Coli* (Sample) and the highest precision on the remaining datasets with $e = 0.11$. For the multi-mapped case, BLASR had the highest precision on the *E. Coli* dataset with an improvement of 0.55% over the precision BELLA's precise mode $0.11e$; here, BELLA outperformed BLASR on *E. coli* (Sample) with of 3.19%. BELLA's precise mode $0.15e$ had a precision 19.34% higher than Minimap2 on average on single-mapped ground truth and 7.90% when compared against multi-mapped ground truth; Minimap2 had on average 38.43% higher recall than BELLA's precise mode $0.15e$ compared against single-mapped ground truth and 51.90% compared against multi-mapped ground truth. The precision of BELLA's sensitive mode for both real and synthetic data is moderate compared to that of Minimap2, but it is always within a factor of 2 from the best tool compared. Minimap was the fastest software on both the *E. coli* (Sample) and the *E. coli* dataset, while Minimap2 had the best runtime on *C. elegans* 40X.

In Table 4, we evaluated the filtering performance of BELLA for three representative datasets. Table 4 compares the number and the percentage of candidates discarded by BELLA before and after the alignment step to the squared value of the read number; this number represents the amount of all-to-all pairwise alignments. The overlap detection procedure of BELLA was able to filter over 99% of all read pairs without performing pairwise alignment on all the datasets.

4 Discussion

BELLA proposes a computationally efficient and highly accurate approach for overlapping and aligning noisy long reads, based on novel mathematical models that minimize the cost of overlap

[2] https://github.com/PacificBiosciences/FALCON/issues/27

Table 3: Results for three real PacBio datasets.

The fields included are data set short-name, alignment tool, recall, precision, running time, and whether the considered aligner does actual alignment or just overlap detection. The parameter e reported alongside BELLA represents the data set error rate. When the exact error rate was not known, BELLA was tested with different settings as the value of e affects the amount of reliable data used to find overlaps, as explained in Section 2.3. Precision and recall numbers are reported for both multi-mappend and single-mapped ground truth and are expressed in percentages. See Table 1 for further information on each data set. *The results were collected using 64-way thread parallelism on a single compute node, except DALIGNER's on *C.elegans* which were collected with 32 threads.

		Multiple Mapping		Single Mapping			
Data Set	Tool	Recall	Precision	Recall	Precision	Time (s)	Alignment
E. coli (Sample)	BELLA 0.15e `sensitive mode`	93.19	74.97	98.30	61.10	47.32	Y
	BELLA 0.13e `sensitive mode`	93.38	73.83	98.45	60.14	46.47	Y
	BELLA 0.11e `sensitive mode`	**93.52**	71.99	**98.56**	58.60	48.42	Y
	BELLA 0.15e `precise mode`	61.46	97.47	71.71	**88.23**	117.39	Y
	BELLA 0.13e `precise mode`	54.38	98.04	66.99	84.90	122.43	Y
	BELLA 0.11e `precise mode`	46.85	**98.46**	58.78	86.84	124.05	Y
	DALIGNER	86.29	77.18	92.17	63.60	24.24	Y
	BLASR	73.24	95.42	79.90	80.18	301.72	Y
	Minimap	84.46	94.20	91.09	78.05	**14.42**	N
	Minimap2	89.64	89.23	95.58	73.28	14.77	N
	MHAP	73.85	80.16	80.84	67.32	54.79	N
E. coli	BELLA 0.15e `sensitive mode`	85.28	82.68	91.50	67.27	202.63	Y
	BELLA 0.13e `sensitive mode`	86.55	81.57	92.69	66.26	206.58	Y
	BELLA 0.11e `sensitive mode`	**88.89**	79.65	**94.81**	64.49	215.15	Y
	BELLA 0.15e `precise mode`	54.92	97.83	64.93	87.98	674.37	Y
	BELLA 0.13e `precise mode`	49.15	98.27	58.75	89.42	676.36	Y
	BELLA 0.11e `precise mode`	43.29	98.61	52.28	**90.80**	709.00	Y
	DALIGNER	78.24	76.06	83.58	61.61	134.42	Y
	BLASR	31.81	**99.15**	36.19	84.93	1,719.80	Y
	Minimap	80.65	95.27	87.80	78.31	**52.71**	N
	Minimap2	86.74	91.38	93.22	74.38	72.12	N
	MHAP	70.47	79.37	77.74	66.06	205.48	N
C. elegans 40X	BELLA 0.15e `sensitive mode`	79.52	36.77	95.99	29.23	5,763.91	Y
	BELLA 0.13e `sensitive mode`	**80.77**	31.85	**97.19**	25.24	6,827.52	Y
	DALIGNER	52.69	**74.60**	63.66	**59.69**	*6,641.82	Y
	BLASR	2.67	22.98	3.31	18.80	114,744.57	Y
	Minimap	77.66	51.32	94.33	41.03	1,843.64	N
	Minimap2	78.80	42.26	96.60	34.12	**1,644.61**	N
	MHAP	56.62	46.82	70.05	38.07	8,371.50	N

