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

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

[1]Department of Electrical Engineering and Computer Sciences, University of California at Berkeley
[2]Department of Molecular and Cell Biology, University of California at Berkeley
[3]Computational Research Division, Lawrence Berkeley National Laboratory
[4]DOE Joint Genome Institute

Abstract

Recent advances in long-read sequencing allow characterization of genome structure and its variation within and between species at a resolution not previously possible. Detection of overlap between reads is an essential component of many long read genome pipelines, such as *de novo* genome assembly. Longer reads simplify genome assembly and improve reconstruction contiguity, but current long read technologies are associated with moderate to high error rates.

In this work, we present Berkeley Efficient Long-Read to Long-Read Aligner and Overlapper (BELLA), a novel overlap detection and alignment algorithm using sparse matrix-matrix multiplication. In addition, we present a probabilistic model that demonstrates the feasibility of using k-mers for overlap candidate detection and shows its flexibility when applied to different k-mer selection strategies. Based on such a model, we introduce a notion of *reliable k-mers*. Combining reliable k-mers with our *binning* mechanism increases the computational efficiency and accuracy of our algorithm. Finally, we present a new method based on Chernoff bounds to separate true overlaps from false positives by combining alignment techniques and probabilistic modeling. Our goal is to maximize the balance of precision and recall.

For both real and synthetic data, BELLA is among the best F1 scores, showing a stability of performance that is often lacking in competing software. BELLA's F1 score is consistently within 1.7% of the top performer. In particular, we show improved *de novo* assembly quality on synthetic data when BELLA is coupled with the miniasm assembler.

1 Introduction

Recent advances in long-read sequencing technologies have enabled characterization of genome structure and its variation between and within species that was not previously possible. However, post-sequencing data analysis remains a challenging task. One of the most challenging aspects of analyzing DNA fragments from high-throughput sequencing, namely *reads*, is whole-genome assembly [32], i.e., the process of aligning and assembling DNA fragments to reconstruct the original sequence. More specifically, *de novo* genome assembly reconstructs a genome from redundantly sampled reads without prior knowledge of the genome, enabling the study of previously uncharacterized genomes [30].

Long-read technologies [11, 14] generate long reads with an average length often exceeding 10,000 base pairs (bp). These allow resolution of complex genomic repeats and enable more accurate ensemble views that were not possible with previous short-read technologies [28, 25]. However, the improved read length of these technologies is accompanied by lower accuracy, with error rates from 0.5% to 15%. Nevertheless, the error distribution for Pacific Biosciences long reads [13] is more random and uniform compared to short-read technologies.

A long read assembler typically uses the Overlap-Layout-Consensus (OLC) assembly paradigm [2]. The first step in OLC assembly is to detect overlaps between reads to create an overlap (or string) graph. The OLC paradigm benefits from longer reads, as significantly fewer reads are needed to cover the genome, limiting the size of the overlap graph. Highly accurate overlap detection is a major computational bottleneck in OLC assembly [24], mainly due to the computationally intensive nature of pairwise alignment.

Currently, 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 (i.e., substrings of fixed length k) or suffix array, to identify a set of initial candidate read pairs, thereby reducing the high cost of computing pairwise alignments in a second step [8].

The process of identifying a set of initial candidate read pairs, sometimes referred to simply as "overlap", affects both the accuracy and runtime of the algorithm. Accurate identification of initial candidate read pairs minimizes the runtime of pairwise alignment, while retaining all pairs that truly overlap in the genome. Computationally efficient and accurate overlapping and alignment algorithms have the potential to improve existing long-read assemblers, enabling *de novo* reference assemblies, detection of genetic variations of higher quality, and accurate metagenome classification.

Our main contributions are:

1. Using a Markov chain model [22], we show that a

k-mer seed-based approach is useful for accurately identifying overlap candidates. Our model is general enough to be applicable to different types of sequencing data and k-mer selection strategies.

2. To increase computational efficiency without loss of accuracy, we develop a simple procedure for pruning k-mers and prove that it preserves almost all true overlaps with high probability.
3. To take advantage of high performance techniques, we reformulate the problem of overlap detection in terms of sparse matrix-matrix multiplication (SpGEMM), which has not been previously applied in the context of long-read overlap and alignment. This approach is flexible and can be implemented independently of the k-mer selection strategy.
4. By coupling our overlap detection with our newly developed seed-and-extend alignment algorithm, we present a novel method to separate true alignments from false positives.

2 Related Work

The increasing popularity of long read sequencing data has led to many efforts in the literature to perform accurate overlap detection. In the context of long read overlap detection, we can distinguish between "base-level alignment" and "overlap-only" strategies below.

Base-Level Alignment: DALIGNER [24] uses k-mers to find overlap candidates and then perform alignment. It parses the sequences in k-mers, sorts them, and finds overlapping sequences with a merge operation. To filter out spurious overlap candidates, a pairwise alignment is performed using a linear expected-time heuristic based on the difference algorithm [23]. BLASR [6], originally developed to align noisy long-read sequencing data to reference genomes, later became popular as a read-to-read aligner. It too first uses k-mers to detect initial overlap candidates and then filters them using alignment. In addition to the aligner itself, Chaisson and Tesler [6] presented a mathematical model that proved the feasibility of using a k-mer seed to find a match between a noisy long-read sequence and a correct reference sequence. In this paper, we make a similar contribution by presenting a different model that proves the feasibility of using a k-mer seed to find a match between two noisy long-read sequences, and that is not restricted to regular k-mer selection strategies.

Overlap Only: Li's minimap2 [19] also uses seeds to find matches. However, it uses a different kind of k-mer, called a minimizer, which reduces the number of seeds because it selects only one minimizer in a window w whose value is the minimum according to a function. It does not perform any alignment. Instead, it computes an approximate alignment score based on the location of the minimizers on the sequences and excludes those whose quality is below a defined threshold. MECAT [31] identifies overlap candidates using k-mers and introduces a pseudo-linear alignment scoring algorithm to filter excessive candidates using a distance difference factor to score k-mer matches. MHAP [2] is a probabilistic algorithm for sequence overlap detection. It estimates Jaccard similarity by compressing sequences to their representative identity composed of min-mers, filtering out false candidates.

