Long Reads Alignment

Giulia Guidi[1,2], Aydın Buluç[2]
{gguidi, abuluc}@lbl.gov
[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy
[2]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, USA

July 7, 2017

1 Problem Statement and General Overview

High-throughput sequencing technologies produce a large number of short and low-quality DNA sequences, called *reads*. These data are used as starting point to reconstruct the whole DNA sequence in a process called *de novo* genome assembly.

Recent advances in this field, such as the Pacific Biosciences Single Molecule Real-Time (SMRT) Sequencing technology, led to higher consensus accuracy, unbiased error distribution and longer reads. These improvements are crucial to assemble high-quality DNA sequences. On the other hand, the SMRT technology is prone to an higher error rate with respect to previous sequencing technology, about 15%. In general, high error rates lead to a significant computational effort in throwing away useless data. The presented work proposes a novel approach to select significant data from the input and recognize overlapping reads pairs to reconstruct the whole DNA sequence.

To give a general overview of our idea, let's consider a pair of reads that share a region of length k (k-mer), as shown in Figure 1. The probability that such region is correclty sequenced on both the reads is approximately $(1-e)^{2\cdot k}$ (summation of Bernoulli trials), where e is the error rate. Then, the probability $P(1, L)$ that at least one k-mer out of L consecutive k-mers being correct in both the sequences is:

$$P(1, L) = 1 - (1 - (1 - e)^{2\cdot k})^L$$

Considering an overlapping region L of 300 base-pairs (bp), with $k = 17$ and $e = 0.15$, we obtain that $P(1, 300)$ is greater than 70%. Starting from these consideration, we decided to base our approach on shared k-mers between reads. However, we have to face the issue derived from having a high error rate. As a matter of fact, an error rate equal to 15% implies that for large genomes, every possible k-mer will be seen at least once, given enough depth (d, sequencer parameter). Consequently, we would need a k-mer look-up table of size 4^k. Furthermore, we have to consider at least a k value equal to 17 as, choosing a smaller k, k-mers are not even unique in large genomes. Using $k = 15$, we can only encode 1 Gbp possibilities whereas, for instance, the *Human* genome is 3 Gbp. To address this issue and identify overlapping regions, we take into account just a reliable set of k-mers.

2 Reliable k-mers

The first computed analysis regards the identification of a reliable set of k-mers (RKS). Our algorithm relies on detecting overlaps between long reads efficiently. We treat the k-mer occurrences in each long read as the feature vector of that read. However, due to high error rates, the number of distinct k-mers

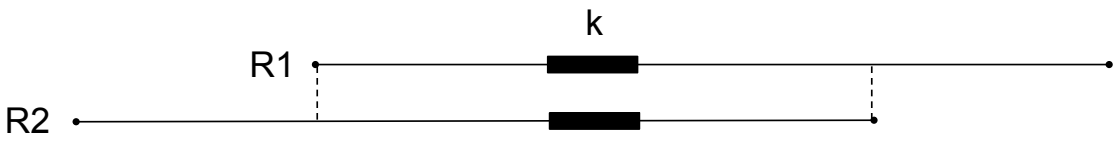

Figure 1: Pair of overlapping reads sharing a region of length k.

in a dataset can be orders of magnitude larger the actual correct k-mers. Keeping all the k-mers in our feature set would not only increase the computational costs and memory requirements, it would also lower our precision.

Ideally, we want k-mers that occur only once in the genome. Multiple occurences of the same k-mer in the genome correspond to repeat regions. If we kept non-unique k-mers in our feature set, they would increase the number of spurious alignments and hence increase the computational costs. Our rationale for ignoring non-unique k-mers comes from the observation that either (a) the repeated region is small enough compared to the length of the read that the unique part of the read can still be used to find overlaps of this read with other reads, or (b) that the repeated region is almost as long as the read itself, in which case there is no benefit in aligning this read to other reads because it does not increase our information about the final genome.

The histogram in Figure 2 shows the distribution of unique, non-genomic and repeated k-mers in the genome, given their frequency in the input set of reads. With *unique* k-mers we define k-mers that occur only once in the genome, while $non-genomic$ k-mers are those that not exist in the genome and *repeated* k-mers those that appear multiple times in the genome. To obtain the plot, we divide k-mers based on their occurrence in the reads and then, having the reference genome, for each occurrence we compute the amount of unique, non-genomic and repeated k-mers. We see by looking at the k-mer histogram that the majority of k-mers in the right tail either occur multiple times or do not occur in the genome at all. Furthermore, Figure 3 points out the percentage of unique k-mers over the total amount of k-mers belonging to a certain frequency; the number above each bar represents the absolute number of unique k-mers for a given frequency. Both Figure 2 and Figure 3 are obtained using the *Escherichia Coli* genome with $d = 30$, $k = 17$ and $e = 0.15$.