Table 4: Filtering perfomance of BELLA.

The table shows the amount of read pairs without checking the overlapping step is able to discharge. Furthermore, it reports the number and the percentage (in brackets) of discharge read pairs after the pairwise alignment step with respect to the read squared value. For the alignment step, BELLA was run in `sensitive mode`. See Table 1 for further information on each data set.

Dataset	Read^2	Overlapping Pair	Pairwise Alignment
E.coli (Sample)	285 M	2.38 M (0.83%)	560 K (0.200%)
C.elegans 40X	177 G	1.38 G (0.78%)	39 M (0.020%)
C.elegans 20X	449 G	415 M (0.09%)	16 M (0.003%)

detection without missing large numbers of overlaps. Tables 2, 3, and 4 demonstrates the superiority of BELLA compared to state-of-the-art software for overlap detection. BELLA's sensitive mode retains the highest amount of true overlaps among the state-of-the-art software for all considered datasets. BELLA's sensitive mode provides a good trade-off between precision and runtime while prioritizing high values of recall. The runtime of BELLA's sensitive mode was, for the most part, within the average of its competitors, which is noteworthy given that BELLA also outputs alignments accurate enough to facilitate the downstream analysis and assembly tasks.

On simulated data, BELLA's precise mode combined high values of precision with equally high values of recall, at the cost of higher runtime with respect to its sensitive mode. Table 2 showed that BELLA's precise mode outperformed the main competitor Minimap2 with an average of 1.96% higher recall and 18.90% higher precision.

On real data, BELLA's sensitive mode had the highest values of recall with value of the parameter e (i.e. error rate) depending on the dataset. The error rate is one of the parameters, alongside the depth and the k-mer length, that we use to compute the reliable k-mer range. Consequently, the error rate of the dataset influences the amount of k-mers retained for overlap detection, which in turn affects recall and precision. For real *E. coli* data and for synthetic *P. aeruginosa* data, the values of recall of Minimap2 were comparable to those achieved by BELLA; for the rest of the datasets, BELLA clearly achieved the highest recall. For real data, Table 3 showed that the precision of BELLA's sensitive mode is moderate compared to Minimap2, while the precision of BELLA's precise mode $0.15e$ is on average 19.34% higher than the precision of Minimap2 on single-mapped ground truth and 7.90% when compared against multi-mapped ground truth.

The increase in precision of BELLA's precise mode is coupled with a loss in recall on real data. The error rate influences both the amount of reliable k-mers used for overlap detection and the adaptive alignment score threshold used as cutoff in the precise mode. Thus, we hypothesize that the uncertainty upon the exact error rate for a real dataset plays an important role in this recall loss. The integration within BELLA of an error rate estimator could potentially improve BELLA's performance on real data. A possible path could be to estimate the error rate of the dataset from the phred quality scores of the raw sequences, as proposed by Ono et al. [29].