SpGEMM is a relatively unknown primitive in genomics. Most notably, Besta et al. [3] used SpGEMM to compute similarity between genomes in distributed memory, after the appearance of our preprint [15].

3 Proposed Algorithm

In this paper, a computationally efficient and accurate overlap detection and alignment algorithm for long read genomic pipelines is developed and implemented in a software package called BELLA.

BELLA uses a seed-based approach to detect overlaps in the context of long read applications. Such an approach parses reads into k-mers (i.e., substrings of fixed length k), which are then used as feature vectors to identify overlaps between all reads. Using a Markov chain model, we demonstrate the feasibility of a k-mer seed-based approach for long read overlap detection. The descriptiveness of our model allows us to model the probability of finding a correct common seed between two sequences even when k-mer strategies other than ours are used, such as minimizers [19] and syncmers [10].

Importantly, not all k-mers are equal in terms of their usefulness for overlap detection. For example, the vast majority of k-mers that occur only once in the dataset are errors (and are also not useful for detecting overlaps between read pairs). Similarly, k-mers that occur more frequently than one would expect given the sequencing depth and error rate are likely from repetitive regions. It is a common practice to prune the k-mer space using various methods [18, 21, 5].

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 retained by BELLA are considered *reliable*, where the reliability of a k-mer is defined as its probability of originating from a unique (non-repetitive) region of the genome. Our reliable k-mer detection maximizes retention of k-mers from unique regions of the genome using probabilistic analysis given error rate and sequencing depth.

BELLA uses a sparse matrix to represent its data internally, where the rows are reads, the columns are reliable k-mers, and a nonzero $\mathbf{A}(i, j) \neq 0$ is the position of the j-th k-mer within the i-th read. The construction

of this sparse matrix requires efficient k-mer counting.

Overlap detection is implemented in BELLA using SpGEMM, which allows our algorithm to achieve fast overlap without using approximate approaches. SpGEMM is a highly flexible and efficient paradigm that allows for better organization of computation and generality, as it can manipulate complex data structures, such as those used to perform overlap detection using common k-mers. Our k-mer selection can be easily replaced by other selection strategies without compromising the SpGEMM-based overlap detection, demonstrating the generality of our approach. Implementing this method in our pipeline enables the use of high-performance techniques that have not been previously applied in the context of long read alignment. It also enables continuous performance improvements in this step due to increasingly optimized implementations of SpGEMM [26, 9].

BELLA's overlap detection has been coupled with our high-performance seed-and-extend algorithm, which means that the alignment between two reads starts from a common seed (identified in the previous overlap detection) and not necessarily from the beginning of the reads. To refine the seed selection, we introduce a procedure called *binning*. The k-mer positions in a read pair are used to estimate the length of the overlap, and the k-mers are "binned" based on their length estimates. We consider for the alignment only k-mers that belong to the most crowded bins, which we call *consensus* k-mers. During the alignment phase, BELLA uses a new method to separate true alignments from false positives as a function of the alignment score. We prove that the probability of false positives decreases exponentially as the length of overlap between reads increases.

Existing tools also implement approximate overlap detection using sketches. A sketch is a space-reduced representation of a sequence. Several randomized hash functions convert k-mers into integer fingerprints, and a subset of them is selected to represent the sketch of a sequence according to some criterion. For example, Berlin et al. [2] keep only the smallest integer for each hash function and use the collection of these minimal-valued fingerprints as the sketch. These methods are fast, but only approximate, since the sketch is a lossy transformation. In contrast, BELLA uses an explicit k-mer representation that allows us to couple our overlap detection with a seed-and-extend alignment to refine the output and improve the precision of our algorithm.

4 Methods

4.1 Overlap Feasibility Chaisson and Tesler [6] proposed a theory of how long reads contain subsequences that can 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 [13]. The result obtained from their theory is the minimum sequence length to have an *anchor* within a confidence interval.

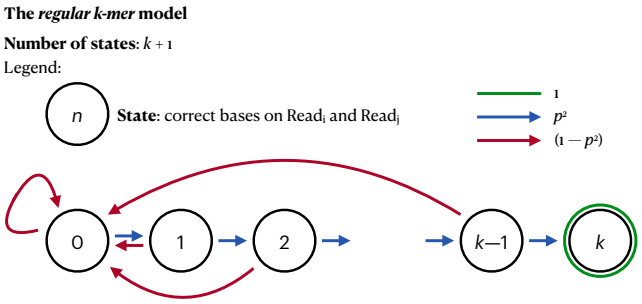

Figure 1: Proposed Markov Chain model demonstrating the feasibility of using k-mers for overlap detection.

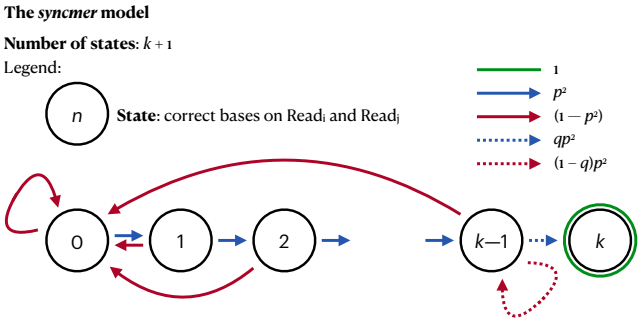

Figure 2: Proposed Markov Chain model using syncmer approach instead of the k-mer one, where $q = 1/c$.

Here we present an alternative model of how these subsequences, also called k-mers, can be used to anchor alignments between two erroneous long read sequences, allowing accurate overlap detection between all 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 [22].