The probability of a k-mer being sequenced correctly is approximately $(1-e)^k$ where e is the error rate. If the sequencing depth is d, then observing this k-mer in the input data d times is a very slim $(1-e)^{dk}$. Our analysis here is only correct if no other distinct section of the genome has been morphed into this k-mer by error. We acknowledge that such morphing occurs in practice but the majority of the high-frequency k-mers in the input set are due to correct sequencing if the value of k is chosen appropriately. Take the *Human* genome that is approximatyely 3 Gbp and for the sake of argument, only consider substitution errors. For $k = 17$, it encodes $4^{17} = 16$ billion different k-mers. Assuming independence, every possible k-mer exists in the genome with probability 3/16. The probability that a 17-mer we have seen being the result of off-by-1 error in sequencing is $(3/16) \cdot 17 \cdot e \cdot (1-e)^{17-1} \approx 3e(1-e)^{16}$, whereas the probability of sequencing a 17-mer correctly is $(1-e)^{17}$.

The probability that observing a k-mer that corresponds to a unique (non-repetitive) region $d-1$ times in the input would be approximately $P(d-1) = d(1-e)^{(d-1)k}(1-(1-e)^k)$. To generalize, the probability to observe $d-t$ times is:

$$P(d-t) = \binom{d}{t}(1-e)^{(d-t)k}(1-(1-e)^k)^t$$

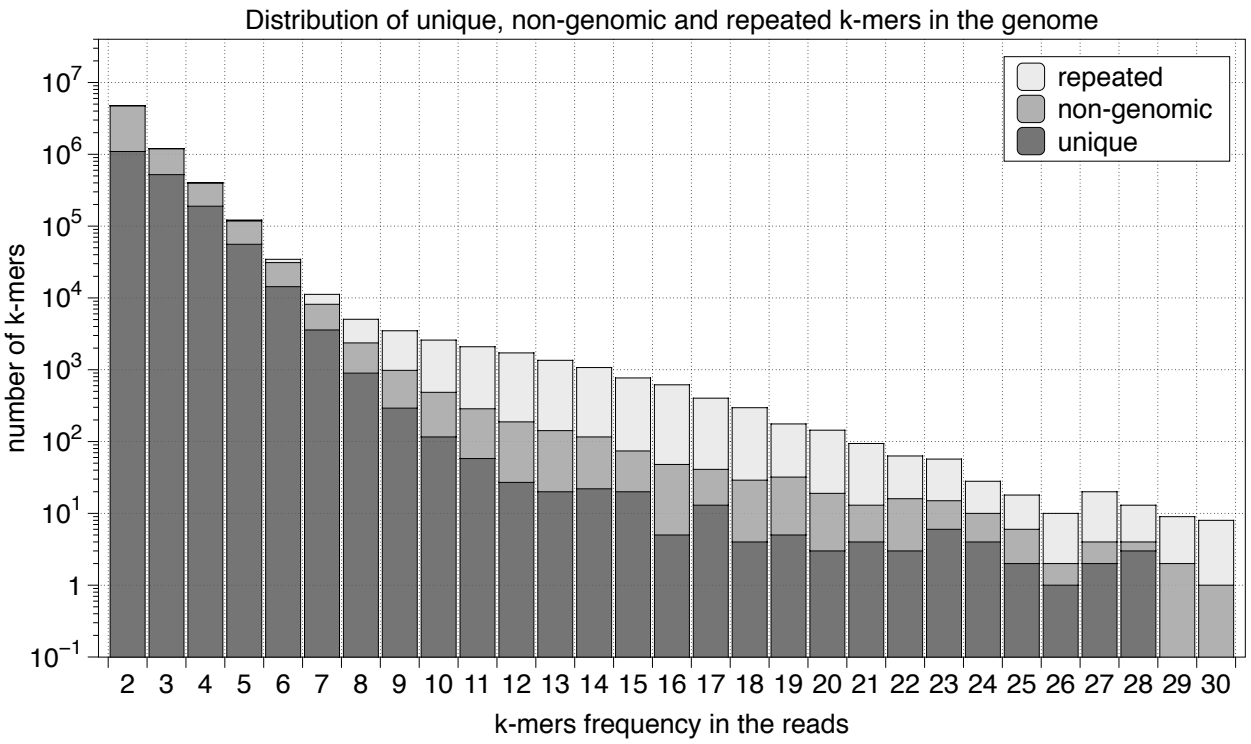

Figure 2: Distribution of unique, non-genomic and repeated k-mers over the total amount of k-mers belonging to a certain frequency.