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 truths 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. Likewise, BELLA's margin of improvement over the second-best tool available jumped from an average of 2.42% for single-mapped data, to an average of 3.41% for multi-mapped data. The inclusion of multiple mappings of a read minimizes the erroneous exclusion of valid read-to-reference alignments from the ground truth. Therefore, it is possible that BELLA's true precision 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; this choice was influenced by experimental results demonstrating that a higher number of seeds does not significantly improve recall. BELLA's sensitive mode achieved 98.30% recall with $e = 0.15$ for the *E. coli* (Sample) dataset (as shown in Table 3); if all the shared k-mers were used as seeds, BELLA achieved 98.51% recall. This slight increase in recall came at the cost of a $\approx 31\times$ slower runtime. The maximum recall achievable by BELLA on the considered dataset is 98.77%, which happens if the overlapping module is used alone by skipping pairwise alignment. Therefore, the loss of information using only two seeds seems negligible.

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.

5 Conclusion

Single-molecule sequencing technologies have favored overlap-based approaches (OLC) for *de novo* assembly, enabling the resolution 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 seed-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. BELLA's alignment step comprises two alternatives: the sensitive mode and the precise mode. In the precise mode, we 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 achieved higher recall than state-of-the-art tools in its sensitive mode and higher precision than state-of-the-art tools in its precise mode, 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 an error rate estimator to improve BELLA's ability to separate true alignments from false positives.

Availability Software package available at: https://github.com/giuliaguidi/bella.

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

Funding This work is supported by 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.

References

[1] Dana Angluin and Leslie G Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. *Journal of Computer and system Sciences*, 18(2):155–193, 1979.

[2] Ariful Azad, Grey Ballard, Aydin Buluc, James Demmel, Laura Grigori, Oded Schwartz, Sivan Toledo, and Samuel Williams. Exploiting multiple levels of parallelism in sparse matrix-matrix multiplication. *SIAM Journal on Scientific Computing*, 38(6):C624–C651, 2016.

[3] David R Bentley, Shankar Balasubramanian, Harold P Swerdlow, Geoffrey P Smith, John Milton, Clive G Brown, Kevin P Hall, Dirk J Evers, Colin L Barnes, Helen R Bignell, et al. Accurate whole human genome sequencing using reversible terminator chemistry. *nature*, 456(7218):53–59, 2008.

[4] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623–630, 2015.

[5] Aydin Buluç, Tim Mattson, Scott McMillan, José Moreira, and Carl Yang. Design of the graphblas API for C. In *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 643–652. IEEE, 2017.

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

[7] Herman Chernoff et al. 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, 1952.

[8] Justin Chu, Hamid Mohamadi, René L Warren, Chen Yang, and Inanc Birol. Innovations and challenges in detecting long read overlaps: an evaluation of the state-of-the-art. *Bioinformatics*, 33(8):1261–1270, 2016.

[9] Steven Dalton, Luke Olson, and Nathan Bell. Optimizing sparse matrix—matrix multiplication for the GPU. *ACM Transactions on Mathematical Software (TOMS)*, 41(4):25, 2015.

[10] Mehmet Deveci, Christian Trott, and Sivasankaran Rajamanickam. Performance-portable sparse matrix-matrix multiplication for many-core architectures. In *IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 693–702. IEEE, 2017.

[11] Andreas Döring, David Weese, Tobias Rausch, and Knut Reinert. SeqAn an efficient, generic C++ library for sequence analysis. *BMC bioinformatics*, 9(1):11, 2008.

[12] John Eid, Adrian Fehr, Jeremy Gray, Khai Luong, John Lyle, Geoff Otto, Paul Peluso, David Rank, Primo Baybayan, Brad Bettman, et al. Real-time DNA sequencing from single polymerase molecules. *Science*, 323(5910):133–138, 2009.

[13] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *AofA: Analysis of Algorithms*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

[14] Francesca Giordano, Louise Aigrain, Michael A Quail, Paul Coupland, James K Bonfield, Robert M Davies, German Tischler, David K Jackson, Thomas M Keane, Jing Li, et al. De novo yeast genome assemblies from MinION, PacBio and MiSeq platforms. *Scientific reports*, 7(1):3935, 2017.