The Markov chain process is characterized by a *transition matrix* $\mathbf{P}$ that contains the probabilities of transitioning from one state to another. Each row index *start* of $\mathbf{P}$ represents the initial state, and each column index *end* of $\mathbf{P}$ represents the final state. Each entry of $\mathbf{P}$ is a non-negative number indicating a *transition probability*. Our transition matrix has $(k+1)$ possible states, resulting in $(k+1)^2$ transition probabilities for the transition from *start* to *end*. The probability of having a correct base in both reads is p^2. For every state except the *absorbing* state k, an error in at least

Algorithm 1 Probability of observing at least one 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
```

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 since both $read_1$ and $read_2$ have already seen k successive correct bases. Therefore, its transition probability is 1. Figure 1 describes the process: each state contains the number of successfully sequenced bases obtained on both reads up to that point, while the arrows represent the transition probabilities.

One can then find the probability of being in one of the states after L steps in the Markov chain by computing the L-th power of the matrix $\mathbf{P}$, where L is the length of overlap between the two sequences. More efficiently, this can be computed iteratively with only L sparse matrix vector products, starting from the unit vector $\mathbf{v} \leftarrow (1, 0, \ldots, 0)$ (Algorithm 1). This approach is sufficient since we are only interested in the probability of being in the absorbing final state. This operation leads to the probability of having a correct k-mer at the same location on both reads, given a certain overlap region. This model is the driving factor for choosing the optimal k-mer length used in overlap detection.

Our Markov chain can be modified to accommodate different k-mer selection strategies, such as syncmers and minimizers. Given a compression factor $c > 1$ that sets a minimal value for the k-mer code, a k-mer κ is a *mincode syncmer* if $code(\kappa) \leq H/c$, where H is the maximal possible code [10]. The probability that a given k-mer is chosen as a mincode syncmer is $1/c$. In this case, we can modify our model to include the probability that a k-mer is correct *and* that it is retained as a mincode syncmer. The transition from the $(k-1)$-th state to the k-th state in the Markov chain will model syncmer selection so that we have a probability of transitioning from $k-1$ to k that is equal to qp^2, where $q = 1/c$, i.e., the k-th is correctly sequenced and the k-mer is a mincode syncmer. Then, we have a probability of $(1-q)p^2$ to stay in the $(k-1)$-th state, which means that we have sequenced a correct base on both sequences, but the k-mer is not selected as a syncmer. The probability of returning to the initial state is unchanged. The Markov chain model that is modified for mincode syncmers is shown in Figure 2.

We used the mincode syncmer as an example but the same probabilistic model applies to other syncmer types, including those with better spacing properties, such as closed syncmers [10]. This is because the selection of a k-mer as a syncmer is by definition a local decision and is not affected by neighboring k-mers.

The case of the minimizer is slightly different. A k-mer is a minimizer if it has the smallest code among w consecutive k-mers, where w is called the window length [29]. For a k-mer κ that is correctly sequenced in both reads, we need to consider the number of competing k-mers in its windows to determine the probability that κ is selected as the minimizer *from both reads*. If there are no sequencing errors, there are w competing k-mers including κ itself. Because errors can change the competing k-mers in each read independently, the maximum number of competing k-mers including κ itself is $2w-1$. It is possible to compute the exact expected number of competing k-mers, but since this range is narrow within a factor of two, we can also use the upper and lower bounds instead when choosing minimizer parameters (w, k) in practice.

4.2 Reliable k-mers Repetitive regions of the genome cause certain k-mers to occur frequently in input reads. K-mers from these regions pose two problems for pairwise overlap and alignment. First, their presence increases the computational cost, both in the overlap and alignment phases, as these k-mers generate numerous and possibly incorrect overlaps. Second, they often do not provide valuable information.

Our argument here is that k-mers that originate from a repetitive region in the genome can be ignored for seed-based overlap. This is because either (a) the read is longer than the repeat, in which case there should be enough sequence data from the non-repetitive section to find overlaps, or (b) the read is shorter than the repeat, in which case their overlaps are inherently ambiguous and uninformative and will not be particularly useful for downstream tasks such as *de novo* assembly. In the case of a nearly identical region, we would expect to find

a k-mer that comes from a unique region of that nearly identical repetition to identify that region.

Following the terminology proposed by Lin et al. [21], we refer to k-mers not present in the genome as *non-genomic* and thus characterize k-mers present in the genome as *genomic*. A genomic k-mer may be *repeated* if it occurs multiple times in the genome, or *unique* if it does not. One can think of the presence of k-mers in each read as the feature vector of that read. Therefore, the feature vector should contain all unique k-mers, as they are often the most informative features.

Since we do not know the genome before assembly, we estimate the genomic uniqueness of k-mers from redundant, error-prone reads. Here we present a mathematically based procedure that selects a frequency range for k-mers that we consider *reliable*. The basic question guiding the procedure for selecting reliable k-mers is the following: "Given that a k-mer is sequenced from a unique (non-repeated) region of the genome, what is the probability that it occurs at least m times in the input data?" For a genome G sequenced at depth d, the conditional modeled probability is:

$$Pr(\text{FREQ}(k\text{-}mer, G, d) \geq m | \text{COUNT}(\text{MAP}(k\text{-}mer, G) = 1) \tag{4.1}$$