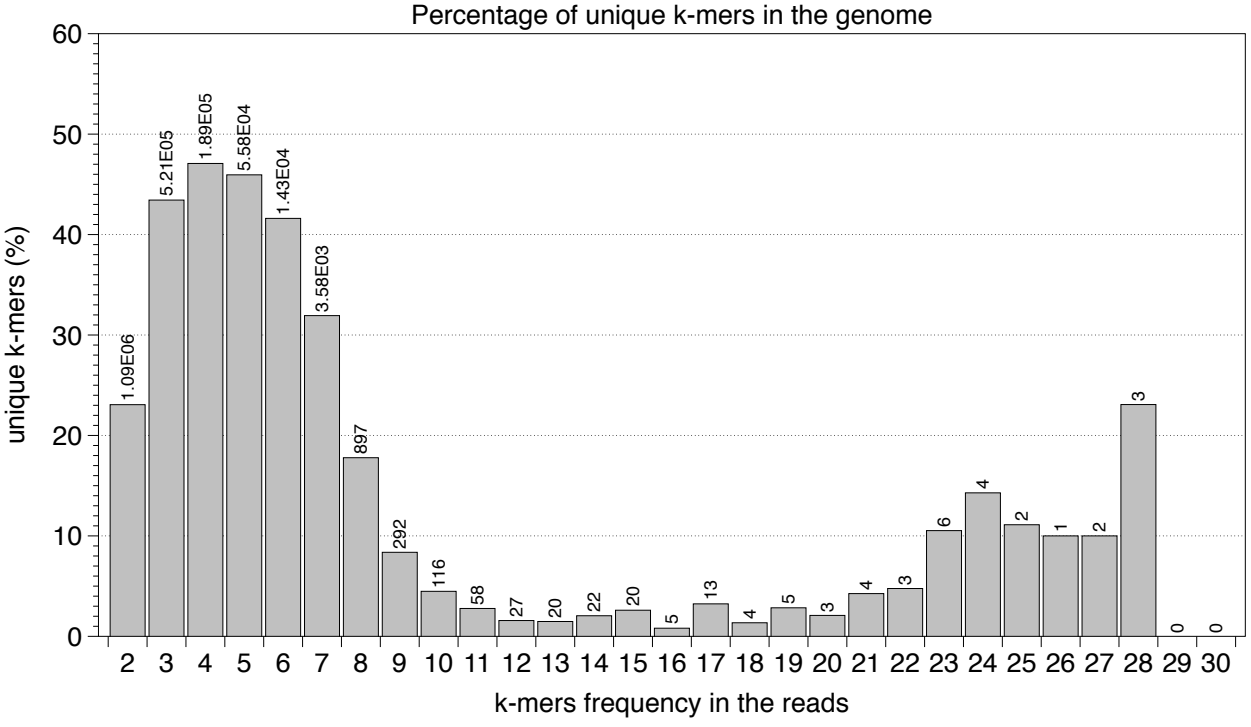

Figure 3: Percentage of unique k-mers over the total amount of k-mers belonging to a certain frequency. The number above each bar represents the absolute number of unique k-mers for a given frequency.