[15] Sara Goodwin, James Gurtowski, Scott Ethe-Sayers, Panchajanya Deshpande, Michael C Schatz, and W Richard McCombie. Oxford nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome research*, 25(11):1750–1756, 2015.

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

[17] Jeremy Kepner and John Gilbert. *Graph algorithms in the language of linear algebra*. SIAM, 2011.

[18] Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736, 2017.

[19] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

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

[21] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[22] Xiaozhou Li, David G Andersen, Michael Kaminsky, and Michael J Freedman. Algorithmic improvements for fast concurrent cuckoo hashing. In *Proceedings of the Ninth European Conference on Computer Systems*, page 27. ACM, 2014.

[23] Zhenyu Li, Yanxiang Chen, Desheng Mu, Jianying Yuan, Yujian Shi, Hao Zhang, Jun Gan, Nan Li, Xuesong Hu, Binghang Liu, et al. Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph. *Briefings in functional genomics*, 11(1):25–37, 2012.

[24] Yu Lin, Jeffrey Yuan, Mikhail Kolmogorov, Max W Shen, Mark Chaisson, and Pavel A Pevzner. Assembly of long error-prone reads using de bruijn graphs. *Proceedings of the National Academy of Sciences*, 113(52):E8396–E8405, 2016.

[25] Andrey Markov. Extension of the limit theorems of probability theory to a sum of variables connected in a chain. 1971.

[26] Pall Melsted and Jonathan K Pritchard. Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC bioinformatics*, 12(1):333, 2011.

[27] Gene Myers. Efficient local alignment discovery amongst noisy long reads. In *International Workshop on Algorithms in Bioinformatics*, pages 52–67. Springer, 2014.

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

[29] Yukiteru Ono, Kiyoshi Asai, and Michiaki Hamada. Pbsim: Pacbio reads simulator—toward accurate genome assembly. *Bioinformatics*, 29(1):119–121, 2012.

[30] Adam M Phillippy, Michael C Schatz, and Mihai Pop. Genome assembly forensics: finding the elusive mis-assembly. *Genome biology*, 9(3):R55, 2008.

[31] Jared T Simpson and Richard Durbin. Efficient de novo assembly of large genomes using compressed data structures. *Genome research*, 22(3):549–556, 2012.

[32] Hajime Suzuki and Masahiro Kasahara. Introducing difference recurrence relations for faster semi-global alignment of long sequences. *BMC bioinformatics*, 19(1):45, 2018.

[33] Wenyu Zhang, Jiajia Chen, Yang Yang, Yifei Tang, Jing Shang, and Bairong Shen. A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. *PloS one*, 6(3):e17915, 2011.

[34] Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller. A greedy algorithm for aligning DNA sequences. *Journal of Computational biology*, 7(1-2):203–214, 2000.

Supplementary Material

S1 Alignment to Reference: BWA-MEM and Minimap2

Table S1 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[3] that Minimap2 is better than BWA-MEM to align sequences to reference when long-read data is used.

Table S1: 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. BELLA was run in its sensitive mode. Brackets refer to negative values.