where $\text{MAP}(k\text{-}mer, G)$ is the set of sites in genome G to which k-*mer* can be mapped, the function $\text{COUNT}()$ computes the cardinality of a given input set, and $\text{FREQ}(k\text{-}mer, G, d)$ is the expected number of occurrences of k-*mer* within sequenced reads, assuming that each region of G is sequenced d times (*sequencing depth*). In this sense, BELLA's approach to selecting reliable k-mers is very different from the way Lin et al. [21] select their *solid strings*. While solid strings discard rare k-mers, our model discards highly recurrent k-mers because (a) unique k-mers are sufficient to find informative overlaps, and (b) a unique k-mer has a low probability of occurring frequently.

The probability of correctly sequencing a k-mer is approximately $(1-e)^k$, where e is the error rate. The probability of correctly sequencing a k-mer once can be generalized to the probability of seeing it multiple times in the data, provided 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 approximately $(1-e)^{dk}$. More generally, the number of correct sequencing of a unique k-long genome segment at a sequencing depth d follows a binomial distribution:

$$B(n = d, p = (1-e)^k) \tag{4.2}$$

where n is the number of trials and p is the probability of success. From this, we derive that the probability of observing a k-mer k_i (corresponding to a unique non-repetitive region of the genome) m times within a sequencing input data with depth d is:

$$Pr(m; d, (1-e)^k) = \binom{d}{m}(1-e)^{km}(1-(1-e)^k)^{(d-m)} \tag{4.3}$$

where m is the multiplicity of the k-mer in the input data, e is the error rate, d is the sequencing 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 calculated.

Equation 4.3 is used to identify the range of reliable k-mers. To choose the lower bound l, we compute $Pr(m; d, (1-e)^k)$ for each multiplicity m and sum these probabilities cumulatively, starting from $m = 2$. The cumulative sum does not start at $m = 1$ because a k-mer that occurs only once in the input data (and therefore appears on a single read) cannot be used to identify the overlap between two reads. The lower bound l is the smaller m value after which the cumulative sum exceeds a user-defined threshold ϵ. The choice of l is significant 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 these cases, a k-mer with small multiplicity has a high probability of being incorrect.

The upper bound u is chosen in a similar way. Here, starting from the largest possible value of m (i.e. d), the probabilities are added up cumulatively. In this case, u is the largest value of m after which the cumulative sum exceeds the user-defined threshold ϵ. The k-mers that are more frequent than u have too low a probability of belonging to a unique region of the genome, and multiple mapped k-mers would lead to an increase in computational cost and possibly misassembly.

K-mers with multiplicity greater than u and multiplicity less than l in the input set are discarded and not used as read features in the downstream algorithm. Our reliable k-mers selection discards at most 2ϵ useful information when the data fits the model, in the form of k-mers that can be used for overlap detection.

When using syncmers instead of k-mers, the reliable range calculations are unchanged. This is because any given k-mer is either selected as a syncmer for all its occurrences in the dataset, or it is never selected as a syncmer. For minimizers, the exact computation of the reliable range is non-trivial, since errors in flanking sequences affect whether a k-mer is retained as a minimizer. Therefore, we leave this as future work.

4.3 Sparse Matrix Construction and Multiplication BELLA uses a sparse matrix format to store its data and a sparse matrix multiplication (SpGEMM) to

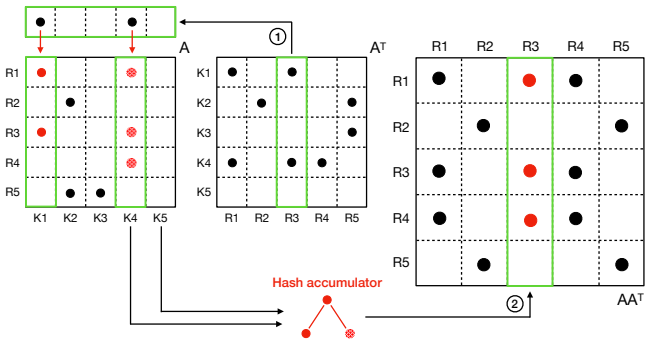

Figure 3: Column-wise sparse matrix multiplication. $\mathbf{A}^\mathsf{T}(:, R_3)$ is chosen as the "active column": its non-zero elements define which columns of $\mathbf{A}$ to consider by looking at their corresponding row indices in $\mathbf{A}^\mathsf{T}$.

identify overlaps. Sparse matrices express data access patterns concisely and clearly, and allow for 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 rows and the entries of the k-mer dictionary as columns. If the j-th reliable k-mer is present in the i-th read, the (i, j) cell of $\mathbf{A}$ is nonzero. $\mathbf{A}$ 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 cell (i, j) of the overlap matrix that is non-zero contains the number of shared k-mers between the i-th read and the j-th read, and the corresponding positions in the corresponding read pair of (at most) two shared k-mers.

Column-wise sparse matrix multiplication is efficiently implemented 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 apply to the overlap problem and allow for further performance improvements, as several software packages already implement these primitives, including Intel MKL and Sandia's KokkosKernels [9].

The SpGEMM algorithm shown in Figure 3 is functionally equivalent to a k-mer-based seed index table common in other long read alignment codes. However, unlike hash tables, the CSC format allows true, constant-time random access to the columns. More importantly, the computational problem of accumulating the contributions of multiple shared k-mers to each read pair is automatically handled by choosing appropriate data structures within SpGEMM. Figure 3 illustrates the merging operation of BELLA, which uses a hash table data structure indexed by the row indices of $\mathbf{A}$, following the multi-threaded implementation proposed by Nagasaka et al. [26]. This hash table based accumulator is merely one way to compute SpGEMM and many other methods have been proposed in the literature [12]. Finally, the content of the hash table is stored in a column of the final matrix once all the required nonzeros for that column are accumulated.

Since BELLA performs a seed-and-extend alignment from pairs of reads that share at least t (by default $t = 1$) k-mers, the overlap phase must keep track of the positions of the shared k-mers. In cases where multiple k-mers are shared between any pair of reads, it is not economical to store all k-mer matches. Even though the previously described reliable k-mer selection procedure eliminates most matches from repeated regions, it is still possible to observe misleading k-mer matches due to sequencing errors and repetitions.