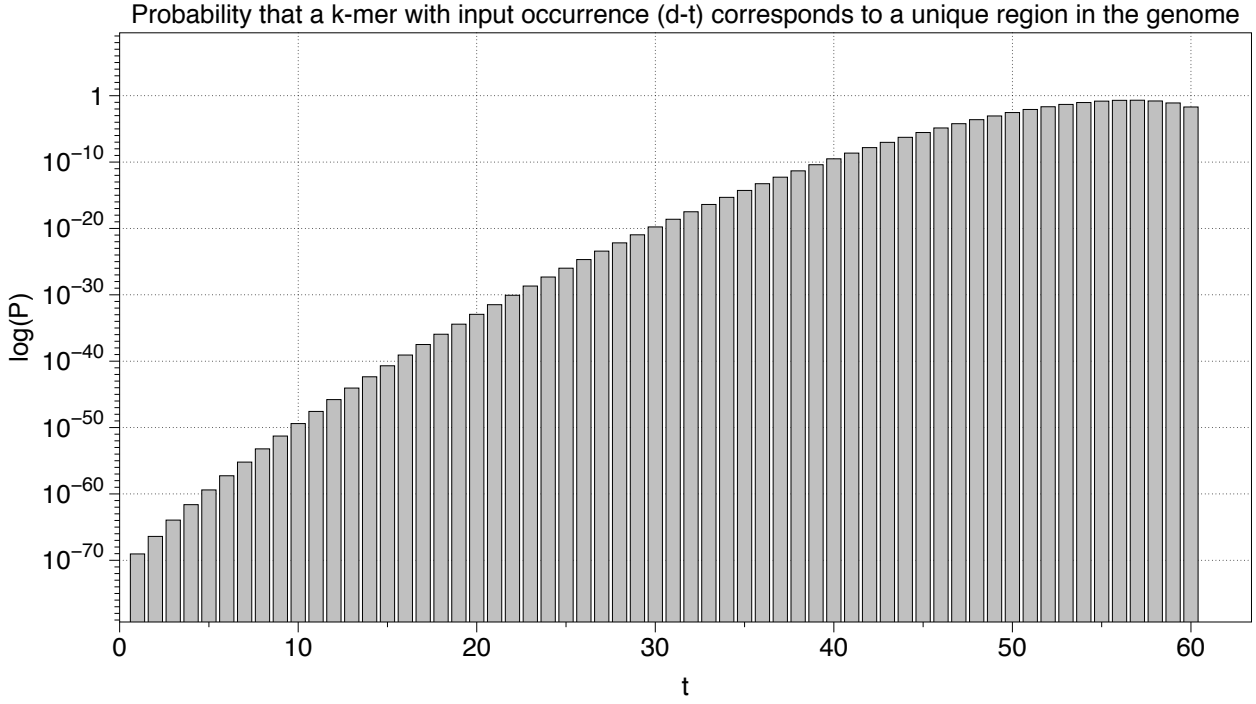

Figure 4: Probability that observing a k-mer that corresponds to a non-repetitive region $d - t$ times in the input for $d = 60$, $k = 17$ and $e = 0.15$.

Figure 4 shows the probability that a k-mer with input occurrence $d - t$ corresponds to a unique region in the genome given $d = 60$, $k = 17$ and $e = 0.15$.

3 Proposed algorithm

To identify the overlapping reads pairs, the proposed approach exploits sparse matrices. This because sparse matrices express the data access patterns in a concise and clear manner, allowing better organization of computation and generality. The column-by-column algorithm we use is equivalent to the k-mer index table concept in HipMer and many other assembler. The first step consists in the creation of a sparse matrix where row are represented by the reads and columns by the set of reliable k-mers. The cell (i, j) contains the k-mer id and the positions of that k-mer in the considered read. This matrix is used as starting point for the construction of a feature vector in finding alignments among the reads. Consequently, we create the transpose of the first one obtaining a $kmers - by - reads$ matrix, as shown in Figure 5.

After that, exploiting the column-by-column algorithm we multiply these two sparse matrices obtaining a resulted matrix, as illustrated in Figure 6. The final sparse matrix allows us to keep track of all the read pairs that share some k-mers and the corresponding positions on both the reads. Here, the cell (i, j) consists in a vector containing the k-mers ids are and the corrisponding positions in the read i and read j. The generality of this approach comes from being able to use the same algorithm to also keep track of distances between shared k-mers. In fact, we are interested in keeping track of the distance between shared k-mers as it can provide more evidence of overlap.

To detect which reads pairs show some evidence of potential overlap, we implemented the following method, Figure 7. For each read pairs, (a) we compute the distance between each pair of consecutive k-mers both on read i and read j. For each k-mers pairs, (b) we select the shortest distance between the