		Minimap2		BWA-MEM		Change	
Data Set	Tool	Recall	Precision	Recall	Precision	Recall	Precision
E. coli (Sample)	BELLA 0.15*e*	98.30	61.10	98.53	55.35	(0.23)	10.39
	BELLA 0.13*e*	98.45	60.14	98.77	53.07	(0.32)	13.32
	BELLA 0.11*e*	98.56	58.60	98.67	54.47	(0.11)	7.58
	DALIGNER	92.17	63.60	92.68	57.78	(0.55)	10.07
	BLASR	79.90	80.18	81.19	73.51	(1.59)	9.07
	Minimap	91.09	78.05	91.96	71.10	(0.95)	9.77
	Minimap2	95.58	73.28	96.03	66.45	(0.47)	10.28
	MHAP	80.84	67.32	82.04	61.58	(1.46)	9.32
E. coli	BELLA 0.15*e*	91.50	67.27	92.14	61.38	(0.69)	9.60
	BELLA 0.13*e*	92.69	66.26	93.28	60.43	(0.63)	9.65
	BELLA 0.11*e*	94.81	64.49	95.30	58.75	(0.51)	9.77
	DALIGNER	83.58	61.61	83.95	56.08	(0.44)	9.86
	BLASR	36.19	84.93	37.21	78.94	(2.74)	7.58
	Minimap	87.80	78.31	88.83	71.69	(1.16)	9.23
	Minimap2	93.22	74.38	93.72	67.72	(0.53)	9.83
	MHAP	77.74	66.06	79.08	60.76	(1.68)	8.72
C. elegans 40X	BELLA 0.15*e*	95.99	29.23	93.31	24.79	2.87	17.91
	BELLA 0.13*e*	97.19	25.24	94.68	21.44	2.65	17.72
	DALIGNER	63.66	59.69	61.09	50.09	4.21	19.17
	BLASR	3.31	18.80	3.13	15.47	5.75	21.53
	Minimap	94.33	41.03	93.01	35.28	1.42	16.30
	Minimap2	96.60	34.12	93.81	28.89	2.97	18.10
	MHAP	70.05	38.07	72.22	34.20	(3.00)	11.32

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

S2 Alignment Constraint

Table S2: Alignment constraint on synthetic data.

The table shows how recall and precision change when the constraint on the end of the alignment is applied. The filtering rule is applied in its "relaxed" form only on pairs sharing one k-mer. Numbers in brackets refer to negative values.

		Default		Constrained		Change	
Tool	Data Set	Recall	Precision	Recall	Precision	Recall	Precision
BELLA 0.15*e* `sensitive mode`	*P. aeruginosa*	99.63	47.53	99.60	61.29	(0.03)	22.44
	V. vulnificus	99.44	56.61	99.41	59.46	(0.03)	5.04
	A. abaumannii	99.58	56.45	99.55	59.83	(0.03)	5.97
	C. elegans 20X	98.63	47.05	98.48	55.45	(0.15)	17.84
BELLA 0.15*e* `precise mode`	*P. aeruginosa*	99.59	89.05	99.58	89.05	(0.01)	0.00
	V. vulnificus	99.31	83.21	99.30	83.22	(0.01)	0.01
	A. abaumannii	99.47	83.92	99.47	83.92	0.00	0.00
	C. elegans 20X	97.70	86.88	97.70	86.88	0.00	0.00

In Section 2.6, we described the rationale behind forcing alignment to extend until the end of the sequences in order to consider a read pair as true overlapping. Table S2 compares the performance of the BELLA on synthetic data when this constrained is applied. It emerges that this constrained is convenient when the goal is to increase precision on sensitive mode, while it does not bring any significant benefit to the precise mode given the cutoff implementation in this setting. Similarly, Table S3 compares the performance of the BELLA on real data when applying this constraint, and it shows consistent results with Table S2.

Table S3: Alignment constraint on real data.

The table shows how recall and precision change when the constraint on the end of the alignment is applied. The filtering rule is applied in its "relaxed" form only on pairs sharing one k-mer. Numbers in brackets refer to negative values.

		Default		Constrained		Change	
Tool	Data Set	Recall	Precision	Recall	Precision	Recall	Precision
BELLA 0.15*e* `sensitive mode`	*E. coli* (Sample)	98.30	61.10	98.08	63.57	(0.22)	4.05
	E. coli	92.14	61.38	90.53	63.65	(1.75)	3.71
	C. elegans 40X	95.99	29.23	95.15	37.99	(0.87)	29.96
BELLA 0.15*e* `precise mode`	*E. coli* (Sample)	71.71	88.23	71.69	88.24	(0.02)	0.01
	E. coli	64.93	87.98	64.92	87.98	(0.03)	0.00

S3 Ground Truth Generation for a Real Dataset

As with the original procedure presented by Li [20], reads were aligned against the reference genome using BWA-MEM `0.7.17-r1188` [19] 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 [21] 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.