To ensure optimal seed selection for the alignment step, BELLA applies the following binning method. Whenever a common k-mer is found between a pair of reads, we use its position in these reads to estimate an overlap length and orientation. A new overlap estimate forms its bin with a single element unless it is already within the boundaries of a previous bin (i.e., within an adjustable distance β, which is 500 by default). In this case, the vote count of the adjacent bin is increased by one, as long as the two originating k-mers do not overlap. Only the two bins with the highest vote count are retained, along with a representative k-mer that supports this overlap estimate.

In the resulting sparse overlap matrix $\mathbf{A}\mathbf{A}^\mathsf{T}$, each non-zero cell (i, j) is a structure consisting of an integer value storing the number of shared k-mers and an integer array of size 4 storing the position of (up to two) shared k-mers on read i and read j corresponding to the overlap estimate bins with the highest number of votes. To enable this special multiplication, which performs scalar multiplication and addition differently, we use semiring abstraction [17]. Multiplication on a semiring allows the user to overload scalar multiplication and addition operations while still using the same SpGEMM algorithm. Many existing SpGEMM implementations support custom semirings, including those that implement the GraphBLAS API [4].

As genome size increases, so does the memory required to construct the final overlap matrix. For large genomes, the sparse overlap matrix $\mathbf{A}\mathbf{A}^\mathsf{T}$ may not fit in memory even if the data matrix $\mathbf{A}$ does. BELLA avoids this situation by splitting the multiplication into batches based on the available RAM. In each phase, only one batch of columns of the overlap matrix is created. The set of nonzeros in this batch of the overlap matrix is immediately tested for alignments (as described in Section 4.4). The pairs that pass the alignment test are written to the output file of BELLA, so that the current batch of the overlap matrix can be discarded.

Due to the nature of our problem, the sparse overlap matrix $\mathbf{AA}^\mathsf{T}$ is a symmetric matrix. Therefore, we compute the multiplication using only the lower triangle of $\mathbf{A}$ and avoid 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 halved the memory requirement. The obvious solution of computing inner projections of rows of $\mathbf{A}$ is suboptimal, since it must perform $\Omega(|reads|^2)$ inner products even though the majority of inner products are zero. In contrast, our column-wise implementation runs faster than $O(|reads|^2)$ when the overlap matrix $\mathbf{AA}^\mathsf{T}$ is sparse. Since the main purpose of the overlap process is to filter overlap candidates, the overlap matrix tends to be sparse over 99%.

4.4 Pairwise Alignment High precision is desirable to avoid unnecessary work in the subsequent steps of *de novo* assembly. Therefore, BELLA filters overlap candidates by performing fast, near linear-time pairwise seed-and-extend alignments.

Unlike approaches that rely on sketches or minimizers, such as Minimap2 [19] and MHAP [2], seed-and-extend alignment can be performed directly using the k-mers from the overlap phase of BELLA. BELLA's alignment module is based on our high-performance seed-and-extend banded alignment, which uses a narrow adaptive band that significantly improves performance and reduces the search space for optimal alignment.

Our binning mechanism chooses at most two seed k-mers to be used as input for the seed-and-extend x-drop alignment. For each read pair in the overlap matrix, the alignment of one or two seeds is extended until the alignment score falls x points below the previous best score. If the final score is less than a threshold n, the sequence pair is discarded.

To filter out spurious candidates, we use an *adaptive threshold*, calculated based on the estimated overlap between a given pair of reads. The choice of scoring matrix used in the pairwise alignment step can justify that the alignment score threshold is a linear function of the estimated overlap length.

Given an estimated overlap region of length L and probability p^2 of obtaining a correct base on both sequences, we would expect $m = p^2 \cdot L$ correct matches within this overlap region. The alignment score χ is:

$$\chi = \alpha m - \beta(L - m) = \alpha p^2 L - \beta(L - p^2 L) \tag{4.4}$$

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

$$\varphi = \frac{\chi}{L} = \alpha p^2 - \beta(1 - p^2). \tag{4.5}$$

The expected value of φ is equal to $2 \cdot p^2 - 1$ when an exact alignment algorithm is used. We want to define a cutoff with respect to $(1 - \delta)\varphi$ such that we keep pairs above this cutoff as true alignments and discard the remaining pairs. To this scope, we use a Chernoff bound [7, 16] to define the value of δ, and prove that there is only a small probability of missing a true overlap of length $L \geq 2000$ bp (which is typically the minimum overlap length for a sequence to be considered a true positive) using the cutoff defined above.

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 on the probability of Z deviating by a certain amount δ from its expected value. In particular, we use a corollary of the multiplicative Chernoff bound [1] defined for $0 \leq \delta \leq 1$ as:

$$\mathbf{Pr}[Z \leq (1 - \delta)\mu_z] \leq e^{\frac{-\delta^2 \mu_z}{2}} \tag{4.6}$$

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

$$X_i = \begin{cases} \alpha, & \text{with probability } p^2 \\ -\beta, & \text{with probability } 1 - p^2 \end{cases} \tag{4.7}$$

where $\alpha, \beta > 0$ are still the values associated with a match and a mismatch and a gap/indel in the scoring matrix, respectively; its expected value $E[X_i]$ is exactly equal to $\varphi = \alpha p^2 - \beta(1 - p^2)$. 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 $Z_i \in \{0, 1\}$. Thus, we define a new random variable $Y_i = X_i + \beta$ as a linear transformation of X_i that can take 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:

$$Z_i = \frac{X_i + \beta}{\alpha + \beta}, \text{ where } Z_i \in \{0, 1\} \tag{4.8}$$

From the linearity of expectation, we obtain:

$$E[Z] = E\left[\frac{X + \beta}{\alpha + \beta}\right] = \frac{E[X] + \beta L}{\alpha + \beta} = \frac{(2p^2 - 1)L + \beta L}{\alpha + \beta} \tag{4.9}$$

Table 1: Datasets used for evaluation. Datasets above the line are real data, while datasets below the line were generated using PBSIM [27]. Download: `portal.nersc.gov/project/m1982/bella/`.

Short Name	Depth	Species and Strain	Fastq Size
E.coli (Sample)	30X	*Escherichia coli* MG1655	266 MB
E.coli	100X	*Escherichia coli* MG1655	929 MB
E. coli (CCS Sample)	29X	*Escherichia coli* MG1655	240 MB
E. coli (CCS)	290X	*Escherichia coli* MG1655	2.60 GB
C.elegans	40X	*Caenorhabditis elegans* Bristol	8.90 GB
P. aeruginosa	30X	*Pseudomonas aeruginosa* PAO1	359 MB
V. vulnificus	30X	*Vibrio vulnificus* YJ016	288 MB
A. baumannii	30X	*Acinetobacter baumannii*	248 MB
C.elegans	20X	*Caenorhabditis elegans*	3.75 GB

Finally, substituting Eq. 4.8 and Eq. 4.9 into Eq. 4.6 and simplifying with our scoring matrix $\alpha, \beta = 1$, we get the final expression:

$$\text{(4.10)}\quad \mathbf{Pr}[X \leq (1-\delta)\mu_x] \leq e^{-\delta^2 p^2 L}, \text{ with } E[X] = \mu_x$$

For two sequences that correctly overlap by $L = 2000$, the probability that their alignment score is more than 10% ($\delta = 0.1$) below the mean is $\leq 5.30 \times 10^{-7}$. BELLA, with an x-drop value of $x = 50$ and an adaptive threshold derived from the scoring matrix, and the cutoff rate set to $\delta = 0.1$, achieves high values for recall and precision among state-of-the-art software tools.

5 Experimental Setting

The datasets used for evaluation are shown in Table 1. The selected genomes vary in size and complexity, as runtime and accuracy depend on these features [20]. In addition, we include a dataset based on PacBio CCS technology that was sampled at varying depths. This technology is more accurate than PacBio CLR, but has higher cost and shorter read length (although it is still classified as a long read technology).

Recall, precision, F1 score, and runtime are used as performance metrics. Recall is defined as the fraction of true-positives from the aligner/overlapper to the total ground truth. Precision is the fraction of true positives from the aligner/overlapper to the total number of elements found by the aligner/overlapper. F1 score is the harmonic mean of precision and recall.

A read pair is considered true-positive if the sequences align for at least 2 kb in the reference genome. The threshold $t = 2$ kb is derived from the procedure proposed by Heng Li [19] and the ground truth is generated using Minimap2. A description of our evaluation procedure and ground truth generation can be found in the supplementary material of our preprint [15].

In addition, we report preliminary assembly results on simulated datasets obtained by coupling the overlappers/aligners with the Miniasm assembler [19]. As metrics for assembler quality, we use the number of contigs, the number of misassemblies, N50, and the total assembly length. A contig is defined as a set of overlapping reads that together represent a consensus region of the genome. A misassembly is a position in a contig whose flanking sequences are more than 1kbps apart. N50 is a measure of assembly contiguity and is defined as the minimum contig length to cover 50% of the genome.

Results were collected on a dual-socket computer with two 20-core Intel Xeon Gold 6148 CPU ("Skylake") processors, each running at 2.4 GHz with 384 GB DDR4 2400 MHz memory, using 2 threads per core (80 threads total). To run MHAP `v2.1.3` on *C. elegans* 40X and *C. elegans* 20X, we increased the Java heap space to our largest machine, as the default setting of 32 GB resulted in out-of-memory failures. DALIGNER results are reported only for real PacBio CLR data because it requires single-cell PacBio sequencing data. For all simulated datasets and real Pacbio CCS datasets, DALIGNER quickly failed and produced the associated error "`Pacbio header line format error`". Our attempt to reformat the data also failed.

6 Results

BELLA is evaluated against several state-of-the-art long read overlap detection and alignment software (see Section 2), using both synthetic and real PacBio data. The synthetic data were generated using PBSIM [27] with an error rate of 15%. The advantage of the synthetic data is that the ground truth is known. Table 2 and Table 3 show the results on synthetic and real data, respectively, in terms of accuracy and runtime. The last column of each table indicates whether the respective overlapper also performs nucleotide-level alignment. The full runtime breakdown for BELLA is shown in Figure 7 in the supplementary material of our preprint [15].

Table 4 shows the assembly results on simulated data. Each overlapper was coupled with the Miniasm assembler [19]. If the tool also computes alignment at the nucleotide level by default, as BELLA does, the post-alignment data is used as input to Miniasm.

Table 2 shows that MECAT trades recall for precision and achieves the highest precision, but misses a large number of the true overlaps. In contrast, BELLA, Minimap2, and BLASR were consistently strong in both precision and recall (generally over 80%), but BLASR had a much higher computational cost (on average 2.6× slower than BELLA). The F1 score of BELLA is consistently higher than that of competing software, with the exception of Minimap2, which had a slight improvement of 1.1% in three of four data sets, while BELLA had an improvement of 1.2% over Miniamp2 in *C. elegans* 20X.

Table 2: Recall, precision, F1 score, and time comparison (**synthetic data**). The last column indicates whether the tool computes alignments. **Bold font** indicates best performance, underlined font indicates second best. DALIGNER was not run on synthetic datasets.

Dataset	Overlapper	Recall	Precision	F1 Score	Time (s)	Alignment
P. aeruginosa 30X	BELLA	97.66	89.68	93.50	140.43	Y
	BLASR	86.86	90.54	88.66	230.97	Y
	MECAT	38.40	**95.20**	54.72	21.76	N
	Minimap2	**99.10**	88.83	**93.69**	**17.76**	N
	MHAP	72.68	63.42	67.74	72.72	N
V. vulnificus 30X	BELLA	**97.27**	84.05	90.18	42.76	Y
	BLASR	87.31	84.74	86.01	179.51	Y
	MECAT	43.53	88.89	58.44	17.20	N
	Minimap2	96.71	**89.33**	**92.87**	**12.93**	N
	MHAP	74.52	45.10	56.19	70.21	N
A. baumannii 30X	BELLA	**97.54**	84.90	90.78	44.69	Y
	BLASR	89.51	84.58	86.98	152.76	Y
	MECAT	46.31	**90.39**	61.25	15.65	N