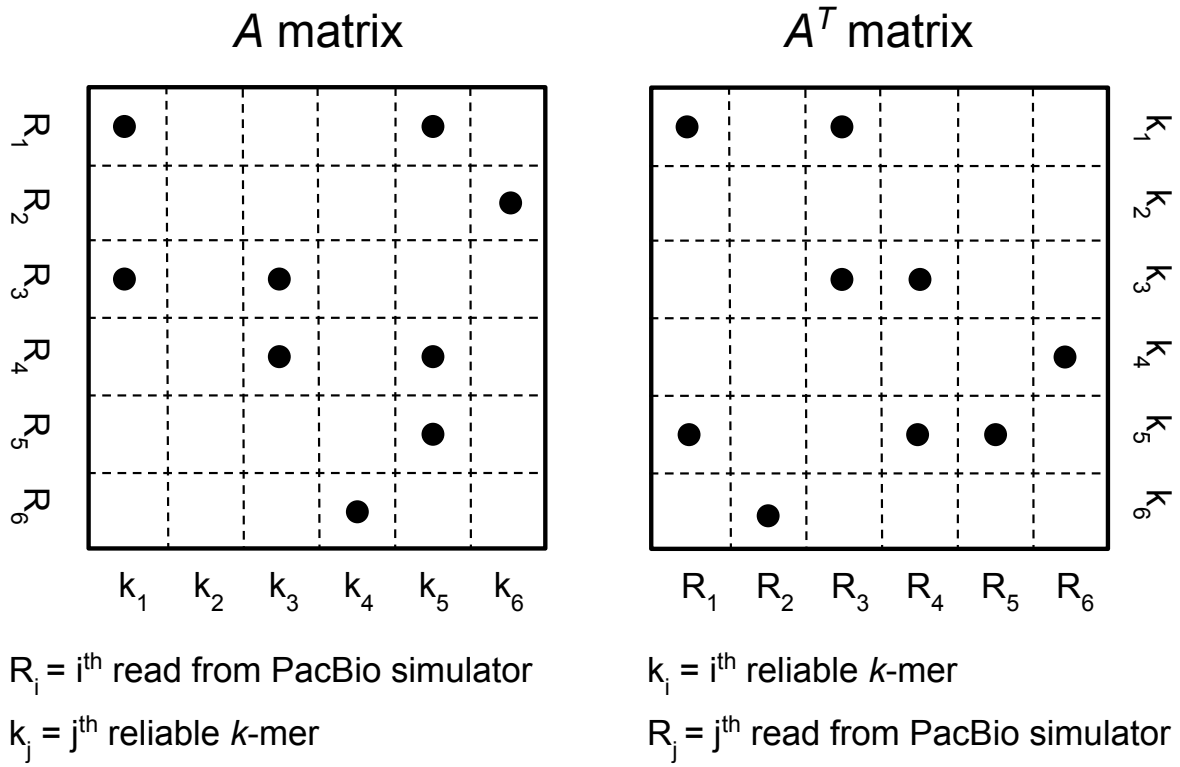

Figure 5: Sparse matrix $reads - by - kmers$ on the left and its transpose on the right.

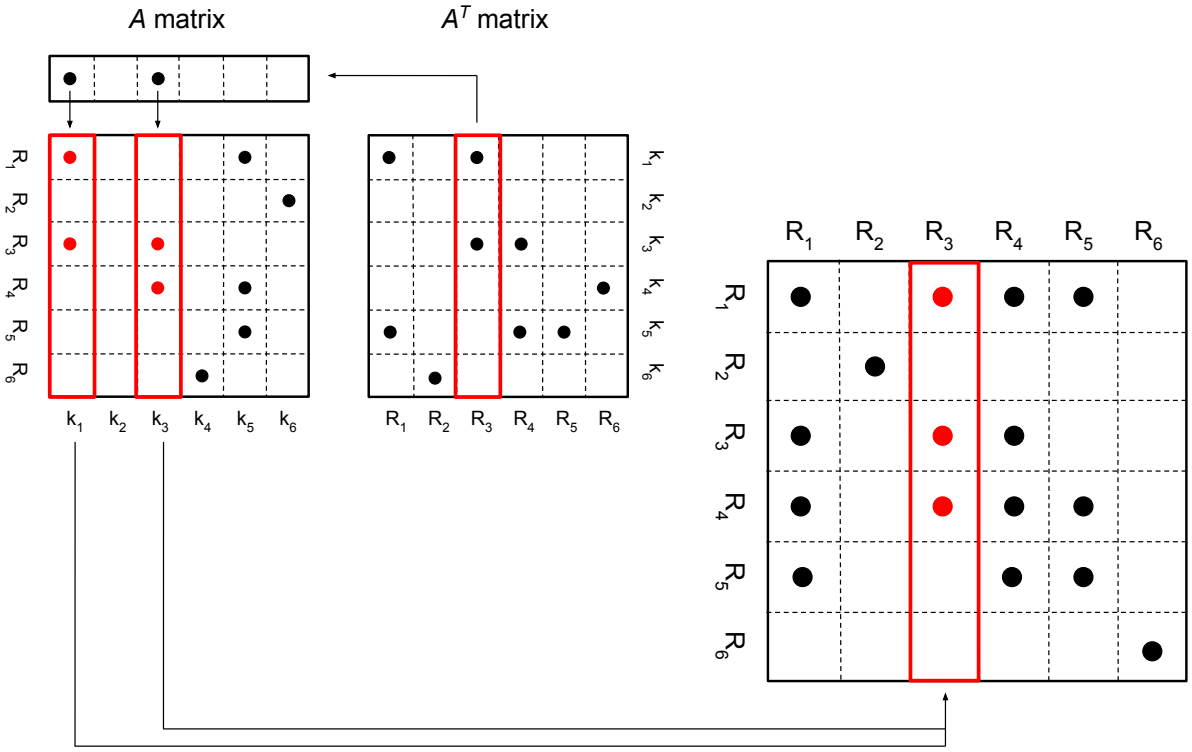

Figure 6: Sparse matrix multiplication exploiting a $column - by - column$ algorithm cite.

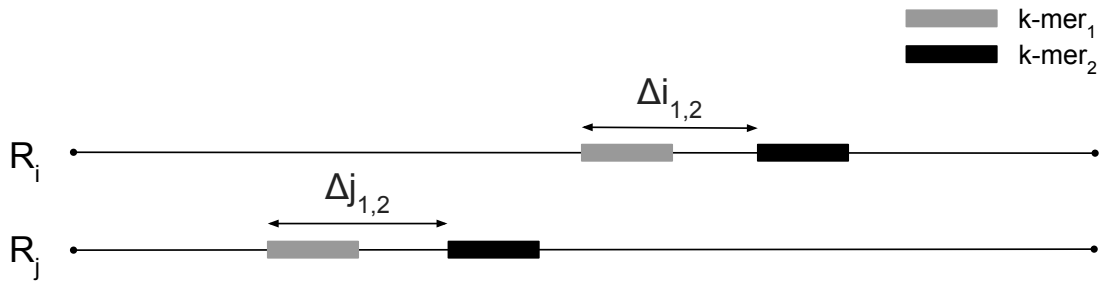

Figure 7: Reads pair sharing two k-mers.

one on read i and on read j and then, given the probabilities of insertion and deletion (that are given by the sequencer), (c) we compute the maximum length of the shortest one (dividing with $1 - p_{deletion}$) and the minimum length of the longest one (dividing with $1 + p_{insertion}$). If the former is greater than the latter, we consider that k-mers pair as evidence of potential overlap. In order to be as conservative as possible and be sure to remove just those reads pairs that certainly not overlap, we decide to keep into account all the reads pairs that present at least one evidence of potential overlap among their k-mers pairs.

3.1 Evaluation

To evaluate our approach we measured the true positive rate as measure of the proportion of positives that are correctly identified as such, that means as the ratio between the number of true overlapping reads pairs we detected and actual number of true overlapping pairs. The false positive rate is calculated as the ratio between the number of non-overlapping reads pairs identified as true overlapping reads pairs (that means the false positives derived from our approach) the total number of actual non-overlapping reads pairs in the entire set of reads. Both the true positive rate and the false positive one were related to the computational cost of the overlap-layout-consensus algorithm that as a time complexity equal to $O(R)$ where R is the number of reads pairs (Figure 8 and Figure 9), and finally compared to each other through a Receiver Operating Characteristic (ROC) curve curve as shown in Figure 10. In the plots, the parameter S represents the minimum number of shared k-mers that two reads share.

We see from the plots that, in the most conservative case than means $S = 1$, our approach achieve a true positive rate of about 91%. In later stages, we could exploit transitivity between reads pairs to detect the missing pairs. For instance, two reads could be interconnected through a k-mer excluded from the reliable range.

3.2 Light version

The previous explained version of the algorithm is implemented on single node and it has very high memory requirements. Due to the memory bound, this version does not allow us to test our approach with genomes greater than the *Escherischia Coli* one. Consequently, we decided to implement a *light* version of the algorithm.

In this second version, the cell (i, j) of the matrices before the multiplication contains just the integer value that represents the k-mer *id* in the k-mers dictionary; while the cell (i, j) in the resulted matrix is an integer value representing the number of shared k-mers between the two considered reads. This change allowed us to test our approch with different genomes, as shown in Table 1.

Due to this modification, we cannot apply the distances filter previously described as to use it we

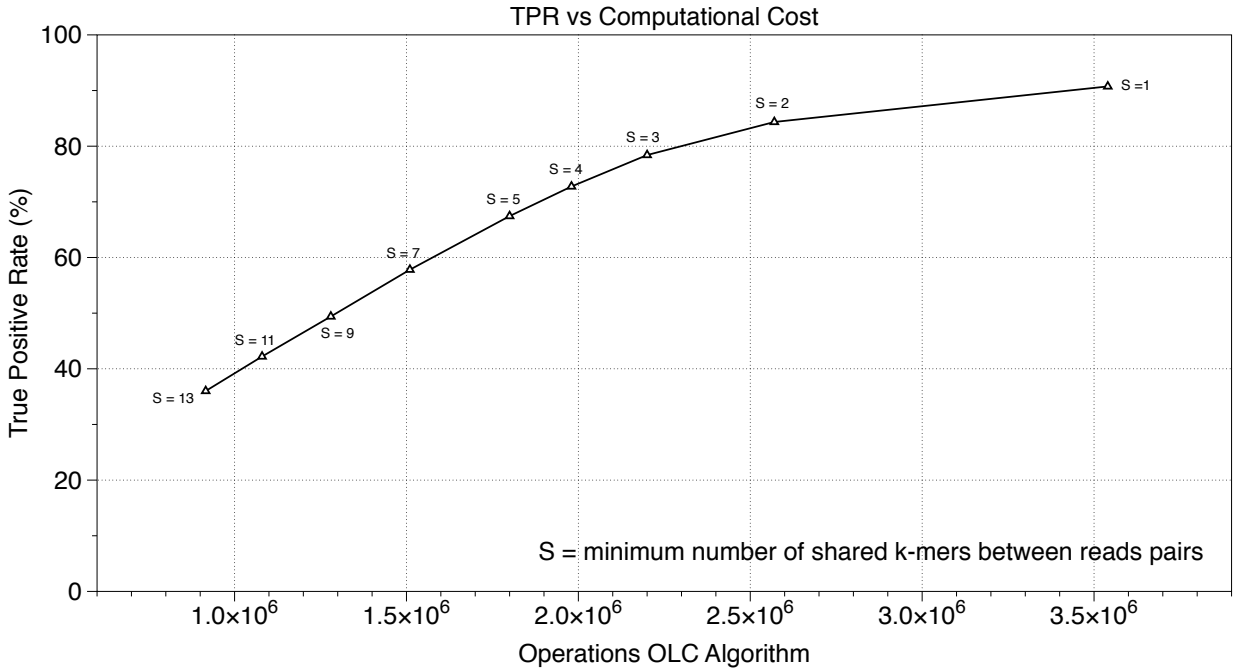

Figure 8: True positive rate compared to the computational cost of the $Overlap-Layout-Consensus$ (OLC) algorithm, the x-axis is in log scale.