	Minimap2	96.89	85.79	**91.01**	**10.06**	N
	MHAP	76.88	28.79	41.89	69.02	N
C. elegans 20X	BELLA	91.80	88.02	**89.87**	2,352.24	Y
	BLASR	95.61	78.19	86.02	3,655.12	Y
	MECAT	13.45	**95.09**	23.57	222.10	N
	Minimap2	**95.76**	82.84	88.83	**194.77**	N
	MHAP	82.57	6.41	11.90	5,353.30	N

Minimap2 was the fastest tool for synthetic data and performed only overlap but no alignment.

Table 3 shows that while BLASR performed reasonably well on synthetic data, it had a lower hit rate than other software on real data. BLASR did not run on *C. elegans* 40X because its latest version (`v5.1`) does not accept `fastq` larger than 4 GB[1]). DALIGNER proved to be the fastest of the tools on *E. coli* 30X and *E. coli* 100X, but its performance drops dramatically when moving to a larger dataset, where DALIGNER has the worst runtime, 2.5× slower than BELLA, which also performs alignment. BELLA's F1 score outperformed competing software except Minimap2 on *E. coli* 100X. It is noteworthy that the precision and F1 score of BELLA is significantly better for PacBio CCS data, the sequencing data with lower error rates and shorter read length, than for competing software. For the CCS data, the δ parameter of BELLA was increased from 0.1 to 0.7.

In Table 4, we show BELLA improved assembly quality compared to competing software. MECAT produced fewer contigs than BELLA on *C. elegans* 20X, but its N50 and assembly length are significantly smaller, implying that MECAT did not retain enough true overlaps for assembly. Miniasm produced no assembly when coupled with MHAP and BLASR.

7 Discussion

BELLA proposes a computationally efficient and accurate approach to overlap and alignment of noisy long reads based on mathematical models that minimize the cost of overlap detection while maximizing the retention of true overlaps. Tables 2 and 3 show the competitive accuracy of BELLA compared to the literature and demonstrate the effectiveness of the methods we introduced and implemented. BELLA's runtime is within the average of competing software, which is remarkable considering that we perform nucleotide-level alignment that is sufficiently accurate to facilitate downstream analysis.

Table 3: Recall, precision, F1 score, and time comparison (**real data**). BLASR result for *C. elegans* 40X is not reported as BLASR `v5.1` does not accept `fastq` larger than 4 GB.

Dataset	Overlapper	Recall	Precision	F1 Score	Time (s)	Alignment
E. coli (Sample)	BELLA	82.66	**85.69**	**84.15**	60.94	Y
	DALIGNER	89.97	62.62	73.84	**8.70**	Y
	BLASR	77.64	79.64	78.63	160.05	Y
	MECAT	78.41	78.71	78.56	24.45	N
	Minimap2	**91.40**	76.36	83.21	16.57	N
	MHAP	79.71	66.93	72.76	43.67	N
E. coli	BELLA	65.08	71.22	68.01	374.37	Y
	DALIGNER	82.18	54.50	65.54	**58.50**	Y
	BLASR	35.41	72.01	47.48	715.19	Y
	MECAT	54.61	**72.69**	62.37	84.21	N
	Minimap2	**80.68**	62.30	**70.30**	107.76	N
	MHAP	67.84	44.60	53.81	287.66	N
E. coli (CCS Sample)	BELLA	96.32	**99.84**	**98.05**	66.49	Y
	BLASR	92.38	97.30	94.77	491.49	Y
	MECAT	98.21	88.39	93.04	66.12	N
	Minimap2	98.90	58.34	73.39	51.78	N
	MHAP	**99.05**	38.29	55.23	**50.98**	N
E. coli (CCS)	BELLA	97.67	99.81	**98.73**	8,444.64	Y
	BLASR	9.11	**100.00**	16.70	11,058.86	Y
	MECAT	15.71	99.95	27.15	**289.38**	N
	Minimap2	98.83	69.94	81.91	5,828.72	N
	MHAP	**98.99**	38.80	55.75	2,277.66	N
C. elegans 40X	BELLA	75.43	73.81	**74.61**	9,042.36	Y
	DALIGNER	62.81	58.66	60.67	22,797.50	Y
	MECAT	73.05	**75.27**	74.14	**733.79**	N
	Minimap2	**94.13**	34.06	50.02	1,733.26	N
	MHAP	86.63	5.31	10.01	9,102.23	N

For synthetic data, BELLA achieves both high recall and precision, consistently among the best. For real PacBio CLR data, BELLA's recall and precision are generally lower than for synthetic data, yet BELLA's F1 results are among the best and show performance stability that competing software often does not. Notably, BELLA has a 49.16% higher F1 score than Minimap2 for *C. elegans* 40X. BELLA's precision and F1 score on real CCS data appreciably outperform competing software. Overall, a good performer on one dataset becomes one of the worst on another, whereas BELLA's F1 score is consistently within 1.7% of the top entry.

Tables 2 and 3 show that BELLA achieves higher F1 score values on synthetic data and real CCS data compared to real CLR data. The way ground truth is generated could explain this behavior. On synthetic data, ground truth comes directly with the dataset itself. Thus, we know exactly where a read in the reference genome comes from and which other reads overlap with it. For real data, the positions of the reads in the reference are determined by mapping the

[1]https://github.com/PacificBiosciences/pbbioconda/issues/46

Table 4: Preliminary assembly results of synthetic datasets. The overlappers' outputs were translated into PAF format and paired with Miniasm [19]. DALIGNER was not run with synthetic datasets. Miniasm produced no output when paired with BLASR and MHAP, and neither tool produced misassemblies.

Dataset	Overlapper	Contigs	N50	Total Length
P. aeruginosa 30X 6,264,404 bp	BELLA	**39**	**299,124**	6,539,838
	MECAT	130	30,078	3,123,445
	Minimap2	118	84,638	6,368,698
V. vulnificus 30X 5,126,696 bp	BELLA	**39**	**201,956**	5,319,876
	MECAT	101	30,132	2,585,336
	Minimap2	105	58,711	4,941,291
A. baumannii 30X 4,335,793 bp	BELLA	**35**	**185,443**	4,486,557
	MECAT	88	33,248	2,234,729
	Minimap2	93	61,967	4,092,102
C. elegans 20X 100,286,401 bp	BELLA	2,792	**35,782**	82,763,749
	MECAT	**366**	10,661	2,398,741
	Minimap2	2,875	19,894	49,724,174

reads to the reference using Minimap2 in its "mapping mode". Intuitively, such a procedure is suboptimal, since there is no guarantee that Minimap2 correctly locates each read. BELLA could potentially find a better set of true overlaps than those identified by Minimap2. Given a uniformly covered genome, we observed that Minimap2 and other long read mappers tend to map reads to "hotspots" within a genome rather than mapping them evenly across the genome. This leads to uneven coverage and overestimation of overlaps by a factor of 1.14×. Recall beyond a certain point on real data would mean that the overlapper also overestimates the overlap cardinality. It is possible that the actual accuracy of BELLA is actually higher for real data. As future work, we plan to investigate these issues in more detail. This bias may not be present when mapping CCS data to the reference, as error rates are lower, making it easier for the mapper to find the correct position on the reference genome.