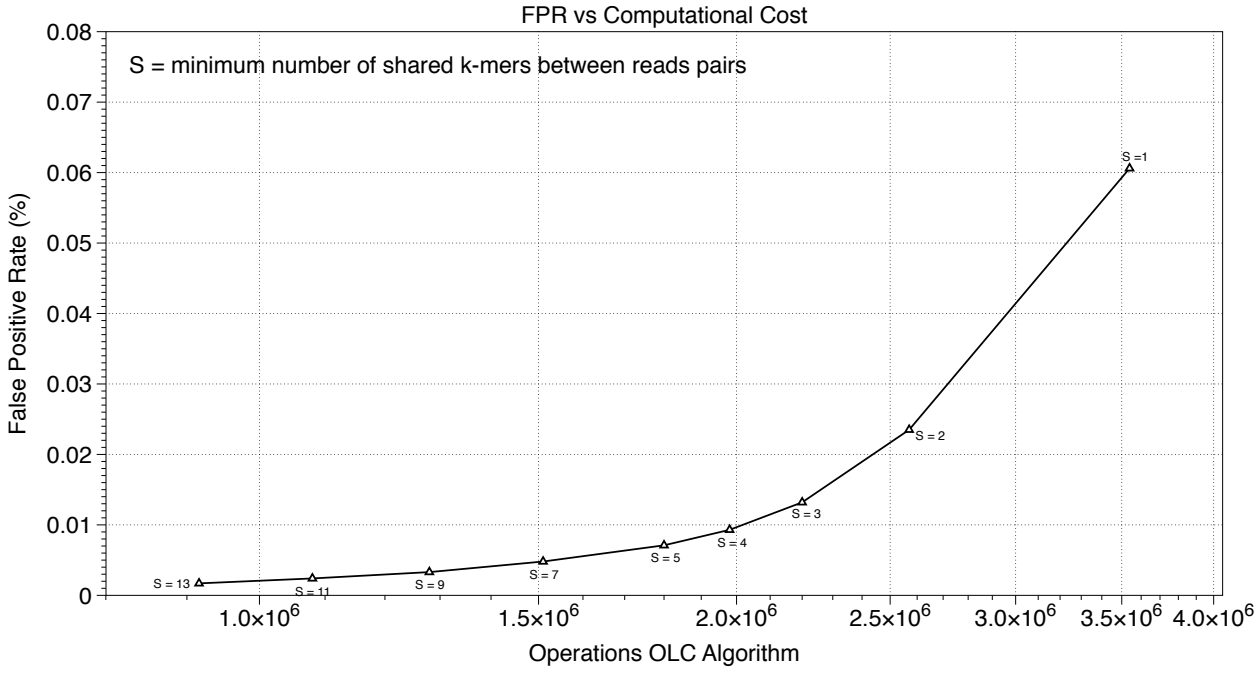

Figure 9: False positive rate compared to the computational cost of the $Overlap-Layout-Consensus$ (OLC) algorithm, the x-axis is in log scale.