Table 4 provides insight into the beneficial impact of BELLA in a *de novo* assembly pipeline. Our methods, such as the reliable k-mer strategy, noticeably improve assembly contiguity compared to MECAT and Minimap2. Importantly, Miniasm produced no output when using MHAP or BLASR, leading us to emphasize that an assembler is often built with a particular overlapping tool in mind and programmed to exploit that tool's methods. To fully exploit the potential of BELLA, we plan to build our assembler on top of it.

8 Conclusion

Long-read sequencing technologies enable highly accurate reconstruction of complex genomes. Read overlapping is a major computational bottleneck in long read pipelines for genome analysis, such as genome assembly.

In this paper, we introduced 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 reads. Then we demonstrated the feasibility of the k-mer-based approach using a mathematical model based on Markov chains and showed the generality of such a model. BELLA provides a novel algorithm for pruning k-mers that are unlikely to be useful for overlap detection and whose presence would only add unnecessary computational cost. Our algorithm for reliably detecting k-mers explicitly maximizes the probability of keeping k-mers that belong to unique regions of the genome.

BELLA achieves fast overlap without sketching using sparse matrix multiplication (SpGEMM), implemented using high-performance software and libraries developed for this subroutine. Any novel sparse matrix format and multiplication algorithm would be applicable to overlap detection and would enable further performance improvements. Moreover, our SpGEMM approach is general and flexible enough that it can be coupled with any k-mer selection strategy. Our overlap detection is coupled with our newly developed seed-and-extend banded alignment algorithm. The optimal k-mer seed is chosen by our binning mechanism, which uses k-mer positions within a read pair to estimate the length of the overlap and to "bin" k-mers to form a consensus.

Finally, we developed a new method to separate true alignments from false positives as a function of the alignment score. This method shows that the probability of false positives decreases exponentially as the overlap length between sequences increases.

BELLA achieves consistently high accuracy scores compared to state-of-the-art tools on both synthetic and real-world data, while being performance competitive. BELLA significantly improves assembly results on synthetic data, validating our approach. Future work includes further characterization of real data properties, performance improvements, and the development of a complete *de novo* assembler built on BELLA.

Code: `https://github.com/PASSIONLab/BELLA`.

Acknowledgements

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. Resources from NERSC were used supported by the Office of Science of the DOE under Contract No. DE-AC02-05CH11231. 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

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

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

[3] M. Besta, R. Kanakagiri, H. Mustafa, M. Karasikov, G. Rätsch, T. Hoefler, and E. Solomonik, *Communication-efficient jaccard similarity for high-performance distributed genome comparisons*, in 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE, 2020, pp. 1122–1132.

[4] A. Buluç, T. Mattson, S. McMillan, J. Moreira, and C. Yang, *Design of the GraphBLAS API for C*, in IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2017, pp. 643–652.

[5] A. B. Carvalho, E. G. Dupim, and G. Goldstein, *Improved assembly of noisy long reads by k-mer validation*, Genome research, 26 (2016), pp. 1710–1720.

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

[7] H. 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 (1952), pp. 493–507.

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

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

[10] R. Edgar, *Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences*, PeerJ, 9 (2021), p. e10805.

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

[12] J. Gao, W. Ji, Z. Tan, and Y. Zhao, *A systematic survey of general sparse matrix-matrix multiplication*, arXiv preprint arXiv:2002.11273, (2020).

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

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

[15] G. Guidi, M. Ellis, D. Rokhsar, K. Yelick, and A. Buluç, *BELLA: Berkeley efficient long-read to long-read aligner and overlapper*, bioRxiv, (2020), p. 464420.

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

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

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

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

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

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

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

[23] E. W. Myers, *An O(ND) difference algorithm and its variations*, Algorithmica, 1 (1986), pp. 251–266.

[24] G. Myers, *Efficient local alignment discovery amongst noisy long reads*, in International Workshop on Algorithms in Bioinformatics, Springer, 2014, pp. 52–67.

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

[26] Y. Nagasaka, S. Matsuoka, A. Azad, and A. Buluç, *Performance optimization, modeling and analysis of sparse matrix-matrix products on multi-core and many-core processors*, Parallel Computing, (2019), p. 102545.

[27] Y. Ono, K. Asai, and M. Hamada, *PBSIM: Pacbio reads simulator—toward accurate genome assembly*, Bioinformatics, 29 (2012), pp. 119–121.

[28] A. M. Phillippy, M. C. Schatz, and M. Pop, *Genome assembly forensics: finding the elusive misassembly*, Genome biology, 9 (2008), p. R55.

[29] M. Roberts, W. Hayes, B. R. Hunt, S. M. Mount,

and J. A. Yorke, *Reducing storage requirements for biological sequence comparison*, Bioinformatics, 20 (2004), pp. 3363–3369.

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

[31] C.-L. Xiao, Y. Chen, S.-Q. Xie, K.-N. Chen, Y. Wang, Y. Han, F. Luo, and Z. Xie, *Mecat: fast mapping, error correction, and de novo assembly for single-molecule sequencing reads*, nature methods, 14 (2017), p. 1072.

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