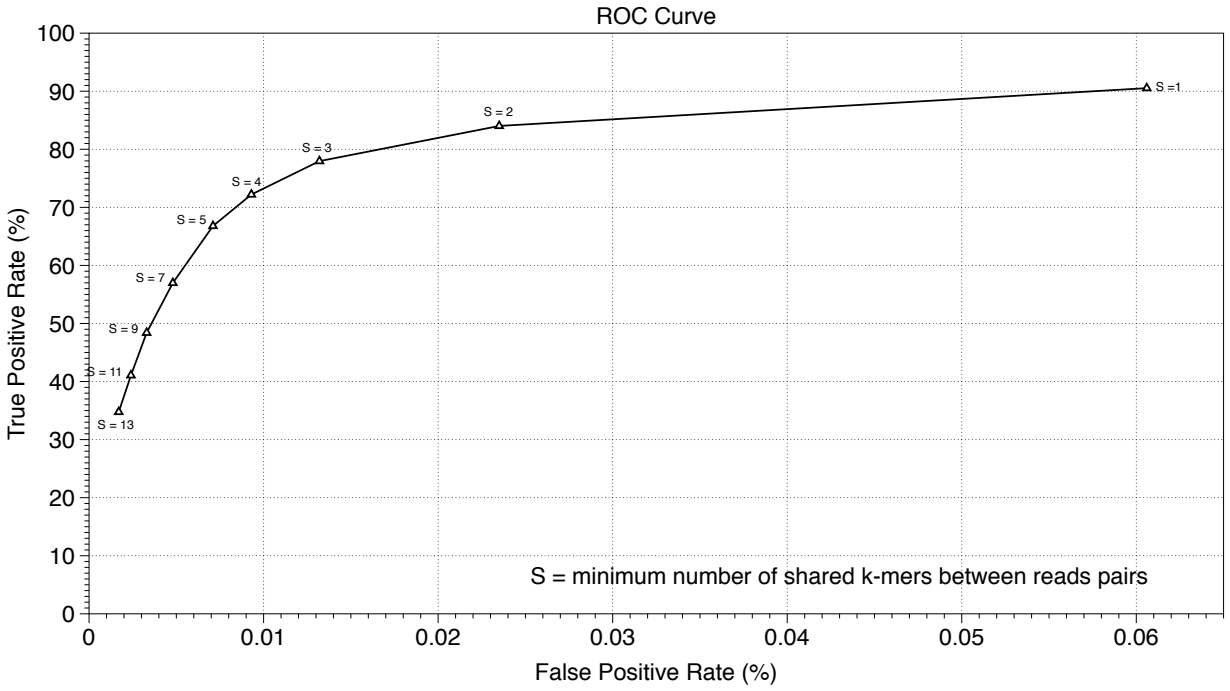

Figure 10: Receiver Operating Characteristic (ROC) curve.

Genome	Sequence size (MB)	Reliable k-mers (million)	RAM Usage (GB)
E. Coli	4.7	1.63	1.3
C. Elegans Chr. 3	14.1	4.97	7.6
C. Elegans Chr. 4	17.8	6.87	8.3
E. Caballus Chr. 31	25.5	16.17	8.8

Table 1: Reliable k-mers and RAM usage of the *light* algorithm for different genomes.

Genome	Sensitivity (%)	Precision (%)
E. Coli	90.77	62.58
C. Elegans Chr. 3	91.14	3.80
C. Elegans Chr. 4	91.53	3.50
E. Caballus Chr. 31	94.16	5.91

Table 2: *Light* version: sensitivity and precision for different genomes.

need additional information, such as the k-mers positions in the reads. However, we saw that removing the filter does not change significantly our TPR (called also sensitivity) and precision. The simple selection of reliable k-mers and the application of the sparse matrix multiplication guarantee the best results (that means almost the same maximum we were able to achieve with the previous version of the algorithm) both for sensitivity and precision. The sensivity and precision for different genomes are shown in Table 2. So far, our precision is very low that means we are potentially wasting a significant amount of computiation in the later stages. Implement an intermediate version that can allow us to compute the potential overlap between two considered reads and try to use this information to throw away false positives..

In the MECAT paper (http://www.biorxiv.org/content/early/2016/12/15/089250), the authors consider just reads with length at least equal to 5000 bp (Figure 2 in their paper) and define that two reads has the correct pairwise relationship if they overlap for more than 2000 bp (Table 2 in their paper). So, we generated the initial set of reads with pbsim setting 5000 bp as minimum length and setting the threshold to consider a pair as correct (both in the initial set and when we filter our pairs) equal to 2000 bp. In this case, for instance for the *E.Coli*, we obtain a recall of 99% and a precision of 34%. Even if the precision is lower with respect to our previous implementation, the number of reads pairs on which compute the local alignment is smaller (about 1.23 millions less than before, but we're still wasting a lot of computation). Considering than in a region of length 2000, we expected to see about 8 shared k-mers (computed as $2000 \cdot (1-e)^{34}$), setting 4 as minimum number of shared k-mers to be more conservative, we obtain 98.14% of recall and 68.6% of precision, setting 8 we got 93.94% of recall and 78.8% of precision. The results obtained applying MECAT thresholds for difference genomes are reported in Table 3. The term *computational saving* in Table 3 is defined as $1 - \frac{TP+FP}{R^2}$, where TP are the true positives, FP are the false positives and R is number of reads.

Add references.

Genome	Sensitivity (%)	Precision (%)	Computational saving (%)
E. Coli	98.15	68.60	99.68
C. Elegans Chr. 3	97.67	30.25	99.76
C. Elegans Chr. 4	97.89	31.06	99.81
E. Caballus Chr. 31	98.94	16.83	99.76
E. Coli	93.94	78.87	99.74
C. Elegans Chr. 3	92.26	73.04	99.91
C. Elegans Chr. 4	92.80	70.79	99.92
E. Caballus Chr. 31	96.13	31.89	99.88

Table 3: *Light* version: sensitivity and precision for different genomes computed using MECAT threshold (minimum read length = 5000 bp and minimum overlap = 2000 bp) with number of shared k-mers S equal to 4 (upper part of the table) and 8 (lower part).